VSAM Demystified

Learn the latest VSAM functions and manage VSAM data

Understand, evaluate, and use VSAM properly

Learn problem determination and recommendations

Mary Lovelace
Jose Dovidauskas
Alvaro Salla
Valeria Sokal

Redbooks

ibm.com/redbooks

International Technical Support Organization

VSAM Demystified

August 2022

SG24-6105-02

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

Third Edition (August 2022)

This edition applies to z/OS Version 1 Release 13 DFSMS (product number 5694-A01).

Contents

Notices

Trademarks

Preface

Virtual Storage Access Method (VSAM) is one of the access methods used to process data. Many of us have used VSAM and work with VSAM data sets daily, but exactly how it works and why we use it instead of another access method is a mystery.

This book helps to demystify VSAM and gives you the information necessary to understand, evaluate, and use VSAM properly. This book also builds upon the subject of Record Level Sharing and DFSMStvs. It clarifies VSAM functions for application programmers who work with VSAM. The practical, straightforward approach should dispel much of the complexity associated with VSAM. Wherever possible an example is used to reinforce a description of a VSAM function.

This IBM® Redbooks® publication is intended as a supplement to existing product manuals. It is intended to be used as an initial point of reference for VSAM functions.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Mary Lovelace is a Consulting IT Specialist at the International Technical Support Organization. She has experience with IBM in large systems, storage and Storage Networking product education, system engineering and consultancy, and system support. She has written IBM Redbooks about Tivoli® Storage Productivity Center, Tivoli Storage Manager, Scale Out Network Attached Storage, and IBM z/OS® storage products.

Jose Dovidauskas is a Senior IT Specialist in IBM Brazil. He has 22 years of experience in storage products. He has worked at IBM for 35 years. His areas of expertise include the IBM GDPS® family of solutions, DFSMS/IBM MVS™, z/OS, and other storage-related software products and offerings. Jose has contributed to other Redbooks in the DFSMS/MVS area, Tivoli Storage Productivity Center for Replication for z/OS, and Copy Services.

Alvaro Salla is an IBM retiree. He worked in IBM for more than 30 years in large systems. He currently teaches ITSO workshops on z/OS performance around the world. Alvaro coauthored many IBM Redbooks publications, and spent many years teaching about large systems, from the S/360 to the IBM S/390®. He has a chemistry engineering degree from the University of Sao Paulo, Brazil.

Valeria Sokal is a Business Partner from Brazil. She has 15 years experience as a z/OS system programmer and eight years as an IT Architect. She has co-authored and contributed to many z/OS IBM Redbooks publications.

Thanks to the following people for their contributions to this project:

Bob Haimowitz
Emma Jacobs
International Technical Support Organization, Poughkeepsie Center

Stephen Branch
Nelson Fincher
Larry Law

Terri Menendez
Helen Witter
DFSMS Development

Janet Sun
Manager, Software Engineering Rocket Software

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-6105-02
for VSAM Demystified
as created or updated on August 23, 2022.

August 2022, Third Edition (minor update)

This revision reflects the addition, deletion, or modification of new and changed information described below.

Changed information

Added reference, corrected block size, and updated recommended block size in “Blocking logical records in physical records” on page 9. Change bars are turned on.

March 2013, Third Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

- The book was at the DFSMS/MVS V1.4 level. It has been updated to the z/OS DFSMS V1.13 level.

Changed information

- The code samples included in the appendix were not retested during this project. They were left in the book to be used as a starting point for your own applications.

1

VSAM basics

This chapter reviews the concepts and terminology that are associated with VSAM. It explains how a VSAM data set is different from other data set types. It also addresses the various types of VSAM data sets including what makes them unique, and describes how VSAM data is stored and accessed.

This chapter includes the following sections:

1.1 VSAM functions by release level

In the early 1970s, Virtual Storage Access Method (VSAM) was introduced by IBM as a collection of three data set organizations: Sequential, indexed, and direct access. These organizations are combined with access method software and utilities used on mainframe IBM operating systems.

The word *virtual* means only that VSAM was introduced at approximately the same time as the initial IBM virtual storage operating systems OS/VS1 and OS/VS2. Since then VSAM has been continually improved and enhanced.

DFSMS/MVS V1R5 was the last independent release. With z/OS V1.3, DFSMS became an integral component of z/OS. This book covers the z/OS V1.12 DFSMS level.

Table 1-1 shows the VSAM enhancements by z/OS DFSMS release level and where to find more information in this book.

Table 1-1 VSAM functions by release level

Function	Available since	Section described in
Data compression for VSAM KSDS	DFSMS/MVS V1 R2	4.4.16, “Data compression” on page 170
Extended addressability for VSAM KSDS	DFSMS/MVS V1.R3	2.4, “Extended addressability (EA)” on page 65
VSAM RLS	DFSMS/MVS V1.R3	Chapter 5, “VSAM Record Level Sharing” on page 209
RLS support of extended addressability of VSAM KSDS	DFSMS/MVS V1 R4	Chapter 5, “VSAM Record Level Sharing” on page 209
SMB for VSAM KSDS	DFSMS/MVS V1 R4	“System managed buffering (SMB)” on page 164
Data striping for VSAM LDS through an SPE	DFSMS/MVS V1 R5	4.4.17, “VSAM data striping” on page 177
Extended addressability for all VSAM data sets	DFSMS/MVS V1 R5	2.4, “Extended addressability (EA)” on page 65
Data striping and multi-layering for all VSAM data set	DFSMS/MVS V2 R10	4.4.17, “VSAM data striping” on page 177
Ignore DEFINE parameters IMBED, REPLICATE	z/OS V1R3 DFSMS	4.4.8, “Index options” on page 143
Ignore DEFINE parameters KEYRANGE, ORDERED	z/OS V1R3 DFSMS	“System managed buffering (SMB)” on page 164
SMB retry capability for DO access bias	z/OS V1R3 DFSMS	“System managed buffering (SMB)” on page 164
SMB support for IBM AIX®	z/OS V1R3 DFSMS	“System managed buffering (SMB)” on page 164
RLS cache CI size bigger than 4 KB	z/OS V1R3 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
VSAM data striping support for data sets with REUSE attribute	z/OS V1R3 DFSMS	4.4.17, “VSAM data striping” on page 177

Function	Available since	Section described in
RLS caching all or some of the data in a Coupling Facility cache structure	z/OS V1R3 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
Maximum Volume Count, volumes added dynamically to a data set without manual intervention and taking down the application	z/OS V1R3 DFSMS	“Implementing VSAM data striping” on page 179
Dynamic Cache Reassignment	z/OS V1R3 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
DFSMS Data Set Separation, to allocate data sets in distinct physical DASD controllers	z/OS V1R3 DFSMS	“I/O wait time for VSAM data sets” on page 196
Real addresses greater than 2 GB available for all VSAM data sets	z/OS V1R4 DFSMS	2.8, “VSAM and 64 bits in real storage” on page 78
Media Manager the unique VSAM I/O driver, exception for improved control interval processing (ICIP)	z/OS V1R4 DFSMS	1.2, “What is VSAM?” on page 4
RLS system-managed duplexing rebuild process	z/OS V1R4 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
RLS Coupling Facility duplexing	z/OS V1R4 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
More than 255 extents for SMS-managed VSAM data sets	z/OS V1R7 DFSMS	Chapter 4, “VSAM performance” on page 127
VSAM RLS: Data buffers above the bar	z/OS V1R7 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
Enhancements in VSAM problem determination: VDUMP option in F CATALOG command	z/OS V1R8 DFSMS	Chapter 7, “VSAM problem determination and recovery” on page 345
VSAM RLS DIAG command to find latch contentions	z/OS V1R8 DFSMS	Chapter 7, “VSAM problem determination and recovery” on page 345
SMF data of VSAM RLS usage of 64-bit data buffers	z/OS V1R8 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
Speeding up the SMSVSAM address space with TERMINATESERVER command	z/OS V1R8 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
Enhanced RLS recovery	z/OS V1R8 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
Limit the SMB buffer size for DO in DATACLASS	z/OS V1R9 DFSMS	“System managed buffering (SMB)” on page 164
Health Checks for VSAM RLS: Latch contention and SPOF in the SHCDS	z/OS V1R9 DFSMS	Chapter 5, “VSAM Record Level Sharing” on page 209
EAV: Storing VSAM data sets beyond 64K cylinders addressing	z/OS V1R10 DFSMS	Chapter 2, “Managing your VSAM data sets” on page 45
Trap for data component and Trace enhancements	z/OS V1R11 DFSMS	Chapter 7, “VSAM problem determination and recovery” on page 345
Enhancement in SMB for DO: Buffers for index allocated that are based on SMBVSP value or data set size	z/OS V1R11 DFSMS	“System managed buffering (SMB)” on page 164

Function	Available since	Section described in
Support for VSAM striping in RLS mode	z/OS V1R12 DFSMS	5.1, "Introducing VSAM RLS" on page 210
Dynamic trace for record management	z/OS V1R12 DFSMS	7.1, "VSAM problem determination hints and tips" on page 346

1.2 What is VSAM?

VSAM is one of several access methods in z/OS. It applies only to data stored on direct access storage devices (DASDs). An access method is re-entrant code that is loaded at IPL time at PLPA and contained in DFSMSdfp, a component of the z/OS DFSMS product. This access method makes it easier for an application to run an I/O operation (moving data between an I/O device and memory).

1.2.1 What is an access method?

An access method is an optional function to simplify the logic of the application when it requests I/O operations to z/OS. An application in EXCP mode does not require an access method to help run I/O operations. An access method is implemented through a set of programs that belong to DFSMS that are in PLPA or in the SMSVSAM private address space. The access method runs under the application task that starts it through a branch instruction (no PSW change status). For the VSAM RLS option, the invocation is done through a Program Call (PC), which is a cross memory instruction.

The address of the access method program is stored in the DCB/ACB by the Open service routine. This address is used when the application wants to access data along one I/O operation. VSAM has these major access method functions:

- Blocking by storing several logical records in one physical block
- Buffering by managing the buffer pool efficiently
- An access method understands the way logical records are organized on the 3390 tracks. It is in charge of writing virtual channel programs for accessing such records.
- Starting the I/O driver through an SVC (usually SVC 00 - EXCP)
- Synchronize through the Wait function, the running task with the end of the I/O operation (if needed).
- Start I/O error recovery.

1.2.2 VSAM access types

An application has several patterns for requiring data access depending on business needs. They are called access types. There are three access types in VSAM:

- Random (or direct): This is also called direct access. The logical record must be located by the use of a search argument that comes from the application. There is no search argument connection between two consecutive logical record accesses.
- Sequential: The entire data set is processed (for Read or Write), one logical record after the other. The application does not need to provide any search argument. The access method keeps a place holder always pointing to the record for the next request. The access method can implement a Read Look Ahead technique. This technique loads logical records that are not yet required by the application program in the buffers.

- Skip sequential: A combination of the two previous types of access. The application randomly provides one search argument and, from the located logical record on, all records are processed sequentially. An example is sequentially processing all the customers of a bank branch office in a data set that contains all the customers of the bank.

1.3 VSAM functions

There are two major parts to VSAM: Catalog management and record management.

1.3.1 Catalog management

VSAM maintains extensive information about data sets and direct-access storage space in an integrated catalog facility (ICF) catalog. A catalog describes data set attributes and indicates the volumes on which a data set is located.

For more information about the Catalog Management functions, see *Enhanced Catalog Sharing and Management*, SG24-5594, and *ICF Catalog Backup and Recovery: A Practical Guide*, SG24-5644.

1.3.2 Record management

The record management part of VSAM contains the access method code. In this book, VSAM refers to VSAM record management, unless otherwise stated.

VSAM is used to organize records into four types of data sets: Key-sequenced (KSDS), entry-sequenced (ESDS), linear (LDS), and relative record (RRDS and VRRDS). The primary difference between the types of VSAM data sets is the way that their records are stored and accessed.

1.4 VSAM terminology and concepts

Before addressing VSAM in detail, review some VSAM concepts and terms that are implemented by VSAM constructs. These concepts are used throughout the book. The concepts range from the smallest VSAM entity, the logical record, and finish with the largest VSAM entity, the sphere. Other topics include splits, buffering, control blocks, and other basic concepts.

1.4.1 Logical record

A logical record is a unit of information that is used to store data in a VSAM or in any other data set type.

An application sees only logical records, which are its logical units of I/O data that are being transferred. A logical record is made up of a set of bytes containing a logical description of an item that is processed by the application program. This item can be a customer with all his or her information, or an employee with all associated data (name, serial number, department).

The logical record is designed by the application programmer from the business model. The logical record is divided in fields, such as the name of the item, its key, the address, and the account information.

The application through a GET (or a READ, depending on the language) requests that a specific logical record be moved from the I/O device to virtual memory to be processed. Through a PUT (or a WRITE), the specific logical record is moved from virtual memory to an I/O device.

A logical record can be of a fixed size or a variable size, depending on the business requirements. VSAM supports both, depending on the specific organization. An example of a variable logical record is account information about a bank customer where all debit and credit transactions are individually described at the end of the register.

Remember: The terms *logical record* and *record* are used interchangeably.

Figure 1-1 shows LISTCAT output where you can find information about a logical record, including AVGLRECL, MAXLRECL, REC-TOTAL, REC-DELETED, REC-INSERTED, and REC-UPDATED.

```
DATA ------- DAWN.KSDSEXG.DATA
      IN-CAT --- MCAT.SANDBOX.R9.VSBOX11
ATTRIBUTES
        KEYLEN-----------------8     AVGLRECL-------------300
BUFSPACE-----------10240     CISIZE--------------4096
        RKP--------------------0     MAXLRECL-------------300
EXCPEXIT----------(NULL)     CI/CA----------------192
STATISTICS
        REC-TOTAL---------321595     SPLITS-CI-----------6466
EXCPS--------------82682
        REC-DELETED-------173530     SPLITS-CA-------------42
EXTENTS----------------4
        REC-INSERTED-------45123     FREESPACE-%CI---------10     SYSTEM-TIMESTAMP:
        REC-UPDATED--------56016     FREESPACE-%CA---------10
X'B3ECB2FDD72E7785'
        REC-RETRIEVED----3186326     FREESPC--------610766848
      ALLOCATION
        SPACE-TYPE---------TRACK     HI-A-RBA-------736100352
        SPACE-PRI-----------3744     HI-U-RBA-------203685888
        SPACE-SEC------------624
```

Figure 1-1 A partial LISTCAT containing HURBA, HARBA, splits, CI size, CI and CA splits

Key field

An important field in the logical record is the *key.* Its contents can be used to retrieve the specific logical record in a random type of access. It identifies the item that is associated with the logical record. Keys, as an example, can be the customer number or the parts number.

There can be multiple key fields in the same logical record. To differentiate between keys that are used in VSAM objects, the key that is used in a base cluster is called a primary key or base key. The key that is used in an alternate index is called an alternate or secondary key. For more information, see 1.4.8, “Alternate indexes” on page 15.

In VSAM key sequenced organization, a record must have a unique, embedded fixed-length primary key in the same position within each logical record. Primary keys can be a minimum of one byte and a maximum of 255 bytes.

Unlike the primary keys, which must be unique, identical alternate can occur in more than one logical record. The alternate index allows the search with a given alternate key to read all base cluster records containing this alternate key.

Figure 1-1 on page 6 shows Listcat output where you can find information about Key Field such as KEYLEN and Relative Key Position (RKP).

Ways to identify logical records

In VSAM, there are three ways to identify a logical record for a random search. All these ways must be finally translated into a CCHHR format (the format, that the IBM FICON® channel understands). This translation is done by VSAM.

VSAM has these ways to identify logical records:

- Key field
- Relative byte address (RBA)
- Relative record number (RRN)

Key field

An important field in the logical record is the *key.* Its contents can be used to retrieve the specific logical record in a random type of access. It identifies the item that is associated with the logical record. Keys, as an example, can be the customer number or the parts number.

There can be multiple key fields in the same logical record. To differentiate between keys that are used in VSAM objects, the key that is used in a base cluster is called a primary key or base key. The key that is used in an alternate index is called an alternate or secondary key. For more information, see 1.4.8, “Alternate indexes” on page 15.

In VSAM key sequenced organization, a record must have a unique, embedded fixed-length primary key in the same position within each logical record. Primary keys can be a minimum of one byte and a maximum of 255 bytes.

Unlike the primary keys, which must be unique, identical alternate can occur in more than one logical record. The alternate index allows the search with a given alternate key to read all base cluster records containing this alternate key.

Relative byte address (RBA)

The RBA of a logical record is the offset of its first byte from the beginning of the data set.

The first logical record in a VSAM data set has an RBA of zero. The second logical record has an RBA equal to the length of the first record, and so on. VSAM returns to the application the RBA of the stored logical record. The RBA of a logical record in VSAM includes the free space and control information that is stored sequentially before this logical record. RBA might change when records are added, deleted, or changed in size.

With compressed data sets, the RBA for compressed records are not predictable. Therefore, access by RBA is not suggested for normal use. There are two important RBA values, which are kept in the VSAM catalog:

- High used RBA (HURBA) is the RBA of the first available byte for inclusion at end of the used part of a VSAM data set. HURBA is not rounded to the CA size.
- High allocated RBA (HARBA) is the RBA of the last byte in the allocated part of a VSAM data set. HARBA is rounded to the CA size.

Figure 1-1 on page 6 shows Listcat output where you can find information about the numeric values of HURBA and HARBA.

Relative record number (RRN)

RRN is the relative number of a logical record in a VSAM-specific organization (RRDS). If a logical record has an RRN of 3, it is the fourth logical record in the data set.

1.4.2 Physical record

In general terms, a physical record is a set of logical records. A physical record is also called a physical block or simply a block.

To understand what a physical record is, review the 3390 count key data (CKD) track format. Every track starts with an index marker, a gap, a home address (describing the cylinder and track number), and a gap. After that are the physical records. Each track has a gross amount of almost 57 KB, and in a 3390 volume there are 15 tracks per cylinder. The number of cylinders is variable depending on the 3390 model. Figure 1-2 shows the track format.

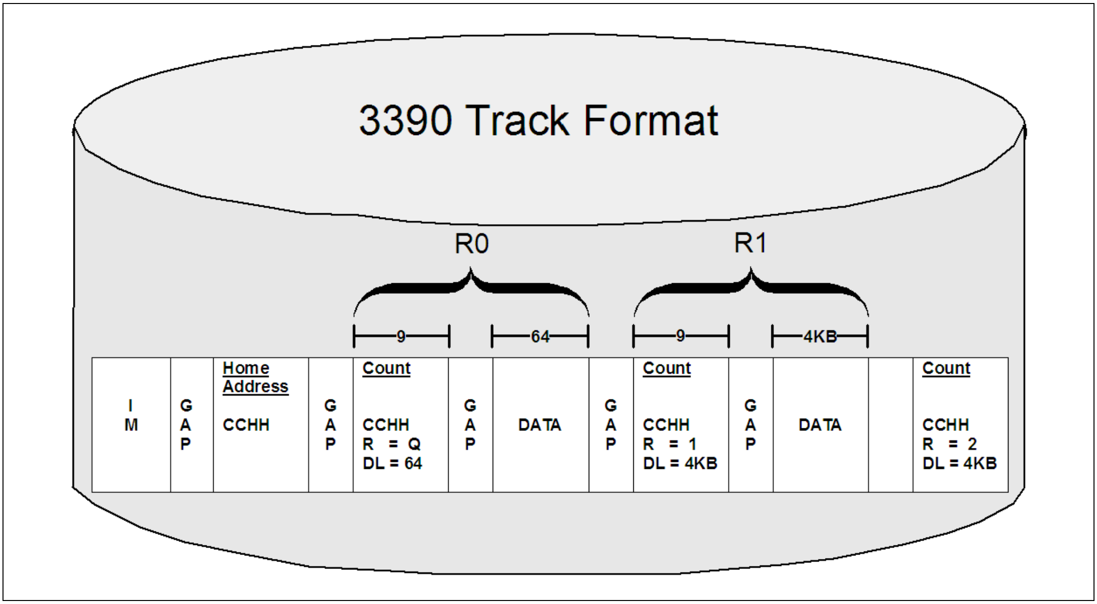


Figure 1-2 3390/3380 track format

Then, a physical record is formed by a Count, a gap, an optional Key, a gap and the Data. The Key field is only present in non-indexed VTOC and partitioned data set (PDS) directories. The use of the Key field causes bad performance. The Count part (8 bytes) indicates the address of the physical record in the track and its Data length, which can vary. The Data size is derived by the access method at the time the data set is defined through the BLKSIZE parameter. In VSAM, all physical records in one data set have the same size. The Data portion of the physical record is usually a set of logical records packed together. These records are transferred from or to memory by just one Read or Write channel command word (CCW). The act of placing together logical records in one physical block is called blocking.

Channel command word (CCW) and CCHHR

The access method is in charge of writing a channel program to run an I/O operation. The channel program describes the details of I/O operation itself to the FICON channel.

The channel program describes the following details:

- Whether it is a read or a write
- How many bytes are going to be moved along the I/O operation

- The address of the I/O buffer in central storage
- The address of the physical blocks in a 3390 volume. This address is made of:
  - Cylinder number, indicated by two bytes (CC)
  - Track (or head) number in the previous described cylinder, indicated in two bytes (HH)
  - Physical record number in the previous described track, in the previous cylinder, indicated in one byte (R)

If the FICON channel is not running in zHPF architecture, this information is passed to the channel through a set of CCWs. This set is called a channel program.

Blocking logical records in physical records

Appendix E of the manual *z/OS DFSMS Macro Instructions for Data Sets*, SC23-6852 provides VSAM space utilization information for selected device types, as well as supported Control Interval sizes, to assist with making the most efficient use of storage in relation to physical block sizes.

The advantages of blocking are as follows.

- Better utilization of the 3390 track. The larger the block size, the fewer gaps there are. To minimize gaps, just one block per track (56 KB) seems ideal. However, the access methods support only a maximum of 32 KB for DASD. The best BLKSIZE for the 3390 track is 26624 bytes (26 KB), with two blocks per track. You can also use 18432 bytes (18 KB) with three blocks per track and 97% utilization of the track.
- For sequential type of access, the larger the block size, the more efficient the I/Os are. This improvement is because larger block size requires fewer CCWs to move data. The recommendation for sequential processing matches the previous item, that is, 26 KB or 18 KB.

Larger blocks have these drawbacks:

- Larger blocks demand larger buffers in virtual (and real) memory to keep the data.
- For random access that requires one specific logical records, logical records that are not needed are moved to and from memory along an I/O operation.

Today, real 3390 devices are not common. The new DASD controllers logically mimic 3390 CKD devices in fixed-block architecture (FBA) disks. In this architecture, all blocks (still separated by gaps) have the same size, and there are no counts. You care about the BLKSIZE to optimize space, if the 3390 track does not really exist because the DASD controller simulates exactly the 3390 track format. It knows for a specific BLKSIZE how many physical records fit on the real 3390 track, and prevents the channel program from going beyond this amount. Therefore, use a half track (26 KB) BLKSIZE (for sequential processing) to better use the 3390 track capacity.

There are two techniques for an access method to create physical records:

- First, format it with zeros and later fill the Data portion with real data. This technique is used for random data creation or to create software end-of-file (EOF) marks. These logical EOFs are Data parts full of zeros. The DASD controller signals a physical EOF to the channel and the channel to the access method, when they find a Counter with the Data Length field equal to zero.
- Format and load with real data at same time. This technique is used for sequential data creation.

To read or write a physical record, there is a buffer in virtual (and central) storage. Several buffers of same size are called a buffer pool. The size of the data (physical record) affects the amount of used virtual (and central) storage. The Count part of the physical record is not moved to the buffer pool.

As a processor runs instructions, an I/O FICON channel runs CCWs such as Read or Write. In a channel program (a set of CCWs), there is one Read or Write CCW per each block. The block is the smallest piece of data that are transferred in one I/O operation. However, if you have several buffers in central (and virtual storage), you can transfer several blocks in just one channel program. This idea speeds up the sequential I/O processing.

When the VSAM data set is in extended format, the size of the physical block is increased by 32 blocks. For more information, see 1.6, “Data striping” on page 35.

VSAM can control information along with logical records in the data portion of the physical record, as you can see in the control interval VSAM concept.

BLKSIZE information should not be defined by the installation. VSAM calculates that information by using the size of the CI and the optimization of the 3390 track space.

1.4.3 Control interval

Control interval (CI) is a unique VSAM concept. It is the fundamental building block of every VSAM data set. A CI is a contiguous area of DASD volume track that VSAM uses to store data logical records and control information that describes the records in the CI. A CI is the unit of information that VSAM transfers between the DASD device and the central storage during one I/O operation. If the CI is formed by several physical blocks, these blocks are read or written in a single I/O operation (with several Read or Write CCWs). Whenever a logical record is retrieved from a DASD device, the entire CI containing the record is read into a VSAM I/O buffer in virtual storage. The logical record is then transferred from the VSAM buffer to a user-defined logical record buffer or work area.

The CI size can be from 512 byes to 32 KB.

The size of the physical record (block) is determined by VSAM based on the CI size, to better use the 3390 track. VSAM considers whether the data set is in extended format. For more information, see 1.6, “Data striping” on page 35.

For example, a 3390 without EF uses these block sizes:

- With a CI Size of 22528 bytes, the block size is 5632 bytes
- With a CI Size of 24576 bytes, the block size is 24576 bytes

For random access, use small data CIs to avoid bringing unneeded logical records into memory. For sequential access, define large CIs to decrease the number of I/O operations. However, there are other considerations that affect the decision about the CI size. A CI consists of these components:

- Logical records stored from beginning to end
- Free space, for data records to be inserted into or lengthened

- Control information, which is made up of two types of fields. It consists of one control interval definition field (CIDF) per CI, and several record definition fields (RDFs) describing the respective logical records:
  - A CIDF is a 4-byte field. It contains information about the amount and location of free space in the CI.
  - An RDF is a 3-byte field. It describes the length of records. For variable size, there is one RDF for each logical record. For fixed length records there are two RDFs, one with the length and other with how many with that length.

  For more information about the structure of the control information fields, see *z/OS DFSMS Using Data Sets*, SC26-7410.

Figure 1-3 shows the general format of a CI. The CI components and properties can vary depending on the data set organization. For example, a VSAM LDS does not contain CIDFs and RDFs in its CI. All of the bytes in the LDS CI are data bytes. For more information, see 1.5, “VSAM data set organizations” on page 19.

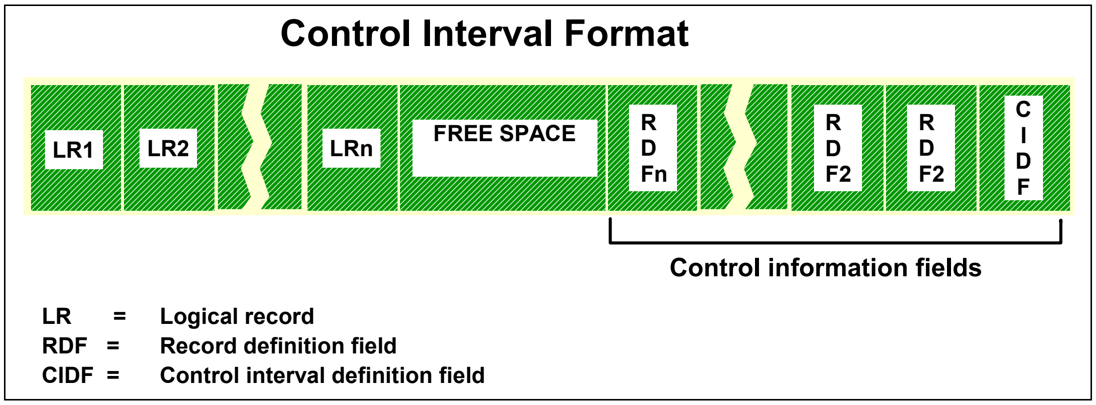

Control Interval Format
LR1
LR2
LRn
FREE SPACE
RDFn
RDF2
RDF2
CIDF
Control information fields
LR = Logical record
RDF = Record definition field
CIDF = Control interval definition field

Figure 1-3 General format of a control interval

The size of CIs can vary from one data set to another. However, all the CIs within the data component of a particular cluster must be of the same length. For more information about the data component, see 1.4.6, “Components” on page 13.

You can request the CI size by using the AMS DEFINE command. You can also allow VSAM determine the CI size, or specify an SMS data class, which uses the CISIZE defined by your storage administrator.

Figure 1-1 on page 6 shows Listcat output where you can find information about control intervals such as CISIZE and CI/CA.

1.4.4 Spanned records

Spanned records are logical records that are larger than the CI size. To have spanned records, the data set must be defined with the SPANNED attribute in IDCAMS DEFINE command at the time it is created. Spanned records can extend across (span) control interval boundaries. The RDFs describe whether the record is spanned or not.

A spanned record must always begin on a control interval boundary and fills one or more control intervals within a single control area. A spanned record cannot share the CI with any other records. The free space at the end of the last segment is therefore not filled with the next record. This free space can be used only to extend the spanned record.

The size limit of a spanned logical record is the size of the control area. For more information, see 1.4.5, “Control area” on page 12.

Spanned records are needed when the application requires long logical records. You can use spanned record in the data component of an Alternate Index (AIX) cluster. Use them when the logical record is large because of many primary keys corresponding to a secondary key. If spanned records are used for KSDS, the primary key must be within the first control interval. For more information, see 1.4.8, “Alternate indexes” on page 15.

Figure 1-4 shows the layout of the spanned record.

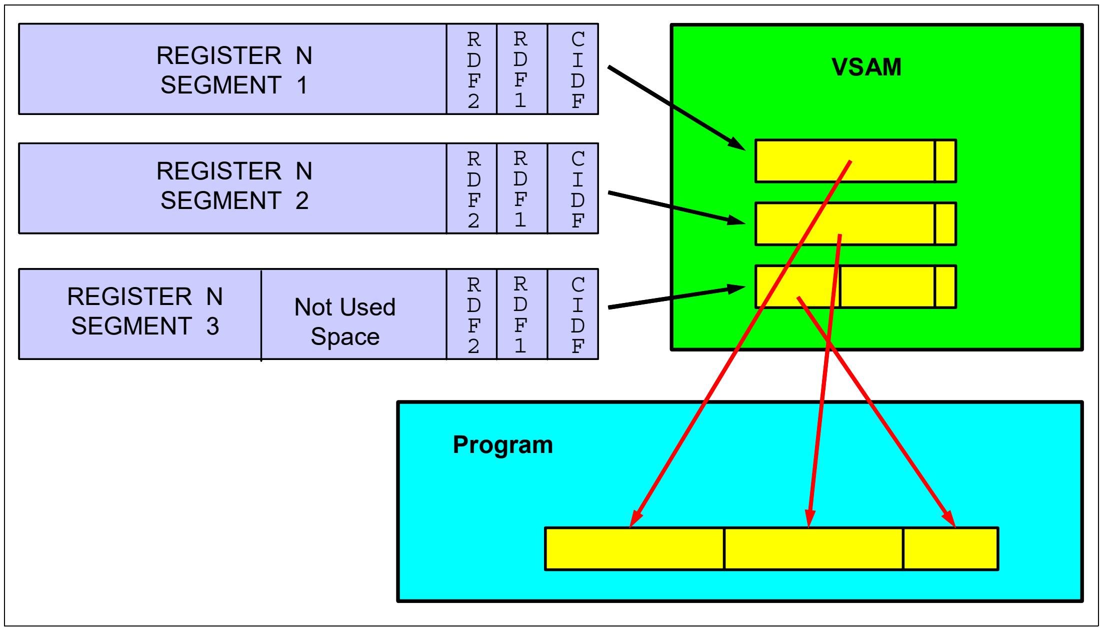


Figure 1-4 Spanned record layout

1.4.5 Control area

Control area (CA) is also a concept that is unique to VSAM. A CA is formed by two or more CIs put together into fixed-length contiguous areas of direct-access storage. A VSAM data set is composed of an integer number of CAs. In most cases, a CA is the size of a 3390 cylinder (15 tracks). The minimum size of a CA is one track. The maximum size of a CA is 16 tracks when the data set is stripped. For more information about data stripping, see 1.6, “Data striping” on page 35.

The CA size is implicitly defined when you specify the size of a data set at definition time. There is not a keyword to set the CA size.

Control Areas are needed to implement the concept of splits that are caused by the inclusion of records. The size of a VSAM data set is always a multiple of its CAs. VSAM data sets are extended in units of integer CAs. A spanned record cannot be larger than a CA.

During sequential access, it is not possible to cross CA boundaries within a same channel program (I/O operation). Figure 1-5 shows one CA and its CIs.

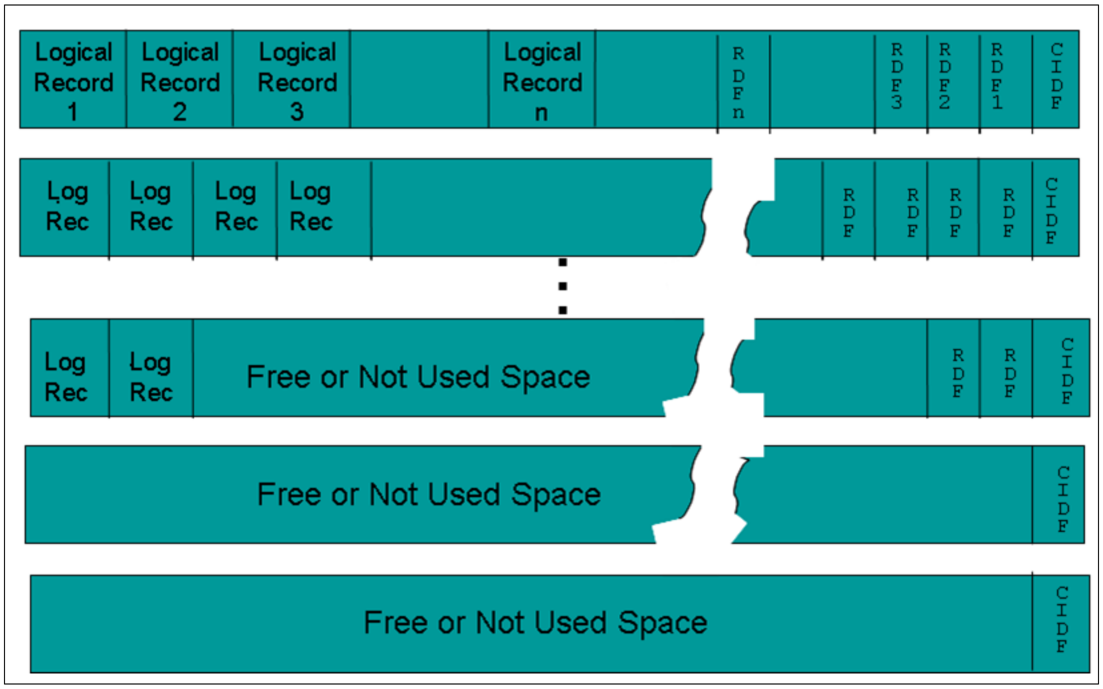


Figure 1-5 Control area with control interval

1.4.6 Components

A component is an individual part of a VSAM data set. Each component has a name, an entry in the catalog, and an entry in the VTOC. The KSDS and VRRDS organizations have data and index components. ESDS, RRDS, and LDS organizations have only data components. A component can be multi-extent and multi-volume. However, VSAM does not allow JCL concatenation between two VSAM components. You receive the IEC161I message with RC 68.

There are two types of components:

- Data component
- Index component

Data component

The data component is the part of a VSAM data set, alternate index, or catalog that contains the data records.

Index component

The index component is a collection of logical records that contains data key fields and the address (RBA) of the data logical records that contain these keys. These data keys are taken from a fixed defined field in each data logical record. The keys in the index logical records are compressed (rear and front) and the RBA pointers are compacted.

Using the index, VSAM can randomly retrieve a record from the data component when a request is made for a record with a certain key. The key determines the record's position in the data set. Because there are random and sequential types of access, VSAM divides the index CIs into two parts (sequence set and index set). This configuration speeds up the search for a key.

A VSAM index can consist of more than one level. Each level contains pointers to the next lower level. It looks like a B-Tree data structure. All the index CIs contain only one logical record that is made of fields. These fields are also called index entries.

Sequence set

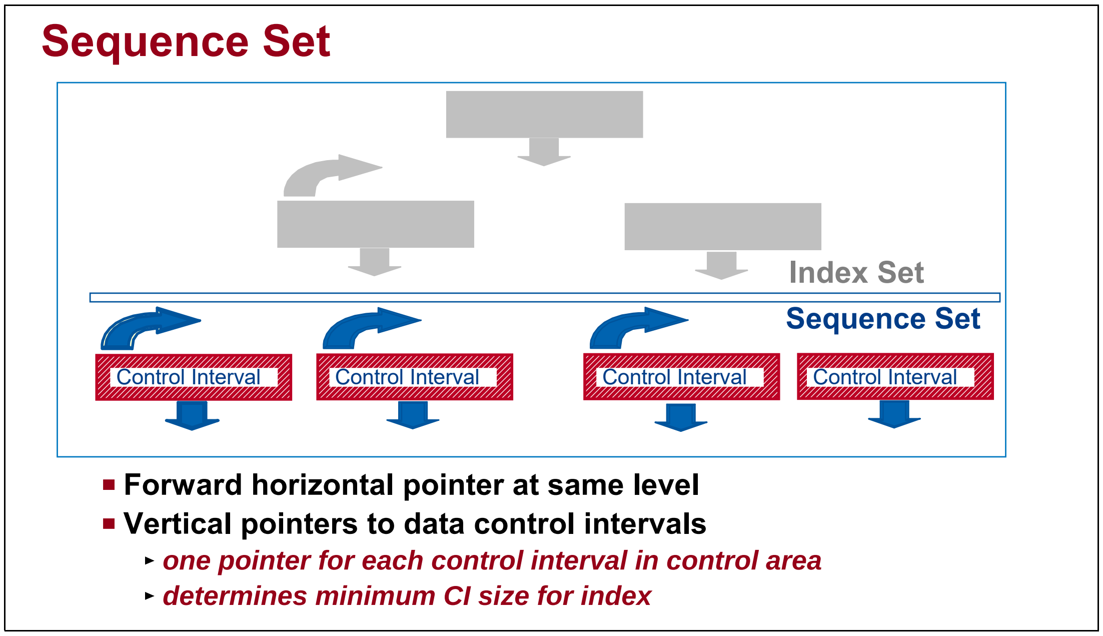
The sequence set is the part of the index component that is used by VSAM for sequential access of the records that are contained in the data component.

The sequence set is the lowest level of index CIs, and directly points (through an RBA) to the data CI in the data component CA. There is one index entry per data component CI and so one index CI (with just one logical record) for each data component CA.

The sequence that is set in the index component provides the RBA of the record in the data component CI. This RBA is translated into CCHHR, which is used in one CCW in the I/O operation to identify the physical record where the data component CI starts. CCHHR contains the following information:

- CC is the cylinder number
- HH is the track number
- R in the physical record number

Every logical record in the sequence set contains pointers and high key information for each data CI. It also contains horizontal pointers from one sequence set CI to the next higher keyed sequence set CI (Figure 1-6). These horizontal pointers are needed because the possibility of splits. Splits make the physical sequence different from the logical collating sequence, which is based on the key values.



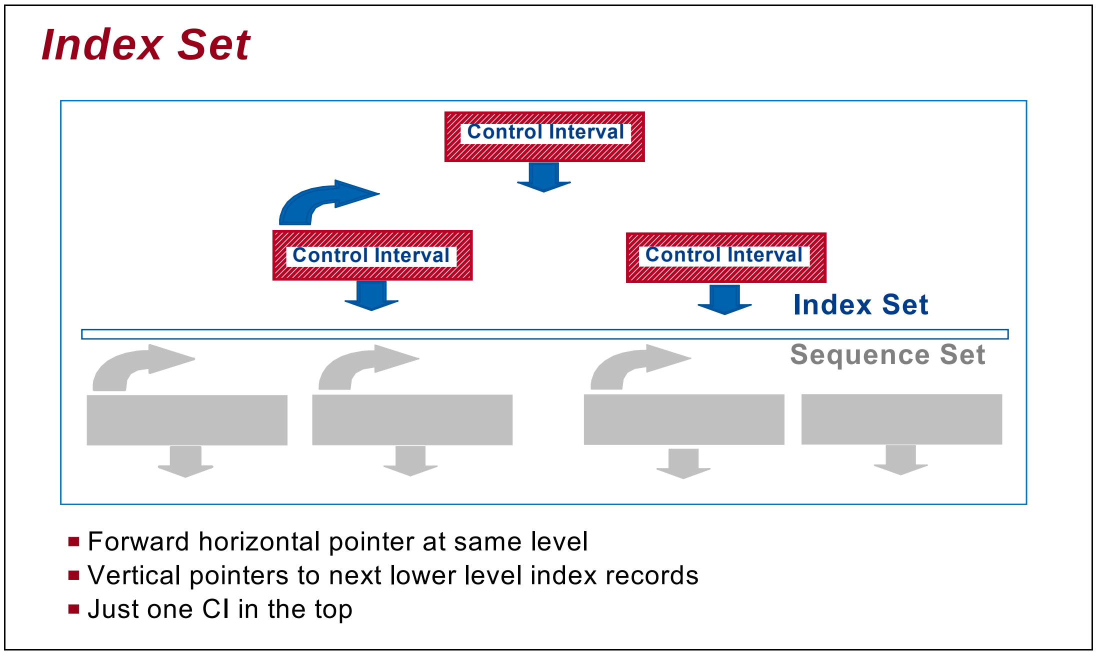
Sequence Set

- **Forward horizontal pointer at same level**
- **Vertical pointers to data control intervals**
  - ***one pointer for each control interval in control area***
  - ***determines minimum CI size for index***

Figure 1-6 General format of a sequence set

Index set

The index set is used for random access. The index set is the index component without the sequence set. If there is more than one sequence set CI, VSAM automatically builds another index CI in another higher level. Each CI in the index set contains pointers and high key information for CIs in the next lower level of the index (Figure 1-7).



Index Set

- Forward horizontal pointer at same level
- Vertical pointers to next lower level index records
- Just one CI in the top

Figure 1-7 General format of an index set

The highest level of the index always contains a single index CI.

1.4.7 Clusters

A cluster is a set of related VSAM components (maximum two). For a KSDS, a cluster is the set of a data component and an index component. The concept of cluster simplifies VSAM processing. It provides a way to treat index and data components as a single entity (for example, by using JCL), with its own cataloged name. You can also give each component a name. You can then process the data portion separately from the index portion.

The RRDS, ESDS, and LDS VSAM organizations are considered to be clusters without index components. To be consistent, they are given cluster names that are normally used when processing the data set.

What in VSAM corresponds to an MVS data set varies. If you are referring to the cluster name (in a DD statement for example), the cluster corresponds to the data set. If you are referring to a component, it corresponds to the data set. When the term data set is used instead of cluster, know that it sometimes refers to a component and sometimes to a cluster. The exception to this terminology rule is when you are explaining the concept of alternate index and sphere.

1.4.8 Alternate indexes

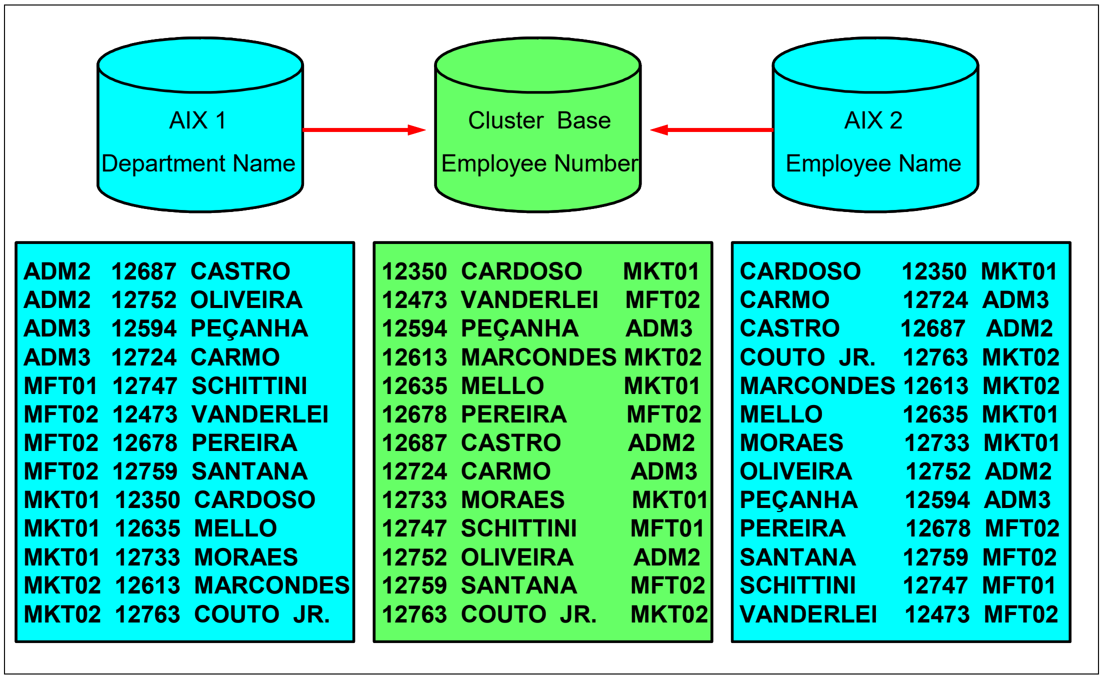
Alternate indexes (AIXs) allow logical records of a KSDS or of an ESDS (in this context, a base cluster) to be accessed sequentially and directly by more than one key field. AIX eliminate the need to store the same data in different sequences in multiple clusters for various applications. Each alternate index is a KSDS cluster that consists of an index component and a data component.

Any field in the base cluster record can be used as an alternate key. It can also overlap the primary key (in a KSDS), or with any other alternate key. The same base cluster can have several AIXs varying the alternate key. The alternate key must follow all the key requirements as described in 1.4.1, “Logical record” on page 5 except for uniqueness. That is, the alternate key can have repeated values. Also, there can be more than one primary key value for the same alternate key value. As an example, the primary key is an employee number and the alternate key is the department name. Obviously, the same department name can have several employee numbers. See Figure 1-8.

The records in the data component of an AIX cluster contain the alternate key value and all the primary keys corresponding to the alternate key value (pointers to data component in the base cluster). The primary keys in the logical record are in ascending sequence within an alternate index value. If you have many primary keys per alternate key, consider defining the AIX as spanned and compressed. Remember that a logical record cannot cross control areas.

By using the IDCAMS utility program, you can define and then create AIX when the BLDINDEX command is specified. An AIX is defined only after its associated base cluster has been defined. It can be built only after its base has been loaded with at least one record.

The BLDINDEX command causes a sequential scan of the specified base cluster. During this scan, alternate key values and primary keys (for a KSDS) or record RBAs (for an ESDS) are extracted and put together to form alternate index records. These records are sorted by ascending alternate keys. The alternate index records are then constructed and written.



ADM2	12687	CASTRO
ADM2	12752	OLIVEIRA
ADM3	12594	PEÇANHA
ADM3	12724	CARMO
MFT01	12747	SCHITTINI
MFT02	12473	VANDERLEI
MFT02	12678	PEREIRA
MFT02	12759	SANTANA
MKT01	12350	CARDOSO
MKT01	12635	MELLO
MKT01	12733	MORAES
MKT02	12613	MARCONDES
MKT02	12763	COUTO JR.

12350	CARDOSO	MKT01
12473	VANDERLEI	MFT02
12594	PEÇANHA	ADM3
12613	MARCONDES	MKT02
12635	MELLO	MKT01
12678	PEREIRA	MFT02
12687	CASTRO	ADM2
12724	CARMO	ADM3
12733	MORAES	MKT01
12747	SCHITTINI	MFT01
12752	OLIVEIRA	ADM2
12759	SANTANA	MFT02
12763	COUTO JR.	MKT02

CARDOSO	12350	MKT01
CARMO	12724	ADM3
CASTRO	12687	ADM2
COUTO JR.	12763	MKT02
MARCONDES	12613	MKT02
MELLO	12635	MKT01
MORAES	12733	MKT01
OLIVEIRA	12752	ADM2
PEÇANHA	12594	ADM3
PEREIRA	12678	MFT02
SANTANA	12759	MFT02
SCHITTINI	12747	MFT01
VANDERLEI	12473	MFT02

Figure 1-8 Alternate Index example

Alternate index paths

Before you access a KSDS or ESDS through an alternate index, a path must be defined in the catalog. A path is the means by which a base cluster is accessed through its alternate indexes. A path is defined and named by using the IDCAMS DEFINE PATH command. At least one path must be defined for each of the alternate indexes through which they access the base cluster. The path name refers to the base cluster and alternate index pair. When a program opens a path for processing, both the base cluster and the alternate index are opened. A base cluster plus all its AIX is called a sphere.

1.4.9 Sphere

A sphere is a base VSAM cluster and its associated clusters. These associated clusters are the alternate indexes (AIX) of the base cluster. An AIX is a KSDS cluster that contains the index entries in the index component that is organized by the alternate keys of its associated base data records. In the AIX data component, there is a set of primary keys that are associated with the referred secondary key. The concept of a sphere provides another way of locating records (by using other keys) in the data component of a cluster.

An AIX can be defined only over a KSDS or ESDS cluster.

1.4.10 Splits

CI split and CA splits occur as a result of data record inclusions (or increasing the length of an already existing record) in KSDS and VRRDS organizations. If a record is to be inserted in key sequence and there is not enough free space in the CI, the CI is split. Approximately half of the records in the CI are transferred to a free CI previously reserved at IDCAMS DEFINE time and provided in the CA. The record to be inserted is placed in the free space available in one of the two CIs.

If there are no free CIs in the CA and a record is to be inserted, a CA split occurs. Half of the CIs of this CA are sent to the first available CA at end of the data component. This movement creates free CIs in the original CA, then the record to be inserted causes a CI split.

Keep in mind that splits are not that bad. The occurrence of a split can worsen the performance, but each subsequent split decreases the probability of having another one. Splits are how VSAM deals with a lack of space, and the process generates free space that helps prevent more splits.

As a result of the split (CI or CA), the physical sequence of records and CIs is no longer the same as the logical sequence by keys. A new index entry is inserted in the sequence set for the new CI, and the existing index entry is updated.

Splits (mainly CA splits) consume resources when they occur. However, the fact that the data is spread in the 3390 volume is no longer a performance issue because there are no longer 3390 seeks. For more information, see 4.3, “VSAM defaults and rule-of-thumbs” on page 133.

The number of CI and CA splits is maintained in the catalog information about each data set.

One performance example of splits is a real life situation where a customer had a KSDS loaded with just one record with key 0. First, a batch job added a few records with a high numeric key. The online transaction program then started adding lots of records randomly with keys less than the ones inputted by the first one. Thousands of splits took place, which negatively affected performance. The solution was to run the batch job after the online program, after which the number of splits went back to normal. VSAM KSDS and VVRDS are not designed for processing of this type.

Figure 1-1 on page 6 shows LISTCAT output where you can see information about splits, such as the SPLIT CI and SPLIT CA.

1.4.11 VSAM buffering

Buffering is one of the key aspects of I/O performance. A VSAM buffer is a virtual storage area where the CI is transferred (to or from) during an I/O operation. In VSAM, there are two types of buffers: Buffers for data CIs and buffers for index CIs. A buffer pool is a set of buffers with the same size. A resource pool is a buffer pool together with some control blocks describing the pool and the data sets with CIs in this resource pool. For more information about VSAM buffering, see 4.4.13, "Buffering options" on page 148.

1.4.12 String multiprocessing

Multiple string processing is a VSAM function that allows the same VSAM data set to be accessed concurrently by tasks (or just one task) in the same address space. Being more technical, there can be multiple independent request parameter lists (RPLs) within an address space for the same data set. There are two VSAM control blocks that are defined within the application code: ACB and RPL.

The access method control block (ACB) defines the logical properties of the file, such as buffering and record insertion algorithm. RPL describes the I/O request itself, such as access type and address of a local buffer. In sequential processing, the RPL is associated with a Place Holder that refers to the next logical record to be retrieved. The Place Holder acts like a sequential memory pointer. There are several ways of implementing multiple string processing through ACB and RPL specifications:

- In the first ACB opened, STRNO or BSTRNO is greater than 1.
- Multiple ACBs are opened for the same data set within the same address space, and are connected to the same control block structure.
- Multiple concurrent RPLs are active against the same ACB by using asynchronous requests.
- Multiple RPLs are active against the same ACB by using synchronous processing with each requiring positioning to be held.

1.4.13 Catalog Search Interface

Catalog Search Interface (CSI) is an application programming interface (API) for searching VSAM catalogs. It can be written in Assembler or even in REXX. CSI is useful in capturing performance and availability information about VSAM data sets and using that data to improve performance and availability.

1.4.14 Extended Address Volume

Large mainframe customers are running out of z/OS addressable disk storage capacity because of the following reasons:

- Rapid data growth on the z/OS platform
- Compatibility issues
- Four-digit device number IBM z/Architecture® limit of 65,280 devices per MVS.

There are a few relief solutions, such as HyperPAV and Space Efficient IBM FlashCopy® (where the FlashCopy is smaller than the original one). HyperPAV allows you to use the total 3390 capacity without a performance penalty because of parallelism implementation. Space Efficient FlashCopy allows for a copy that is smaller than the original one.

A more definitive solution is keeping the compatibility with former channel programs and continue the direction started with the 3390-9 of defining larger volumes by increasing the number of cylinders beyond 65520. This limit is caused by the way a channel is informed by the access method about the following:

- The cylinder number (a 2-byte CC)
- The track number (a 2-byte HH)
- The physical record number (a 1-byte R)

As you can see, the cylinder number (CC) has two bytes, which imposes a limit of 64-K cylinders (54-GB capacity) to the 3390 volume. For more information, see “Channel command word (CCW) and CCHHR” on page 8.

An extended address volume (EAV) is a 3390 volume with more than 65520 cylinders. It uses a new cylinder address format, where initially two bits are taken from HH (not used in a 15 track cylinder) to compose the cylinder number. Then, a 3390 EAV volume has 4 x 64-K cylinders, that is, 256-K cylinders or a full capacity of roughly 223 GB as shown in Figure 1-9.

To use above the 64K cylinders, the channel program must be modified. An important design point is that the 3390 track format and the number of tracks per cylinder remain the same as previous 3390 model devices.

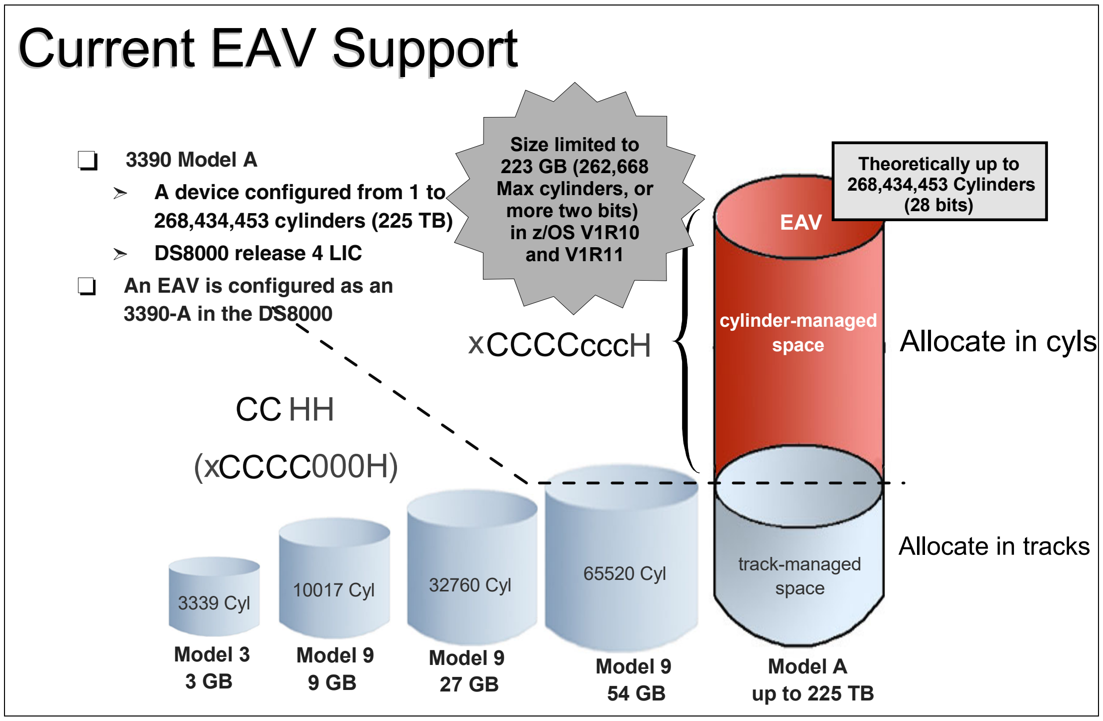


Figure 1-9 EAV architecture

1.5 VSAM data set organizations

VSAM arranges records by index key (KEY), relative record number (RRN), or relative byte address (RBA). VSAM is used for direct or sequential processing of fixed-length and variable length records on DASD. VSAM clusters and components are cataloged in VSAM catalogs for easy retrieval. VSAM supports five cluster organizations:

- Key-sequenced data set (KSDS)
- Entry-sequenced data set (ESDS)

- Relative record data set (RRDS), of which there are two types:
  - Fixed-length relative record data set (RRDS)
  - Variable-length relative record data set (VRRDS)
- Linear data set (LDS)
- Hierarchical file system (HFS) files

1.5.1 Key-sequenced data set (KSDS)

Specify the KSDS organization by using the IDCAMS DEFINE command with the INDEXED parameter.

In a KSDS organization, records are initially loaded in the data component in ascending collating sequence by key. The key (that is the primary) contains a unique value that determines the record's collating position in the data set.

In VSAM KSDS, if a search argument of a logical record is its key field, it has the following restrictions:

- In all logical records, the key length must be the same. The length of the key can be from one byte to 255 bytes.
- In all logical records, the key offset (in relation to the logical record beginning) must be the same.
- The value of the key field cannot be duplicated among the logical records, and therefore must be unique.
- After it is specified, the value of the key cannot be altered. However, the entire record can be deleted.
- If the logical record is to be compressed, only the fields to the right of the key field are compressed.
- In a spanned record, this field must be located totally in the first control interval.

Logical records in a KSDS organization can be fixed or variable length records. When a new record is included in the data component, it is inserted logically in its collating sequence by key. Figure 1-10 shows the structure of a KSDS.

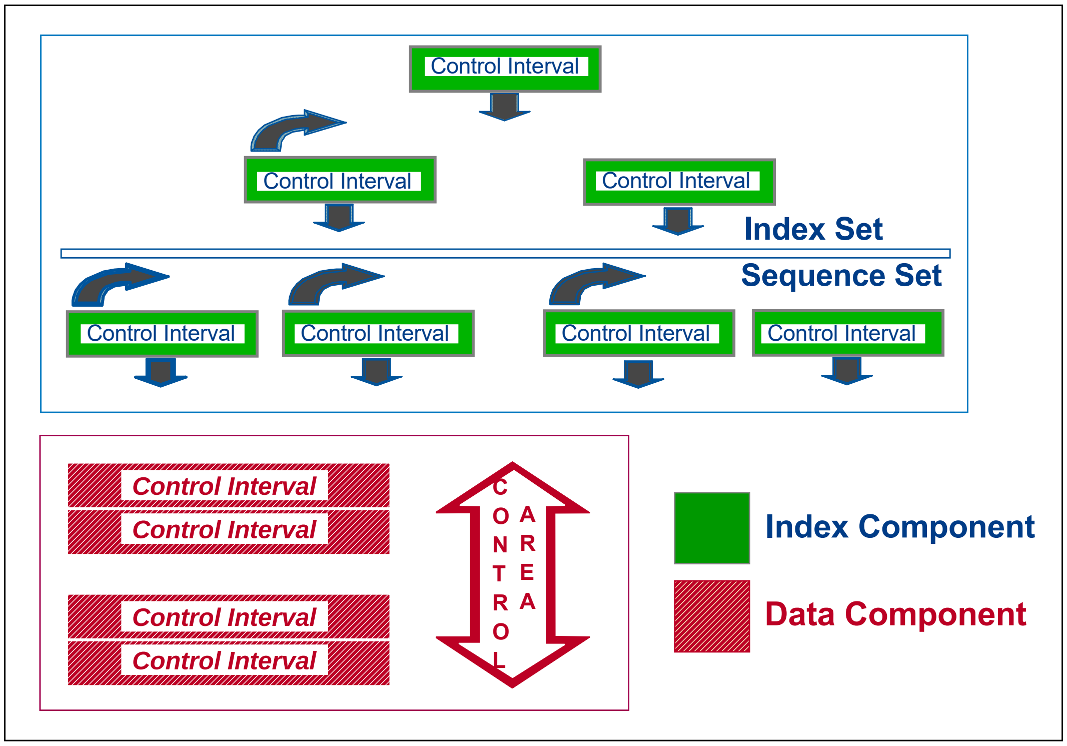


Figure 1-10   Components of a KSDS structure

For VSAM organizations where the key field in a logical record is a search argument (as with the KSDS organization), a “balanced tree” index based on these key values must be created. See Figure 2-1 on page 48. The index has logical records (called index records) that allow a fast random search of a data component logical record with a matching key value. In the index component, there is one index logical record (with the highest key) for each CI of data, and an index CI for each CA of data.

Figure 1-11 shows a numerical example of a data control area of a KSDS data set that is loaded with logical records.

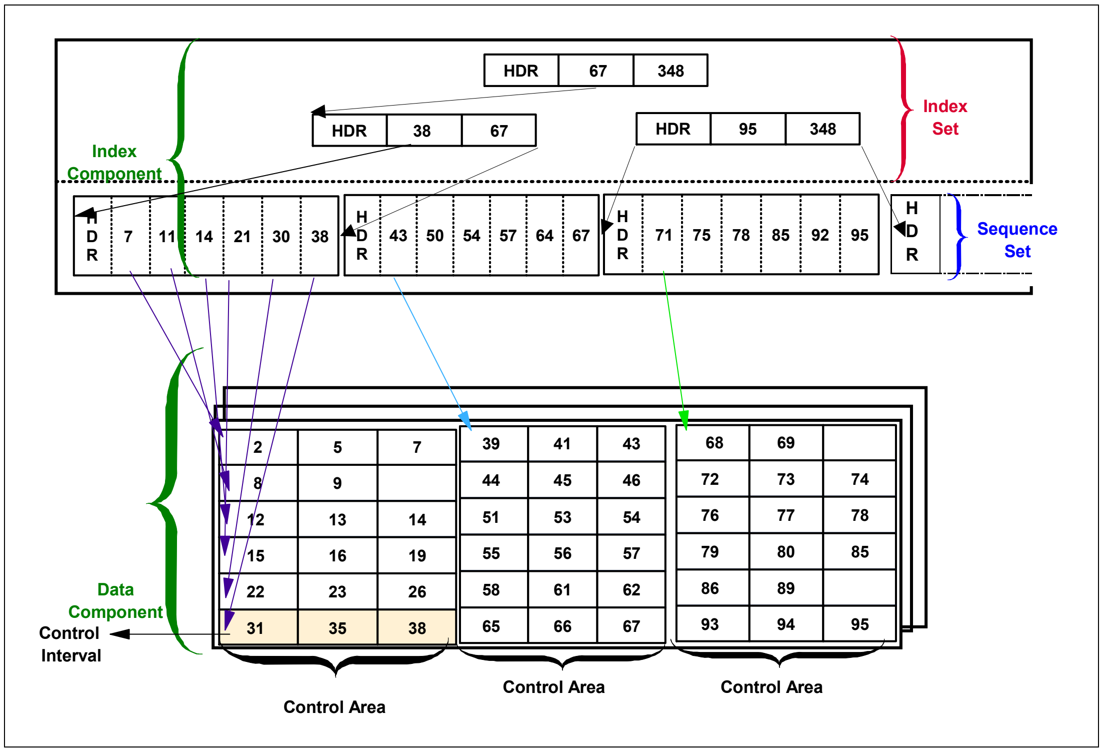


Figure 1-11 KSDS numerical example

KSDS accesses types

There are three access types when you access data records in a KSDS data set:

- Sequential access is used to load a KSDS data set. It can also be used to retrieve, update, add, and delete records in an existing KSDS data set.

  VSAM uses the index component to access data component records in ascending or descending sequence by key. When retrieving (reading) or adding (writing) records, you do not need to specify key values because VSAM automatically obtains the next logical record in key sequence. The sequence set (a low portion of the index) is used to find the next index CI through horizontal pointers.

  Sequential access avoids searching the index more than once. Sequential is faster than direct for accessing multiple data records in ascending key order. The problem is that many times, the application logic does not need to process all of them.

- Direct access is used to retrieve, update, add, and delete records in an existing KSDS data set.

  The application program must supply a key value for each record to be processed (read or written). It can supply the full key or a generic key. The generic key is the high-order portion of a full key. For example, you might want to retrieve all records whose keys begin with XY (where XY is the generic key), regardless of the full key value.

  VSAM searches the index from the highest level index set CI to the sequence set for a record to be accessed. Vertical pointers in the sequence set CI are used to access the data component CA containing the record. The number of index CIs accessed is numerically identical to the number of layers in the index component.

Direct access saves you I/O by not retrieving the entire data set sequentially to process a small percentage of the total number of records.

- Skip-sequential access is used to retrieve, update, add, and delete records in an existing KSDS data set. It is a combination of the two previous types of access. The application randomly provides one search argument and from the located logical record on, all records are processed sequentially.

  VSAM retrieves selected records, but in ascending sequence of key values. Skip sequential access avoids multiple retrieval attempts by using these procedures:

  – Retrieving the entire data set sequentially to process a relatively small percentage of the total number of records in key collating sequence
  – Retrieving the records directly, which causes the index to be searched from top to bottom level for each record

For each request, the sequence set is used to find the next logical CI and to check whether it contains the requested record. If the first skip-sequential search is the first access after you open the data set, a direct search is initiated by VSAM to find the first record. From then on the index sequence set level is used to find the subsequent records. If other operations were run before, either the last position of that operation is used as a starting point to search the sequence set records, or a repositioning is necessary. If spanned records are used for KSDS, the primary key must be within the first control interval.

KSDS numerical example

To help you understand the KSDS organization through the concept of data, index components and splits, a numerical example is introduced in Figure 1-12.

First, the KSDS data set is loaded with logical records. Usually this load is done through the IDCAMS REPRO utility. It reads a sequential data set, where the records are ordered by the ascending key.

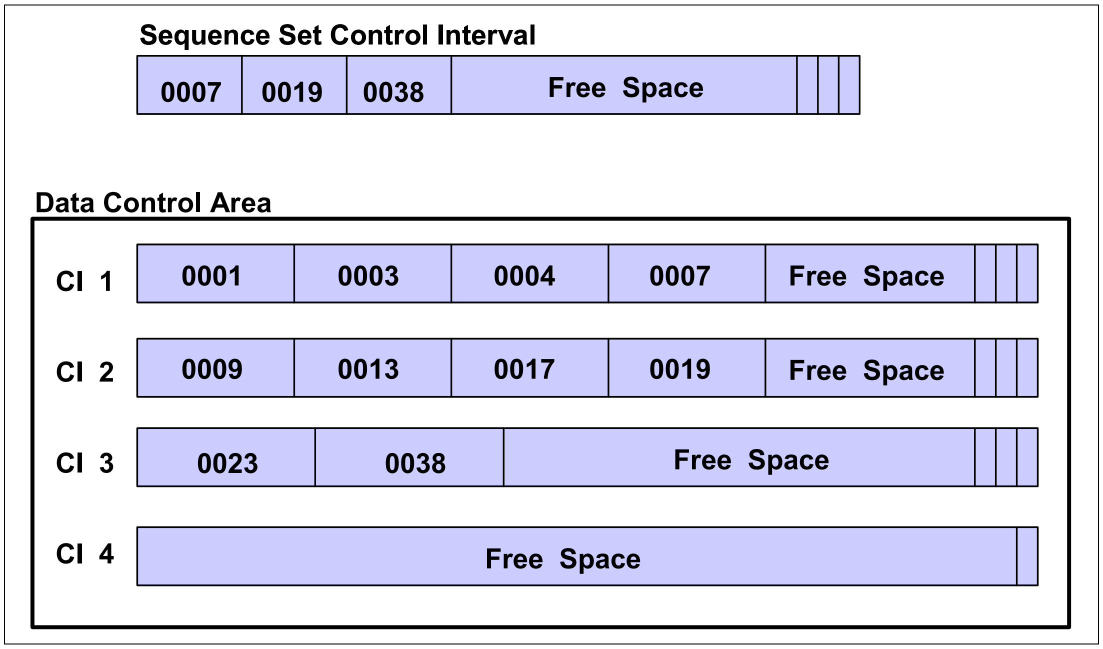


Figure 1-12 Initial loading of a KSDS

As a result of the load, each data component CA has four CIs. Each data CI has a percentage of its area that is reserved as free space, which is reserved for future inclusions. In addition, there is one data CI in each data CA that is completely free that is also reserved for future inclusions. Figure 1-1 on page 6 shows LISTCAT output where you can find information about the free space options such as FREESPACE-%CI and FREESPACE-%CA.

The CIs in the index component describe the location of the data records by the key value:

- The first index record points to the data record in the first data CI with the highest key. In the example, the value of this key is 0007. The index record also has the RBA (not shown in the figure) of this first data CI.
- There is one index record per each data CI, and an index CI per each data CA.

The example in Figure 1-13 has the following characteristics:

- Two data CAs are already filled with data records.
- There are two index CIs in the index sequence set, which is the lower level of the index component.
- A third index CI (in the index set) is already built.
- If an application searches for a data record with key equal to 093, the search sequence includes these steps:
  - Reads the top index CI record (it might already be in the buffer). The value 093 is larger than 038, but is smaller than 169.
  - The index record that shows key 169 has the RBA of an index CI in the sequence set. This index CI is read (or it is already in buffer). The value 093 is larger than 064, but it is smaller than 106.
  - The index record that shows key 106 has the RBA of a data CI in the data component.
  - The data CI is read (or it is already in buffer). The value 093 is larger than 082, and so the data record 093 is found.

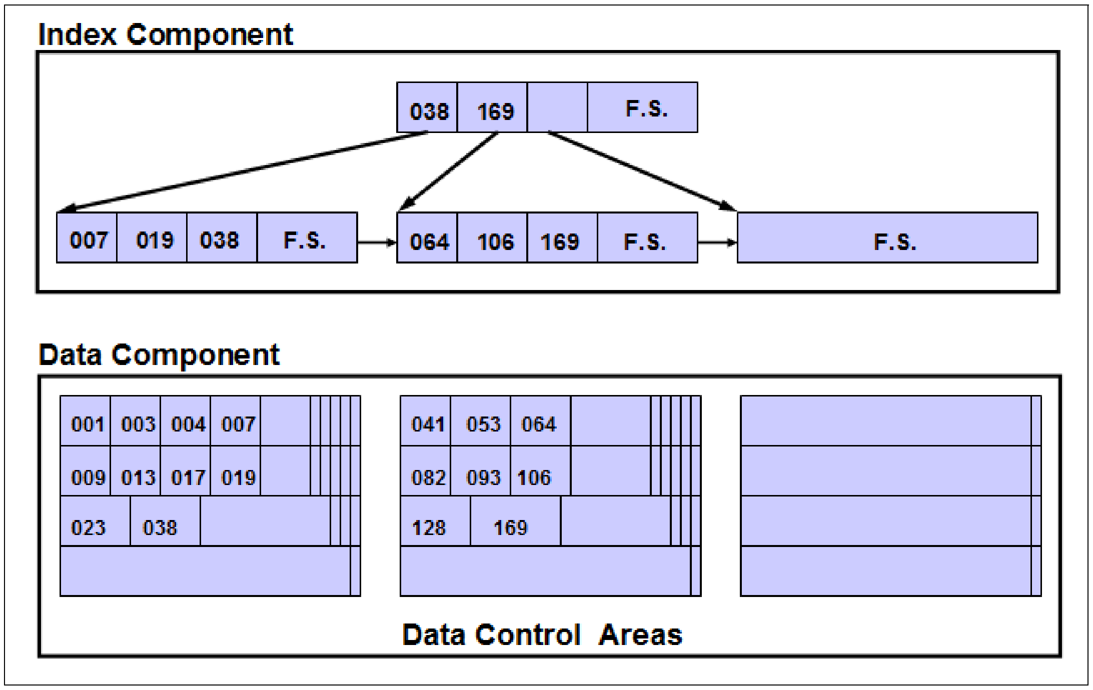


Figure 1-13 After Initial load showing two CAs

Figure 1-14 shows the effect of inserting the data record with Key 0015. Its size is smaller than the free space available in data CI 2 of the data CA 1, so there is no need to change information at the indexes. There is also no need for CI splits.

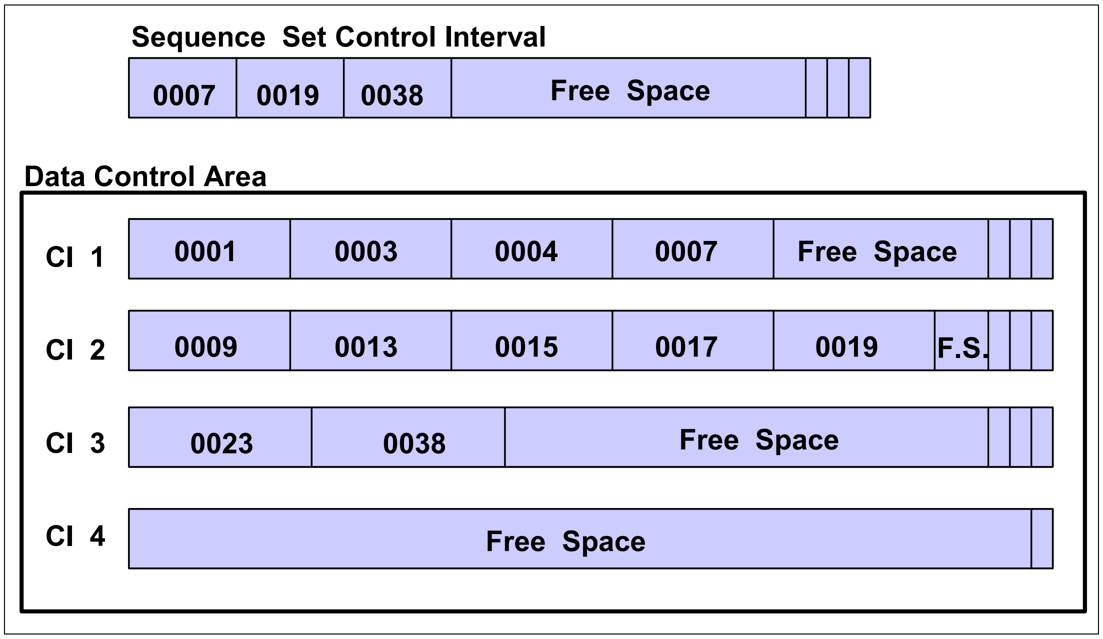


Figure 1-14 Inserting a data record (Key 0015) without a split

In Figure 1-15 on page 26, the application adds the record with Key 0011. It should be inserted in the data CI 2 of the data CA 1. However, it is larger than the available free space. This process causes a CI split. All the data records that should be in the data CI 2 (0011 included) are divided in two groups:

- Data records with Keys 0009, 0011, and 0013 stay in CI 2
- Data records with Keys 0023 and 0038 are moved to the previous empty data CI 4

The sequence set index records are updated to indicate the new distribution of the data records.

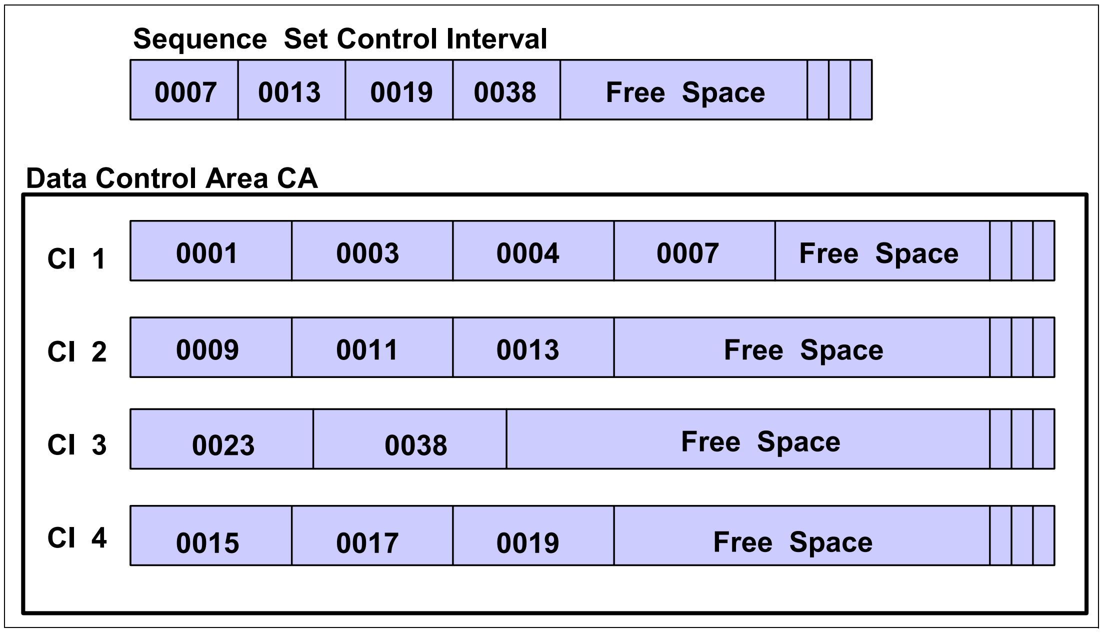


Figure 1-15 Contents of CA 1 after insertion of record with Key 0011 causing CI split

As shown in Figure 1-16, the data records with Keys 0025 and 0029 were inserted in CI 3 without any splits.

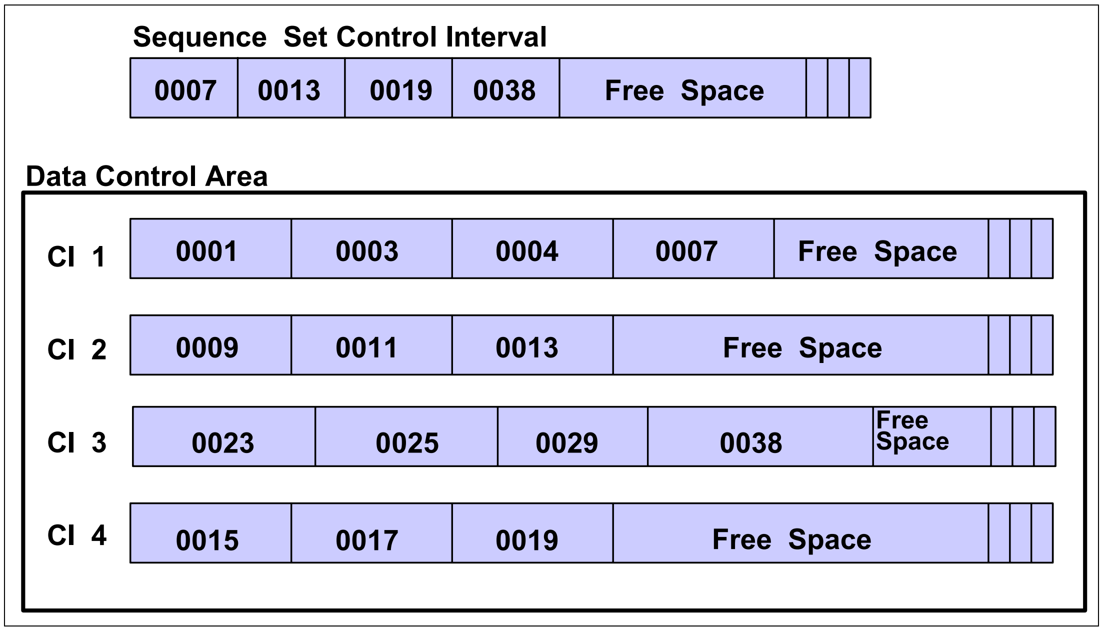


Figure 1-16 CI 3 after insertion of records with Keys 0025 and 0029

In Figure 1-17, the record with Key 0033 is inserted. It is larger that the free space available in the CI 3, and there is not a CI completely free in the data CA 1. This configuration causes a CA split. The contents of CI 3 and CI 4 are copied to the first totally available CA after the HURBA. Because of the CA split, the CA 1 has now two data CIs that are totally free. The indexes are updated to indicate the new data distribution.

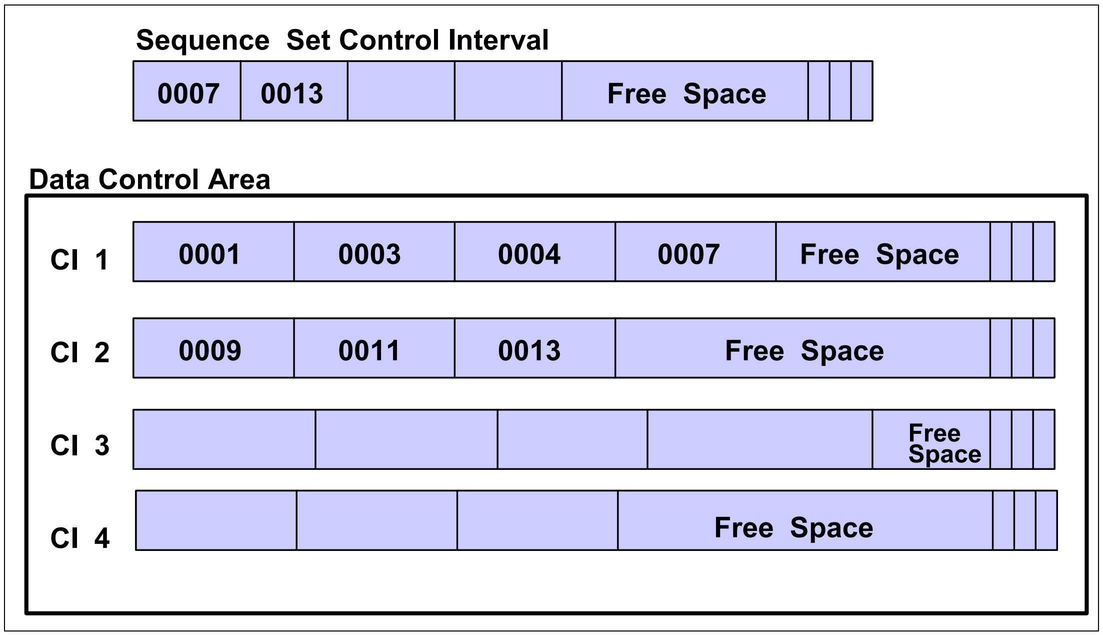


Figure 1-17 The original CA 1 after the CA split

In Figure 1-18, you can see the second CA involved in the CA split. Initially CI 3 and CI 4 are free. After the CA split, CI 3 is used for a CI split to make room for the record 0033.

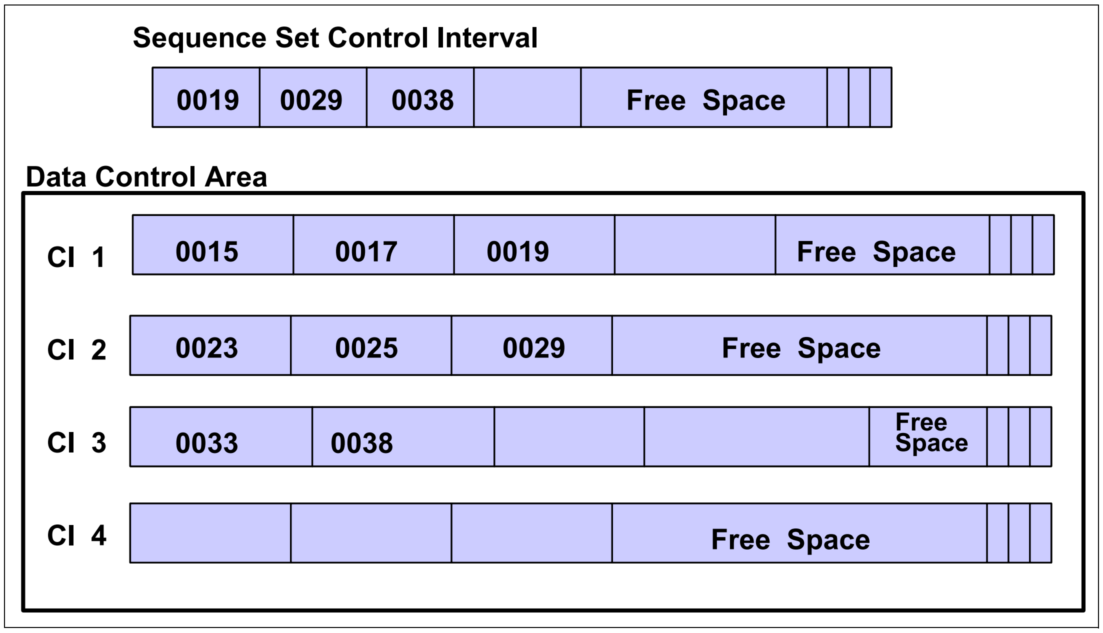


Figure 1-18 Contents of second CA after split

1.5.2 Entry-sequenced data set

You specify entry-sequenced data set (ESDS) organization by using the IDCAMS DEFINE command and specifying the NONINDEXED parameter.

An ESDS is comparable to a sequential non-VSAM data set in the sense that records are sequenced by the order of their entry in the data set, rather than by key field in the logical record, which could be fixed or variable length records.

All new records are placed at the end of the data set. There is no split concept. Existing records can never be physically deleted, or as far as VSAM is concerned, the record is not scratched. It is the responsibility of the application program to identify that record as invalid through an application identified flag.

The content of records can be updated, but their length cannot be changed. To change the size of a record, use one of these techniques:

- Store it at the end of the data set as a new record
- Override an existing record of the same length that you have flagged as inactive

ESDS types of access

A record can be accessed sequentially or directly by its RBA. The specific RBA is converted to CCHHR terms, to prepare the Locate Record CCW for the I/O operation:

- Sequential access

  VSAM automatically retrieves records in stored sequence. Sequential access can be started from the beginning or somewhere in the middle of a data set. If processing is to begin in the middle of a data set, RBA positioning is necessary before sequential access can be run.

- Direct (or random) access

  When a record is loaded or added, VSAM returns its RBA to the application. To retrieve records directly, you must supply the RBA for the record as a search argument.

Skip sequential access is not allowed for an ESDS. For more information about skip sequential access, see 1.2.2, “VSAM access types” on page 4.

Figure 1-19 shows the format of an ESDS.

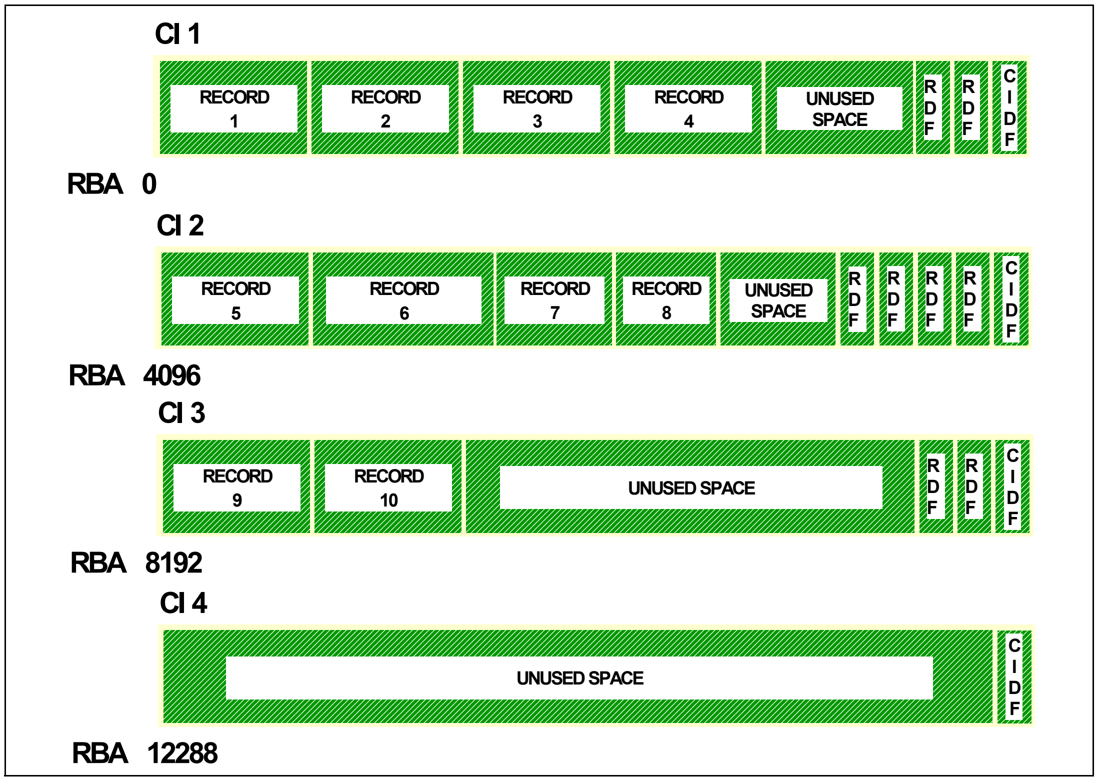


Figure 1-19 Entry sequenced data set

Empty spaces in the CI are called *unused space* because they can never be used. These spaces are a result of CI internal fragmentation.

AIX can also be applied to ESDS organization. Instead of randomly accessing an ESDS by RBA, you can ask for an AIX, where a key field in the logical record is used as a random search argument. In this case, the AIX data component, which is a KSDS data set, automatically relates the key with the RBA in the base data set. The BLDINDEX command causes a sequential scan of the specified base data set. During this scan, key values and record RBAs are extracted and put together to form alternate index records. Before you can access an ESDS through an alternate index, a path must be defined.

An example of ESDS with an AIX is a data set used as a time stamp log, describing all the bank transactions in a checking account application. In this case, most of the access is sequential with a few random accesses by the key (through the AIX).

1.5.3 Relative record data set

Specify the relative record data set (RRDS) organization by using the IDCAMS DEFINE command with the NUMBERED option.

A relative record data set consists of a number of pre-formatted, fixed-length logical records slots. Each slot has a unique RRN, and the slots are sequenced by ascending relative record number. These slots are grouped in CIs.

Each fixed length logical record occupies a slot, and is stored and retrieved by the relative record number of that slot. The position of a logical record is fixed, and its relative record number (RRN) cannot change.

Because the slot can either contain data or be empty, a data record can be inserted or deleted without affecting the position of other data records in the RRDS organization. The RDF shows whether the slot is occupied or empty. Free space is not provided because the entire data set is divided into fixed-length slots. There might be empty spaces in the CI, which are called unused space because they can never be used. This space is a result of CI internal fragmentation.

Figure 1-20 shows the format of an RRDS.

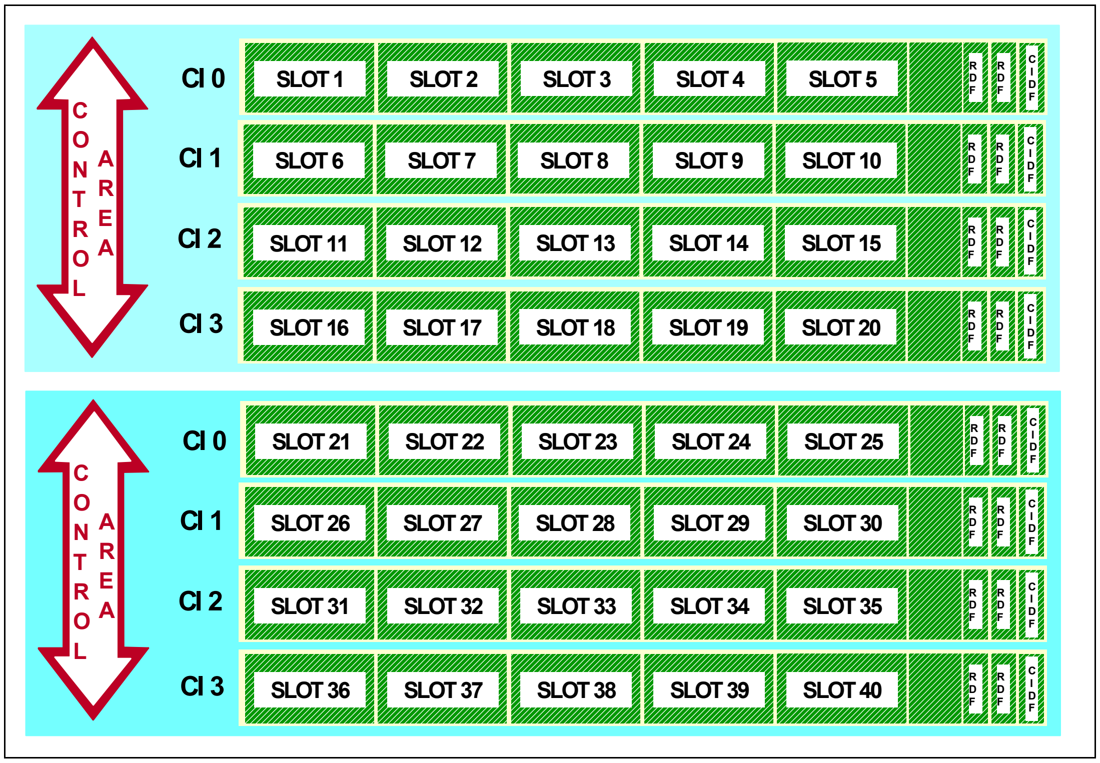


Figure 1-20 General format of a relative record data set

Typical RRDS processing

The application program provides the RRN of the target record. VSAM is then able to find its location (by using CCHHR format conversion) quickly by using a formula that considers the geometry of the DASD device. The relative record number is always used as a search argument for a random access.

An RRDS can be processed sequentially, directly, or skip-sequentially:

- ► RRDS sequential access is treated the same way as ESDS sequential access. However, empty slots are automatically skipped by VSAM.
- ► An RRDS can be processed directly by supplying the relative record number as a key. VSAM calculates the RBA and accesses the appropriate record or slot through CCHHR. RRDS direct address processing by supplying the RBA is not supported.
- ► Skip-sequential access is treated like an RRDS direct processing request, but the position is maintained. After that, records are processed in ascending relative record number sequence.

1.5.4 Variable relative record data set

Specify the VRRDS organization by using the IDCAMS DEFINE command with the NUMBERED option and variable length record.

A VRRDS is like an RRDS, except that it contains variable-length records. Each record has a unique relative record number, and is placed in ascending relative record number order. Each record is stored and retrieved by using its relative record number. VRRDS has no slots.

The relative record number of a record cannot change. When that record is erased, the relative record number can be reused for a new record.

Because the logical records have a variable size, you cannot use a formula to derive its RBA (and then the CCHHR) from its RRN. In a sense, a VRRDS organization is a KSDS processed by RRN instead of a key. The index and data components form the cluster. The search argument is the RRN pointing to the RBA of the corresponding logical record in the data component.

You can specify free space for inserting records and increasing the length of a record. The split concept is implemented. This organization is not used frequently.

1.5.5 Linear data set

Specify the linear data set (LDS) organization with the IDCAMS DEFINE command by specifying the LINEAR parameter.

A linear data set contains data that can be accessed as byte-addressable strings in virtual storage. It is a VSAM data set with a control interval size multiple of 4096 bytes. An LDS has no embedded control information in its CI, that is, no RDFs and CIDFs. All LDS bytes are data bytes. Logical records must be blocked and deblocked by the program. Logical records are not apparent from VSAM's point of view.

In a sense, an LDS is a non-VSAM data set with some of the VSAM facilities, such as the use of IDCAMS and VSAM specific information in the catalog. The most common LDS user is IBM DB2®. LDS is the VSAM data set organization that is used by Data-in-Virtual (DIV) facility. For more information, see "Data-in-Virtual (DIV)" on page 32. Like the ESDS and RRDS, LDS contains a data component only.

Figure 1-21 shows the format of an LDS.

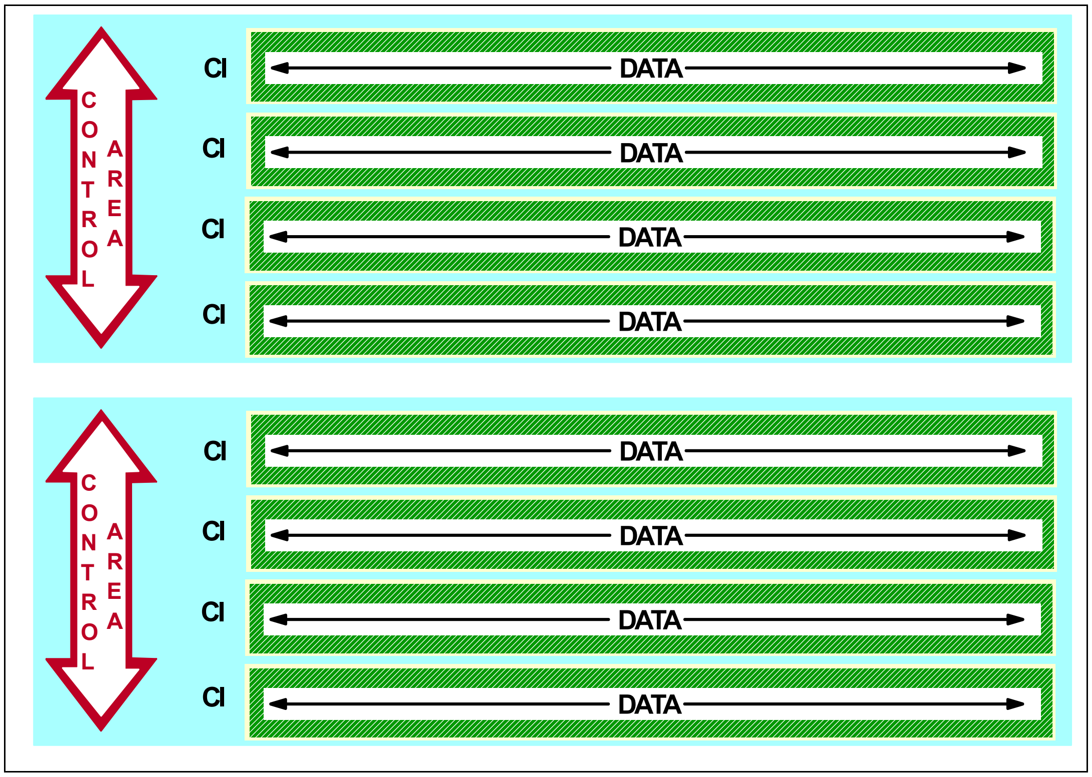


Figure 1-21 Linear data set (LDS)

Data-in-Virtual (DIV)

DIV is an optional and unique buffering technique that is used for LDS data sets only. Application programs can use DIV to map an LDS data set or a portion of it into an address space, a data space, or a hiperspace. An LDS data set is sometimes called a DIV object.

Data is read into central storage through the paging mechanism only when that DIV 4-KB data block is referenced. During Real Storage Management (a z/OS component) page steal processing, only changed pages are written to auxiliary storage. Unchanged pages are discarded because they can be retrieved again from the permanent linear data set.

DIV is designed to improve the performance of applications that process large data sets non-sequentially in an unpredictable pattern. It reduces the number of I/O operations that are traditionally associated with data retrieval. Likely candidates are large arrays and table data sets.

Mapping a linear data set

To establish a map from a linear data set to a window (a program provided virtual area in multiples of 4K on a 4K boundary), the program issues these commands:

- DIV IDENTIFY to introduce (allocate) a linear data set to DIV services.
- DIV ACCESS to cause a VSAM open for the data set and indicate access mode (read or update).
- DIV MAP to enable the viewing of the data object by establishing an association between a program provided area and the data object. The area can be in an address space, data space, or hiperspace.

No actual I/O is done until the program references the data in the window. The reference results in a page fault. This fault causes DIV services in Real Storage Manager to read the data from the linear data set into the window. It can be fixed by using these techniques:

- DIV SAVE can be used to force the write out changes to the data object in the Linear data set.
- DIV RESET can be used to discard changes that have been made in the window since the last SAVE operation.

1.5.6 Comparing VSAM data set organizations

Table 1-2 provides a summary of the characteristics of VSAM data set types.

Table 1-2 Comparison of ESDS, KSDS, RRDS, VRRDS, and linear data sets

ESDS	KSDS	Fixed-length RRDS	Variable-length RRDS	Linear data sets
Records are in the same order as they are entered	Records are in collating sequence by key field after load	Records are in relative record number order	Records are in relative record number order	No processing at record level
Records can be fixed or variable length	Records can be fixed or variable length	Records are fixed length	Records are variable length	No processing at record level
Direct access by RBA	Direct access by key or by RBA	Direct access by relative record number	Direct access by relative record number	Access with Data-In-Virtual (DIV) optionally
Consist of data component only	Consist of data and index components	Consist of data component only	Consist of data and index components	Consist of data component only
Alternate index allowed	Alternate indexes allowed	No alternate index allowed	No alternate index allowed	No alternate index allowed
A record’s RBA cannot change	A record’s RBA can change	A record’s relative record number cannot change	A record’s relative record number cannot change	No processing at record level
Space at the end of the data set is used for adding records	Free space is used for inserting and lengthening records	Empty slots in the data set are used for adding records	Free space is used for inserting and lengthening records	No processing at record level
A record cannot be deleted, but you can reuse its space for a record of the same length	Space that is given up by a deleted or shortened record becomes free space	A slot that is given up by a deleted record can be reused	Space that is given up by a deleted or shortened record becomes free space	No processing at record level
Spanned records allowed	Spanned records allowed	No spanned records	No spanned records	No spanned records
Extended format allowed	Extended format or compression allowed	Extended format allowed	Extended format allowed	Extended format allowed

1.5.7 Selecting a VSAM data set type

During the application development process, the application programmer must make decisions about the data model. These decisions include data organization, function, type of access, performance, and recovery tools you need. VSAM has several organizations and different ways for accessing your data.

Before you select a particular VSAM organization, answer the following questions:

- What is the main purpose of your data set? Does it look more like a log, a database, or an inventory?
- Do you need to access data by a key field in sequential or direct mode?
- Do you need to access the records in sequence, skip sequential, randomly, or all of them?
- Are all the logical records the same length?
- Might the logical record length change?
- Do you need to have insertions and deletions?
- Is the data set going to be extended?
- How often do you need to delete records?
- Do you use spanned records?
- Do you want to keep the data in order by the contents of the key field?
- Do you want to access the data by an alternate index?
- Do you want to use DIV?
- Do you want to use data compression?
- Do you need the utility functions provided by IDCAMS?

Use the answers to these questions to choose the best organization for your data set. Remember that VSAM data sets cannot be processed by using non-VSAM applications. Similarly, non-VSAM data sets cannot be processed by using the VSAM access method.

Guidelines

Following are guidelines for choosing a data set organization. The organizations are presented from lowest to highest in processor and I/O consumption.

- Use QSAM or BSAM in the following circumstances:
  - You use no direct processing.
  - There are no insertions or deletions, and no change in the logical record length.
  - Record additions are only at the end of the data set.
  - You are not concerned about IDCAMS functions.
  - You want to have data compression.
  - Performance and easy recovery are main issues.
- Use LDS in the following circumstances:
  - You want to use DIV.
  - Your application manages logical records.
  - Performance is an issue.
- Use RRDS in the following circumstances:
  - The record processing is sequential, skip sequential, or direct processing.
  - Easy programming for direct processing is not a requirement.

  - The argument for accessing data in direct mode is a relative record number, not the contents of a data field (key). RRDS is suitable for the type of logical records that are identified by a continuous and dense pattern of numbers (such as 1, 2, 3, 4).
  - All records are fixed length.
  - There are few record insertions and deletions, and all the space for insertions must be pre-allocated in advance.
  - Performance is an issue. RRDS performance is better than KSDS, but worse than QSAM or BSAM.
- Use ESDS in the following circumstances:
  - You are adding logical records only at the end of the data set and reading them sequentially (in the application control).
  - The logical record is variable length.
  - You rarely need direct record processing by key (using AIX).
  - You are using a batch processing application.
- Use VRRDS in the following circumstances:
  - You have the same requirements as for a KSDS, but use record number instead of a key field as the argument.
- Use KSDS in the following circumstances:
  - The data access is sequential, skip sequential, or direct access by a key field.
  - You would prefer easy programming for direct data processing.
  - There are many record insertions, deletions, and logical record length variations.
  - You can optionally access records by an alternate index.
  - Complex recovery (because of index and data components) is not a problem.
  - You want to use data compression.

Many times, you must use a specific VSAM organization based on the software product you are running. For example, DB2 uses LDS data sets. In this case, you cannot select the VSAM organization.

1.6 Data striping

Usually, in a multi-extent, multi-volume VSAM data set accessed sequentially, there is no opportunity for any type of parallelism for I/O operations among these volumes. Even parallel access volume (PAV) is not able to run these I/Os in parallel. Therefore, when an I/O operation is run for an extent in a volume, no other I/O activity from the same task or data set is scheduled to the other volumes. If I/O is the major bottleneck, and there are available resources in the channel subsystem and controllers, it is a waste of those resources.

The data striping VSAM function addresses this sequential access performance problem by adding two modifications to the traditional data organization:

- The records are not placed in key ranges along the volumes. Instead, they are organized in stripes.
- Parallel I/O operations are scheduled to sequential stripes (CIs) on different volumes.

By striping CIs, VSAM can spread simultaneous I/Os across multiple devices. This format allows a single application request for records in multiple tracks and CIs to be satisfied by concurrent I/O requests to multiple volumes.

The result is improved performance by allowing more data transfer into the application than any single I/O path.

This function is compatible at application level, and therefore requires no modifications.

An example is shown in Figure 1-22. CI 1 is written at VOL A, CI 2 at VOL B, CI 3 at VOL C, CI 4 at VOL A again. The first stripe is made of VOL A and VOL F. The second stripe is made of VOL B and VOL D and VOL G, and the third stripe of Vol C, Vol E, and Vol H.

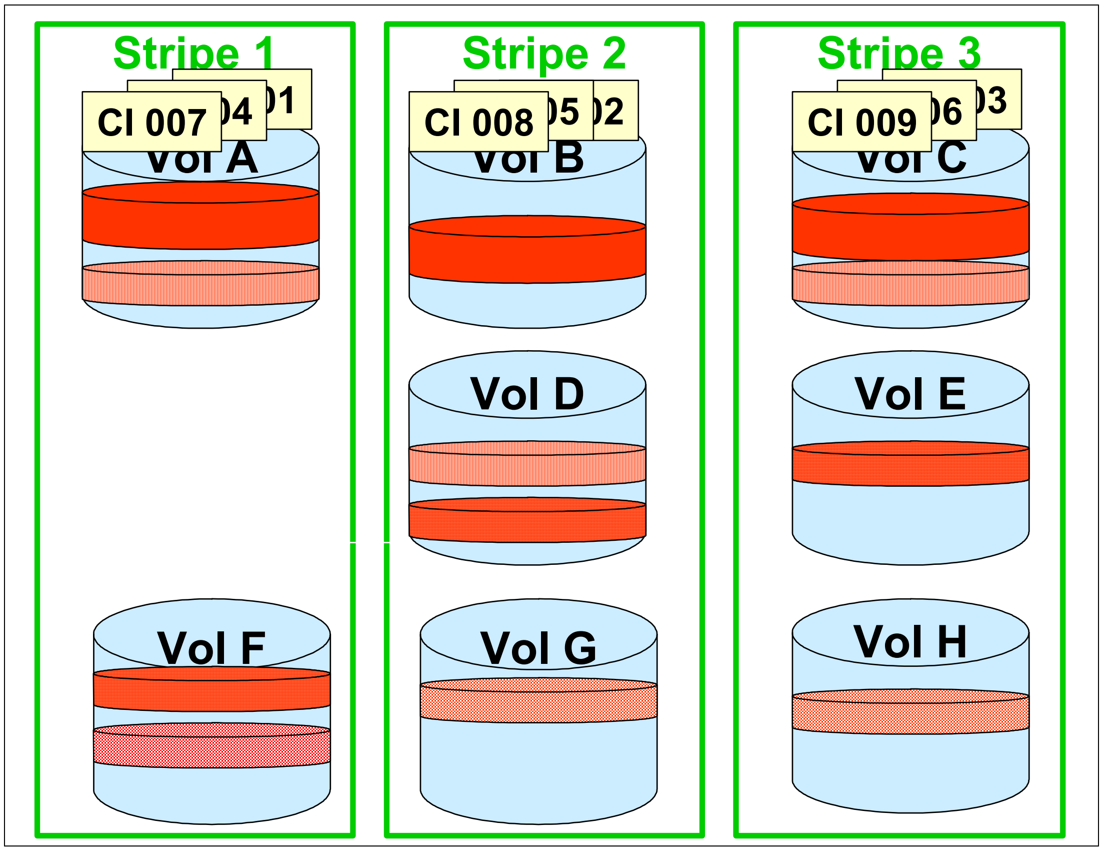


Figure 1-22 Data Striping example

1.7 Processing a VSAM data set

Before you process a VSAM data set, review the source of VSAM processing options. For more information, see 2.6, “Major sources of VSAM processing options” on page 71.

Processing a VSAM data set involves the following actions:

- Define the VSAM data set through the IDCAMS utility program.
- Allocate the data set to establish the logical link between a dispatchable unit (TCB or SRB) program and the data set.
- Open the data set, identifying it with a DDNAME.

- Access the data through GET (or Read) and PUT (or Write) by starting the VSAM access method.
- Close the data set.
- Deallocate the data set.

1.7.1 Defining VSAM data sets

To define a data set is a VSAM expression. It means to create and catalog the data set, but it does not imply allocating the data set. You define a VSAM data set by using IDCAMS commands such as DEFINE and ALLOCATE. The data sets are defined but not allocated to a task, and are empty.

Using IDCAMS

IDCAMS is a program that is provided in DFSMSdfp. It is used to establish and maintain catalogs and VSAM data sets.

When you use IDCAMS to define a VSAM data set, specify these characteristics:

- Component name or cluster name.
- Data set type. The default is INDEXED (KSDS).
- Space allocation, both primary and secondary allocations, and the volumes on which the data set's components are to have space.
- Component attributes, such as record size and CI size. For a KSDS, specify key information and free space.
- Catalog information.

Figure 1-23 shows the syntax of the DEFINE CLUSTER command and the required parameters.

```
DEFINE CLUSTER -
      (NAME(entryname))-
      CYLINDERS(primary secondary)|
      KILOBYTES(primary secondary)|
      MEGABYTES(primary secondary)|
      RECORDS (primary secondary)|
      TRACKS(primary secondary) -
      VOLUMES(volser[volser...]) -
  DATA (parameters) -
  INDEX (parameters) -
  CATALOG(subparameters)
```

Figure 1-23 DEFINE command required parameters

1.7.2 Allocating a VSAM data set

Allocating a data set does not imply creating a data set. You can allocate an already existing data set. Allocating a data set to a task means to establish a logical connection between the task (through a program that runs underneath) with the data set. For a DISP=NEW data set, allocation includes the act of creating the data set (generating an entry in the VTOC). There are several ways to allocate a data set:

- The ALLOCATE TSO command allocates (defining is included) a data set VSAM from a TSO terminal. The TSO commands are described in *z/OS TSO/E Command Reference,* SA22-7782.
- JCL DD statements define and allocate a VSAM data set. In this case, the z/OS component in charge of starting the allocation routine is the Initiator. For more information about using JCL, see *z/OS MVS JCL Reference,* GC28-1757, and *z/OS MVS JCL User's Guide,* GC28-1758.
- Dynamic allocation allocates and defines a VSAM data set through the DYNALLOC macro (which expands using the SVC 99 parameter list) issued within an Assembler program. For more information, see *z/OS MVS Authorized Assembler Services Guide,* GC28-1763.

The functions that are used at allocation time depend on whether the data set exists:

- If it exists, its device is located through a catalog search, a serialization ENQ is issued (depending on the DISP parameter), and a VTOC search is run.
- If it does not exist, a device must be selected by SMS and SRM. The data set is created, an entry made in the VTOC, and cataloged (all VSAM components must be cataloged).

After allocation the data set must be opened to be accessed.

1.7.3 Opening a VSAM data set

The Open routine (SVC19) is a component of the DFSMS. Opening a VSAM data set involves the following actions:

- Enter the remaining options in ACB from JFCB (DD card) and from the catalog
- Create (in one specific ACB field) the address of the VSAM routine in charge of processing the options that are described in ACB. This address is a pointer to be used later for reading and writing to the data set. The address has this order of precedence:

  1.The DD statement AMP parameters

  2.The ACB, EXLST, or GENCB parameters

  3.The catalog entry for the data set

  For example, if both an ACB or GENCB macro and the DD statement have values for buffer space, the values in the DD statement override those in the macro.
- Construct the internal control blocks that VSAM needs to process your requests for access to the data set.
- Construct the Data Extent Block (DEB), indicating to MVS that the data set is opened.
- Check for consistency of updates to the prime index and data components if you are opening a KSDS, an AIX, or a path. VSAM issues a warning message to indicate a time stamp discrepancy.
- Issue a RACROUTE for IBM RACF® authorization for the operation.

Closing a data set is fundamentally involves the same steps in reverse.

1.7.4 Accessing VSAM data set

VSAM data sets can be accessed by different types of programs, such as batch program, IBM CICS® application programs, IDCAMS, and DITTO. TSO REXX and CLISTs cannot be used to access VSAM entities.

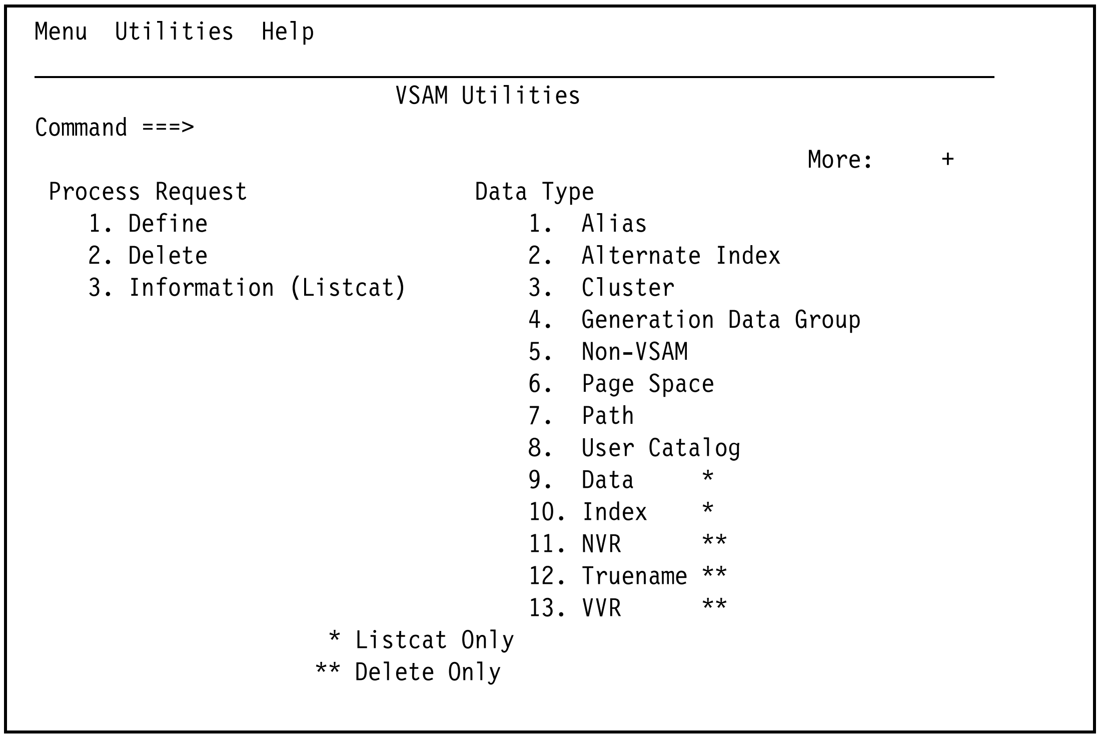
Batch and CICS application programs

VSAM data sets can be written by using languages that support VSAM, such as COBOL, PL/I, Java, and Assembler. To obtain VSAM services, these application programs use VSAM macros, mainly GET/PUT. For more information, see *z/OS DFSMS Macro Instructions for Data Sets*, SC26-7408.

IDCAMS

IDCAMS can be started in one of the following ways:

- As a job or job step by specifying PGM=IDCAMS on the EXEC statement.
- From an application program.
- From a TSO terminal that is running commands. Under ISPF, there is option 3.2.V, which is the VSAM Utilities menu that is shown in Figure 1-24.



```
Menu  Utilities  Help
------------------------------------------------------------------------
                           VSAM Utilities
Command ===>
                                                          More:     +
 Process Request                Data Type
    1. Define                       1.  Alias
    2. Delete                       2.  Alternate Index
    3. Information (Listcat)        3.  Cluster
                                    4.  Generation Data Group
                                    5.  Non-VSAM
                                    6.  Page Space
                                    7.  Path
                                    8.  User Catalog
                                    9.  Data     *
                                    10. Index    *
                                    11. NVR      **
                                    12. Truename **
                                    13. VVR      **
                      * Listcat Only
                     ** Delete Only
```

Figure 1-24 ISPF menu of VSAM Utilities

You can also call the IDCAMS program from within another program, and then pass the Access Method Services command and parameters to the IDCAMS program.

Access Method Services commands

There are two types of Access Method Services commands:

- Functional commands are used to request the actual work, such as defining a data set or listing a catalog (Figure 1-25).
- Modal commands allow the conditional execution of functional commands. TSO users can use functional commands only. For more information, see *DFSMS Access Method Services for Catalogs*, SC26-7394.

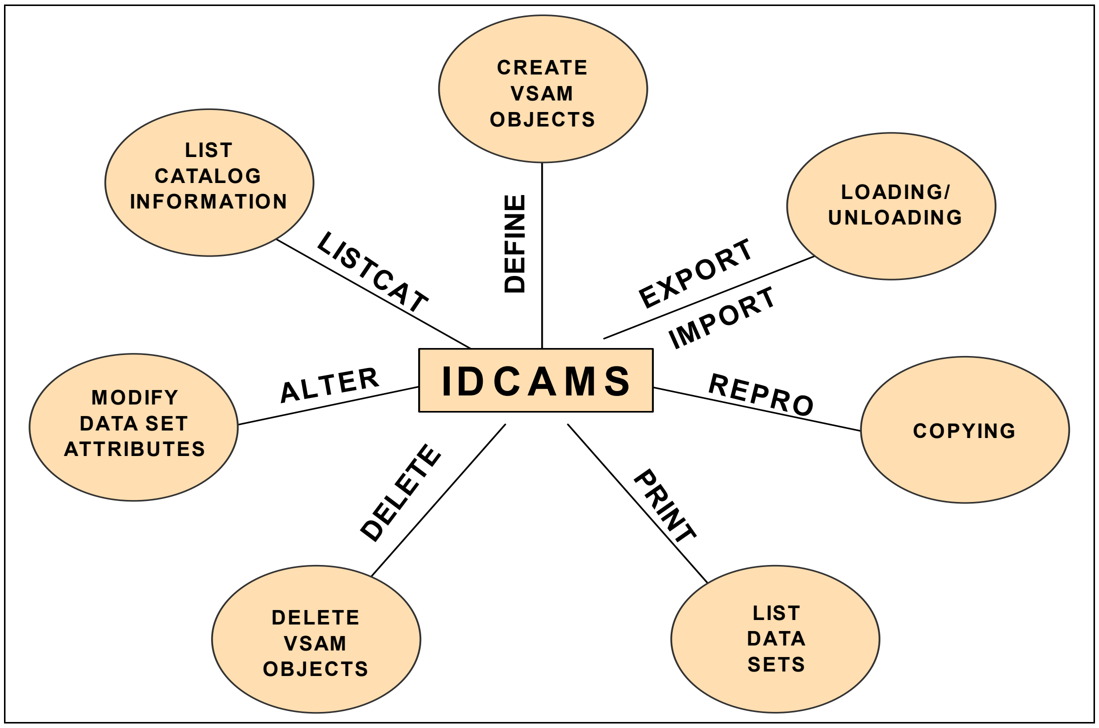


Figure 1-25 Access Method Services commands

1.7.5 Unallocation

Unallocation undoes what allocation did. There are three ways to deallocate a VSAM data set:

- Closing the data set can run this function if FREE=CLOSE is specified for the allocation.
- Your program can run dynamic unallocation.
- During the step termination process, the initiator/terminator program automatically unallocates any remaining allocated data sets.

1.8 VSAM exploiters

The following major applications use VSAM functions:

- DB2
- zSeries file system
- Hierarchical file system
- CICS
- DFSMShsm

- DFSMSrmm
- Java Record I/O (JRIO)

1.8.1 DB2

DB2 uses linear data sets (LDS) for its table spaces without implementing Data-in-Virtual. All the control (including buffer pool) is done by DB2. For example, DB2 implements data striping in LDS data sets.

1.8.2 zSeries file system

IBM zSeries® file system (zFS) is a z/OS UNIX file system that can be used in addition to the hierarchical file system (HFS). zFS provides significant performance gains in accessing files approaching 8K and above in size that are frequently accessed and updated. The performance of smaller files is equivalent to that of HFS. zFS helps prevent loss of updates by writing data blocks asynchronously and not waiting for a sync interval. zFS is a logging file system. It logs metadata updates. If a system failure occurs, zFS replays the log when it comes back up to ensure that the file system is consistent.

Application programming interfaces

zFS file systems contain files and directories that can be accessed with the z/OS hierarchical file system application programming interfaces on the z/OS operating system:

- An application interface is composed of C interfaces. Some of these interfaces are managed within the C Run-Time Library (RTL), while others access kernel interfaces to run authorized system functions on behalf of the unauthorized caller
- An interactive z/OS shell interface by shell users

zFS aggregate

A zFS aggregate is a data set that contains zFS file systems. The aggregate is a VSAM linear data set (VSAM LDS), and is a container that can contain one or more zFS file systems. An aggregate can have only one VSAM LDS, but it can contain an unlimited number of file systems. The name of the aggregate is the same as the VSAM LDS name. Sufficient space must be available on the volume or volumes. Multiple volumes can be specified on the DEFINE of the VSAM LDS. DFSMS decides when to allocate on these volumes during any extension of a primary allocation. VSAM LDS organization greater than 4 GB can be specified by using the extended format and extended addressability capability in the data class of the data set.

1.8.3 Hierarchical file system

Hierarchical file system (HFS) is a UNIX Services data organization with no determined logical records. It is a byte string that is made of embedded directories and data. It can be accessed by Posix code (a UNIX standard for operating system interfaces, APIs) running under z/OS or by z/OS applications. In the second case, an HFS looks like a VSAM ESDS organization and is accessed in the same way.

Accessing HFS files through VSAM

You can access an HFS file through VSAM in one of the following ways:

- JCL DD statement that specifies PATH=pathname
- SVC99
- TSO ALLOCATE command

HFS files are simulated as an ESDS. However, because HFS files are not stored as ESDSs, VSAM cannot simulate all the characteristics of a sequential data set. As a result, there are certain macros and services that have incompatibilities or restrictions when dealing with HFS files. For more information, see *z/OS DFSMS Using Data Sets*, SC26-7410.

1.8.4 CICS

CICS allows the use of VSAM data sets by CICS transactions. CICS uses the MVS system logger data set for logging the updates. Usually when you access a VSAM data set in non-RLS mode, a specific CICS address space is used to centralize all the clutter access. When in VSAM RLS mode, the CICS AOR address spaces connect directly to the SMSVSAM address space. For more information, see Chapter 5, "VSAM Record Level Sharing" on page 209.

1.8.5 DFSMShsm

DFSMShsm has three control data sets:

- Migration control data set (MCDS)
- Backup control data set (BCDS)
- Offline control data set (OCDS)

All of these sets are VSAM KSDSs. In a multi-MVS image sysplex environment, these data sets can be shared (RLS) between distinct DFSMShsm instances. This environment is called HSMplex.

1.8.6 DFSMSrmm

DFSMSrmm also uses VSAM data sets for their control data sets. They are defined with **SHAREOPTIONS(3 3)**, and use the VSI control block to ensure that they have the most recent HURBA on each system.

1.8.7 Java Record I/O (JRIO)

JRIO is a set of APIs that allowing Java code to access I/O record oriented data organizations. It is an addition to java.io APIs, which support only HFS access.

JRIO allows Java applications to access traditional z/OS file systems in addition to the HFS. JRIO makes it easier for Java applications to access records within files and to access file systems through native methods. This configuration is useful when java.io APIs do not support those file systems.

JRIO is a class library, similar to java.io. While java.io provides byte-oriented or field-oriented access to files, JRIO provides record-oriented access, which is much more natural. The two systems have these major differences:

- Read/write sequential and random access for byte string data sets is provided by java.io.
- JRIO allows:
  - Read/write, append, update-in-place (changing length), insert, and delete
  - Different types of records format such as fixed, variable, spanned, and undefined
  - Different types of access as: sequential, key direct, RBA direct, skip sequential

JRIO allows record-oriented applications (supporting multiple file systems) run using files on different file systems. It also provides a set of z/OS native code drivers to access these items:

- VSAM KSDS data sets
- Non-VSAM record-oriented data sets (sequential or random access)
- The system catalog that lists the high-level qualifiers (HLQ) from the system catalog and data sets for an HLQ
- Partitioned data set (PDS) directory
- HFS for sequential and random I/O access to *records*. The HFS support uses pure Java code to provide a set of concrete classes and directory classes that use the underlying java.io. JRIO also provides navigational support for HFS directories.

To run a JRIO application, Java commands implicitly set the CLASSPATH for the JRIO classes. The JRIO classes are in the same subdirectory as the Java for z/OS classes. Update your CLASSPATH to include the application classes by using the following Shell command:

```
export CLASSPATH=.:/u/joe/java/myclasses:$CLASSPATH
```

In this example, the class loader first scans the current directory for the application classes. If that fails, the class loader then scans the `/u/joe/java/myclasses` directory.

To run the JRIO sample programs, update CLASSPATH to include the JRIO sample classes by using the following Shell command:

```
export CLASSPATH=$CLASSPATH:$JAVA_HOME/recordio:
                    $JAVA_HOME/recsamp.jar/
```

JRIO and VSAM

JRIO provides indexed I/O access to records within a VSAM KSDS. The VSAM support uses z/OS native code to provide a set of concrete classes that implement the KeyedAccessRecordFile class. This configuration allows you to access records in these ways:

- In entry sequence order
- By primary unique key
- By alternate unique or non-unique key

For more information and a set of APIs that you can use to access KSDS VSAM data sets, see “JRIO API examples” on page 382.

2

Managing your VSAM data sets

This chapter provides practical tips about the daily maintenance of your VSAM data sets. This chapter includes the following sections:

- Reorganization considerations
- Sharing VSAM data sets
- Extended format
- Extended addressability (EA)
- Catalog Search Interface
- Major sources of VSAM processing options
- Media Manager, Open, Close, EOV in VSAM
- VSAM and 64 bits in real storage
- Special considerations for COBOL users and SMB

2.1 Reorganization considerations

Reorganizing a non-sequential data set achieves these goals:

- Consolidate the data set in fewer extents.
- Avoid internal empty spaces.
- Ensure that the physical sequence of the logical records matches the logical sequence of the logical records (for example, by key).
- Decrease the sequence (and the amount) of pointers in indexes structures that are coupled with data.

Historically, the performance benefits of data set reorganization afterward justify the amount of resources that are consumed during the reorganization. In addition, the business application data is not available during the reorganization process.

If this availability problem is unacceptable, there is now a solution in DB2. You can reorganize the data while transactions are reading and updating the data set. You can also reorganize a data set by doing an initial load from a backup copy.

One reason to reorganize a data set is to avoid long 3390 seeks and 3390 rotational position sensing (RPS) misses in a 3390 logical volume. These long seeks and RPS misses occur when the same data set is fragmented across the 3390 volume. However, the new DASD storage technology avoids these problems because a 3390 does not exist physically. In addition, you do not know how the 3390 logical tracks are mapped in the controller real disks.

The new DASD storage technology that changes the reasons for data set reorganization has these characteristics:

- Numerous high-capacity, small-size FBA, SCSI/SSA/FIBRE disks
- Redundancy by using different types of RAID
- Generous amounts of cache, mainly to avoid the write penalty that is caused by RAID 5 redundancy
- Plenty of Licensed Internal Code (LIC) lines of code to support RAID, cache algorithms, and the mapping between the disks and the logical 3390 volumes (as seen by z/OS)

There are several reasons to reorganize a VSAM KSDS (or VRRDS) cluster (data and index components):

- Reorganizing because of lots of CI/CA splits
- Reorganizing because of loss of useful space in data CA
- DB2 table space reorganization

2.1.1 Reorganizing because of lots of CI/CA splits

As mention previously, do *not* reorganize your KSDS data set when you have many previous CI/CA splits (as seen in Listcat output) just to avoid long seeks.

The other reason to reorganize a KSDS data set with many previous CI/CA splits is to avoid more splits. However, chances are that, after the reorganization, the number of splits will increase.

Usually, the performance effect of the split is caused by many extra I/Os that must be done (mainly at CA splits). After that, the data set has more space (maybe in the correct spot) for insertions, and normally the quantity of splits tend to decrease. Remember that splits generate more free space.

Try to avoid splits, if not possible. However, do not reorganize just because splits are occurring. Monitor the response time and number of CA splits. If you have a high number of splits, in the next IDCAMS Repro, set the FREESPACE option to (0,10).

Another justification for a KSDS reorganization is to decrease the number of index records and the number of pointers at the sequence set and at the index set. At every split, new index records are generated with the logical sequence implemented through pointers. During a reorganization of the data component, the index records are physically ordered. In certain cases, you can even decrease their amount.

However, nowadays most of the index records are kept at I/O buffers in central storage. Therefore, you will not improve the performance much through reorganization. In this case, reorganize at a low frequency.

2.1.2 Reorganizing because of loss of useful space in data CA

There were the following reasons for reorganizing a KSDS (VRRDS) data set because the loss of useful space in the data control area:

- Lack of control area reclaim
- Sequence set index CI size is too small
- CI/CA splits are causing free space in occupied CIs

Lack of control area reclaim

This reason is no longer applicable because of a new function that is called Control Area Reclaim. For more information, see 4.4.7, “Data Control Area Reclaim” on page 142.

Sequence set index CI size is too small

When the sequence set index CI size is defined too small, it cannot contain all information about all the data CIs at the correspondent data CA. An example is shown in Figure 2-1. In this case, the data CA is truncated and the last data CIs are not used. In this case, you must redefine the data set with a larger index CI size to reclaim such space.

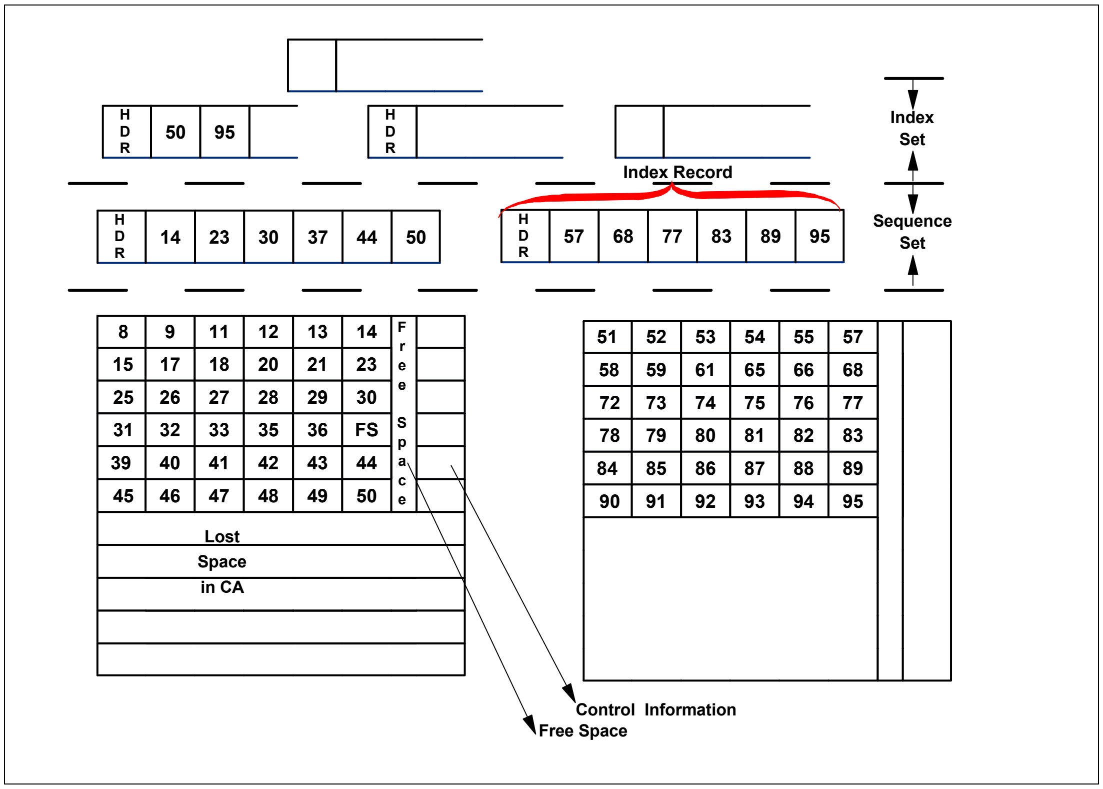


Figure 2-1 Lost space in data CA because the Index CI is too small

For example, DFSMShsm issues message ARC0909E advising that is time to reorganize its VSAM data sets (MCDS, OCDS, BCDS) because a free space threshold has been reached.

You can be informed of that space loss in these ways:

- Use your automation console function to detect the message in Figure 2-2 to generate an alert.

```
IDC3351I ** VSAM I/O RETURN CODE IS 212
Unable to split index; increase index CI size
```

Figure 2-2 Message because index CI size is too small

- Perform the following steps, based on the diagram that is shown in Figure 2-3. For more information, see Chapter 7, “VSAM problem determination and recovery” on page 345.

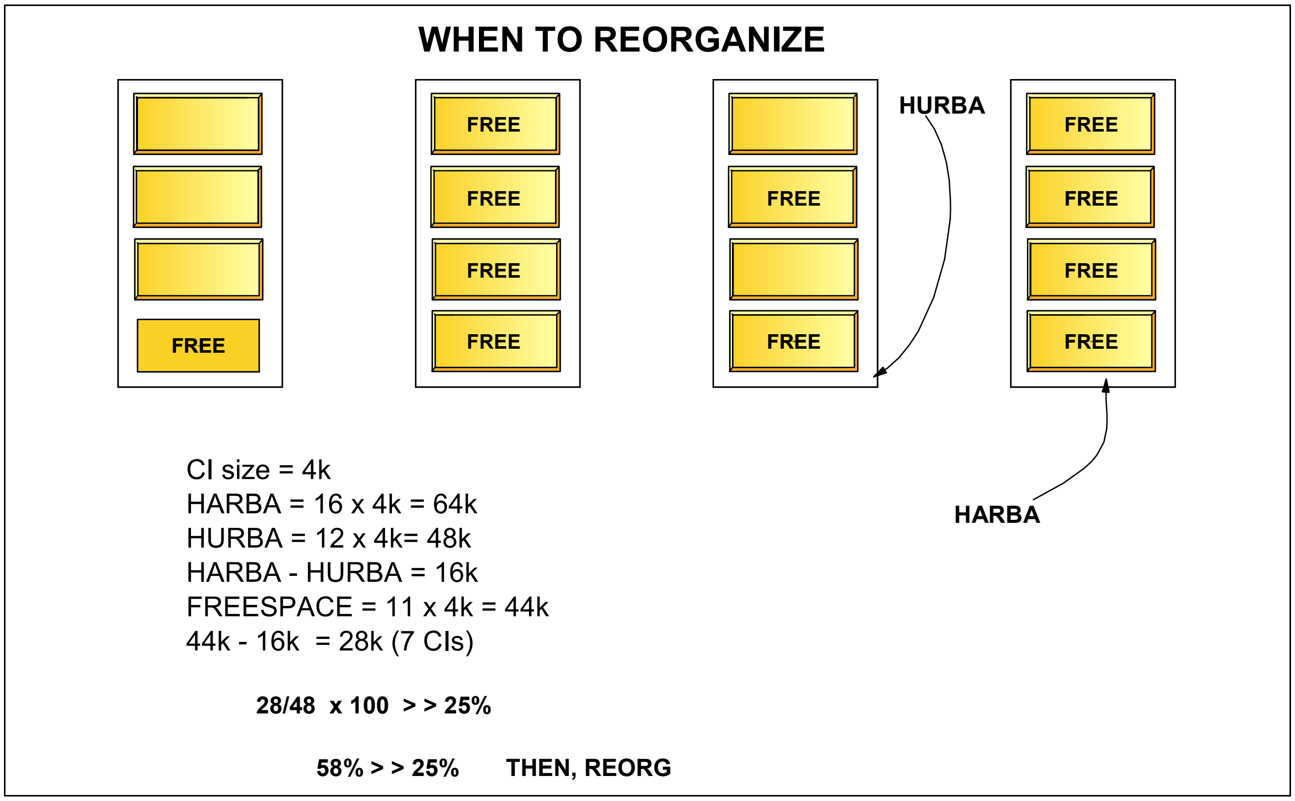


Figure 2-3 HARBA, HURBA, and free space

a. The example data component of a KSDS data set has 4 CAs and each CA has 4 CIs. There is just one CA after HURBA, with its four CIs totally free, as expected. In this scenario, the FREESPACE parameter is set at (0 25). This setting means that 25% (1 CI) of the 4 CIs in one CA (below HURBA) are free at initial load moment.

   In this example, some of the data CIs (below HURBA) are totally free. They are because the FREESPACE parameter and others because its CI index was too small (like the two second CIs at second CA and third CA).

   From Figure 2-3 showing HARBA, HURBA, and free space, you can derive:

   - HURBA = 11 x 4 KB = 44 KB
   - HARBA = 16 x 4 KB = 64 KB

   The formula, HARBA - HURBA = 20 KB, gives the number of free bytes in the totally free CIs contained in the CAs above HURBA.

b. The FREESPACE count in Listcat provides the number of free bytes contained in the totally free CIs below HARBA. If you subtract FREESPC(36 KB) - 20 KB = 16 KB, you get the number of free bytes in totally free CIs embedded in CAs below HURBA.

c. Divide 16 KB by 4 KB to discover that you have 4 totally free CIs below HURBA. This 16 KB includes the CA Freespace% of 25%, as defined at initial load time. Remember that CI splits modified these reserved free CIs into occupied CIs.

d. If (16 KB / HURBA) * 100 is consistently larger than CA Freespace% (25%), it means that it is time to reorganize because of empty CIs. The CI index is too low. The index CI size must be respecified as a larger value. Numerically, (16 KB/44 KB) x 100 = 37% is consistently larger than 25%.

Here is a numeric example for you to try:

```
HI-A-RBA-------184320000 (HARBA)
HI-U-RBA-------176209920 (HURBA)
FREESPACE-%CA---------10
FREESPC---------89616384
```

DFSMShsm issues message ARC0909E, advising that is time to reorganize its VSAM data sets (MCDS, OCDS, BCDS) because of a free space threshold that is reached.

Index CI size calculation algorithm

The modern algorithm is used by VSAM to calculate the KSDS and VRRDS index component minimum CI size when the index CI size is not informed in DEFINE command. It reduces the probability of small index CIs.

The higher key value in a data CI is stored in automatically compressed format in the Index CI record. To calculate the size of the CI index, VSAM assumes that those keys have a compression rate 3:1 (a conservative assumption). A key with 21 bytes, for example, after compression has 7 bytes. The size of the index CI is then derived by VSAM based on the other fields of the index entry and the size of the data CI and data CA.

CI/CA splits are causing free space in occupied CIs

After CI and CA splits, the free space per CI (excluding the totally freed) tends to increase. In sequential read processing, performance is affected because more free bytes are moved to storage. For more information, see 4.1.4, “I/O performance” on page 132.

The effect can be measure indirectly by the number of CI and CA splits that are shown in the catalog. Remember that the free space information in the catalog includes only the bytes in CIs that are totally freed.

2.1.3 DB2 table space reorganization

This book does not have any guidelines about when a DB2 table space should be reorganized. The main reason for such reorganization is to decrease the sequence (and the amount) of pointers in indexes structures that are coupled with data. However, this decision should be made by the DB2 administration staff when appropriated.

2.2 Sharing VSAM data sets

This section addresses the sharing of VSAM components, clusters, and spheres. They are shared by tasks from one address space (or enclave) throughout tasks that are in other address spaces (or enclaves) of the same or different MVS systems. The MVS systems must belong to the same IBM Parallel Sysplex®. This section presents the following important aspects:

- Level of data integrity
- Scope
- VSAM mechanisms to protect data integrity, including Global Resource Serialization (GRS) ENQ
- VSAM locks
- SHAREOPTIONS

Remember: Because of the complexity of sharing concepts, sometimes information is repeated in these sections.

2.2.1 Integrity

To protect the integrity of your VSAM data sets, VSAM uses these mechanisms:

- Internal VSAM locks at the control block structures or at a coupling facility lock structure (in the RLS mode case)
- Issues enqueue in SYSVSAM major name, which starts GRS, a z/OS component, that serializes the resource locally or globally. Do *not place* SYSVSAM resource name in GRS resource name list (RNL) exclusion list. This resource must be global (systems) all the time. The major name SYSVSAM is used to serialize many resources within VSAM control blocks and data sets (for example along CA splits).

Some of the consequences of Placing SYSVSAM in the GRS RNL exclusion list has these consequences, among others:

- Loss of records
- Overlaid and duplicate records
- Invalid index structures
- Invalid catalog records
- Invalid data set dumps and backups
- Incorrect restores
- Imprecise migrations
- Incorrect extents of the data set
- Incorrect release of space in the data set

If you do not follow these guidelines, problems in the following areas can occur:

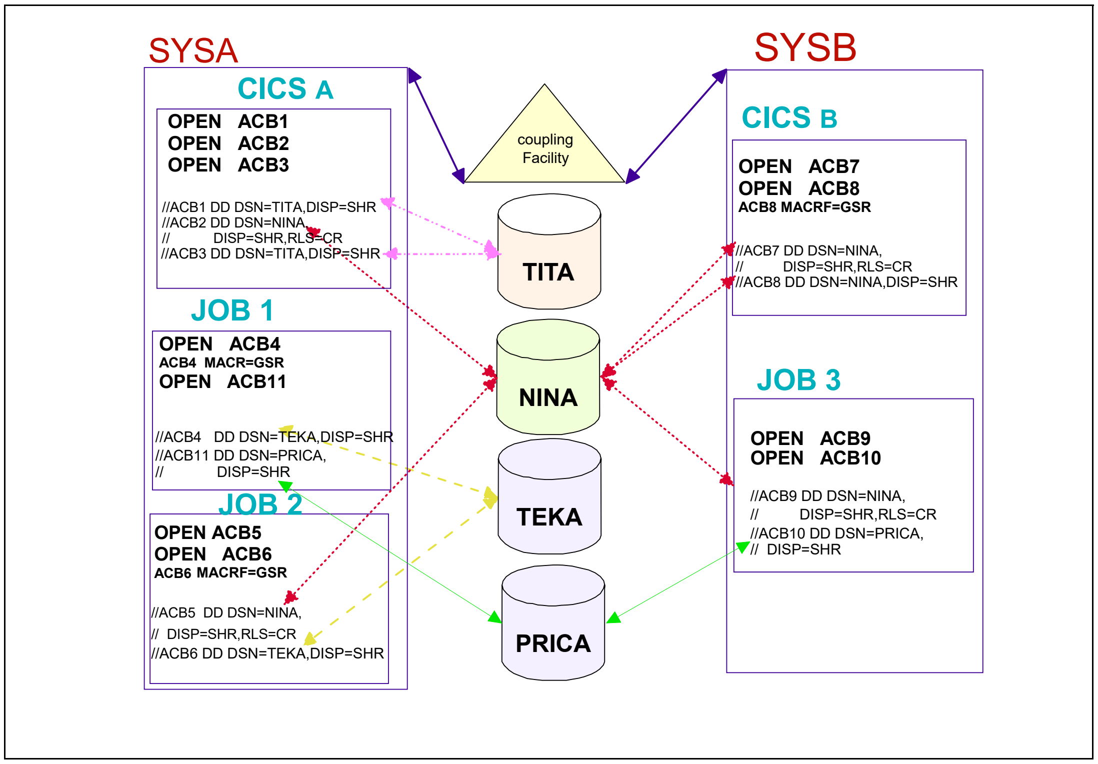
- Enforce VSAM SHAREOPTIONS:
  - VSAM no longer determines whether any user has a data set open on other systems.
- Serialize OPEN, CLOSE, and End of Volume (EOV) routines:
  - Multiple OPEN, CLOSE, and EOV requests towards the same data set can now occur at the same time on multiple systems, causing integrity risks.
  - Multiple extends of a data set can now occur on multiple systems, causing integrity risks.
- Serialize catalog updates:
  - A user can no longer determine whether the data set is open on another system, and so the catalog can be incorrectly updated.
  - Users cannot correctly serialize catalog updates of data set compression tokens.
- You cannot correctly determine the status of a data set for DELETE and RENAME processing.
- DFSMShsm cannot correctly serialize migrations with VSAM OPEN/CLOSE/EOV routines
- DSS cannot correctly serialize DUMP/BACKUP/RESTORE functions with VSAM OPEN/CLOSE/EOV routines.
- VSAM CLOSE cannot determine when the last user for a data set is closing the data set to correctly implement PARTIAL RELEASE.

- The system cannot recognize close failures so it can warn the user:
  - VSAM OPEN cannot determine whether the data set was improperly closed
  - Might not warn users of a close failure
  - Might not attempt to determine correct end of the data set.
- Unable to enforce record-level sharing (RLS)/non-RLS sharing rules.

2.2.2 Sharing VSAM data sets

It is possible to share VSAM data sets between these objects:

- Many program tasks that are running in the same address space. In this case, multiple ACBs point to the same VSAM data set, like data set TITA in Figure 2-4.
- One ACB shared in a task or different subtasks (daughter tasks), in the same address space, by using multi-string processing through Attach.
- Many program tasks that run in different address spaces, in a single z/OS. Tasks can even be running the same program logic. For example, data set TEKA in Figure 2-4.
- Different tasks in different z/OS images, such as data set PRICA and NINA in Figure 2-4.



```
OPEN   ACB1
OPEN   ACB2
OPEN   ACB3

//ACB1 DD DSN=TITA,DISP=SHR
//ACB2 DD DSN=NINA,
//          DISP=SHR,RLS=CR
//ACB3 DD DSN=TITA,DISP=SHR
```

```
OPEN   ACB4
ACB4  MACR=GSR
OPEN   ACB11

//ACB4   DD DSN=TEKA,DISP=SHR
//ACB11 DD DSN=PRICA,
//           DISP=SHR
```

```
OPEN ACB5
OPEN   ACB6
ACB6 MACRF=GSR

//ACB5  DD DSN=NINA,
//  DISP=SHR,RLS=CR
//ACB6 DD DSN=TEKA,DISP=SHR
```

```
OPEN   ACB7
OPEN   ACB8
ACB8 MACRF=GSR

//ACB7 DD DSN=NINA,
//         DISP=SHR,RLS=CR
//ACB8 DD DSN=NINA,DISP=SHR
```

```
OPEN   ACB9
OPEN   ACB10

//ACB9 DD DSN=NINA,
//         DISP=SHR,RLS=CR
//ACB10 DD DSN=PRICA,
//  DISP=SHR
```

Figure 2-4 Sharing VSAM data sets in the parallel sysplex

GRS data serialization

GRSr is a z/OS component that serializes common resources among tasks, even for tasks that are in several MVS systems in a Parallel Sysplex. GRS does not care about the nature of the resource. What matters is the symbolic qualified name, which is formed by its Major name (Qname) and Minor name (Rname).

The serialization is done in the resource symbolic name together with the type of serialization (shared or exclusive) indicating the protection type for such resource:

- When SHARED, any other shared access task is permitted (not placed in wait state). Any exclusive access is denied (through a return code) or queuing (the requesting task is placed in wait state until the resource becomes available).
- When EXCLUSIVE, GRS ensures that only the task that owns the resource can use it. Any other request to the same resource, whether it is exclusive or shared, is denied through a return code or queued. A queued requesting task is placed in wait state until the resource becomes available.

Through the options HAVE, TEST, and USE of the RET keyword at ENQ macro, the requesting task can ask the resource conditionally receiving back a return code, if the resource is unavailable.

The scope of the enqueue indicates whether the resource is serialized at address space level (scope STEP), at z/OS image level (scope SYSTEM) or sysplex-wide (scope SYSTEMS).

Before defining the level of sharing for a VSAM data set, evaluate the consequences of reading incorrect data (a loss of read integrity) and losing data during write updates (a loss of write integrity). These situations can result when one or more of the data set's users program tasks do not adhere to guidelines for accessing shared data sets. It is also important to avoid the unnecessary use of certain serialization functions, which can cause a performance degradation.

2.2.3 Write and read integrity

You can share a VSAM data set in the same program task, or among task programs that run in same or different address spaces. The set can be in same or different z/OS images. Regardless, you must determine what type of integrity (read or write) is required.

Write integrity

The write I/O operation ensures integrity for non-atomic writes (writes that are run in multiple I/O operations). Following are examples of non-atomic writes:

- Logical record update in-place, where you have three writes: Record is read, updated in memory, and written back to storage.
- CI index updates because of CI data insertions and deletions (KSDS/VRRDS)
- CI and CA splits, which involve several write I/O operations
- Data set creation at VTOC and catalog (usually VVDS) synchronous updates
- Program running writes for debit and credit

To have write integrity, when several tasks are accessing the same data set, VSAM must ensure that all the non-atomic writes are run without being pre-empted. No other intervening write I/O operation on the records that are involved in the ongoing non-atomic write can be run.

Read integrity

Read integrity ensures that the record you read is the most current and reliable copy available. Read integrity runs the following precautions:

- If the read is requested during non-atomic writes, the related read I/O operation is postponed.
- The read I/O operation is able to find the most current copy of the logical record:
  - If the requested data is in the buffer pool, VSAM must ensure the most current copy to satisfy the read (also called buffer pool coherency). In a shared environment with different z/OS images that access the data set, there are two ways of doing this:
    - Through RLS mode by using the Parallel Sysplex coupling facility for cross invalidation. For more information about cross invalidation, see Chapter 5, "VSAM Record Level Sharing" on page 209.
    - The VSAM refreshing the buffer at every read request. Specify this setting with SHAREOPTION 4.
  - The write operation must send the current copy to where the next read can access it, even in a multiple z/OS system. The location can be a shared 3390 volume (slow) or with RLS mode by using the Parallel Sysplex in a coupling facility structure.

2.2.4 VSAM sharing mechanisms

Understanding the sharing mechanisms requires an understanding of the VSAM control block structure.

VSAM control block structure

The VSAM control block structure for a data set is formed these objects:

- The input/output buffers and related control blocks. This set is called a resource pool.
- Other VSAM control blocks that are created at open time. These control blocks hold information that is related to the data set being opened, such as space available, type of VSAM data set, and buffering technique.

The control block structure is deleted when the last Close is issued on this z/OS towards the data set.

A VSAM control block structure can be shared by more than one program in the same or in different tasks. Do so by issuing the OPEN macro towards the same VSAM data set. The integrity of such data set can be ensured by VSAM if the control block structure can be shared between these program tasks. To ensure that you can share successfully within a task or between subtasks, ensure that VSAM builds a single control block structure for the data set. This control block structure includes blocks for control information that includes serialization locks and input/output buffers.

To be shared, the structure must be compatible and accessible.

The meaning of *compatible* has to do with what happens when a VSAM data set is opened. To open a VSAM data set, the ACB address must be declared to the Open function. For more information about ACB, see 2.6.1, "Access method Control Block (ACB)" on page 73. This ACB is used along with the open program, and some information about the ACB goes to the VSAM control block structure. For a new ACB to use an existing VSAM control block structure, the new ACB must be compatible with the existing (previous) control block structure options.

The ACB must be consistent in its specification of the following processing options:

- Data set specifications
- The ACB MACRF options DFR, UBF, ICI, CBIC, LSR, and GSR

A VSAM control block structure must also be *accessible*. Accessing a VSAM control block structure depends on its location:

- When using NSR or LSR buffering modes, the VSAM control block structure is in the address space private area (user region). It can be accessed by tasks that run programs in the same address space.
- When using RLS, the VSAM control block structure is in SMSVSAM data spaces or optionally at the SMSVSAM address space private area (above the bar). Some of the control block structure is copied at the coupling facility.
- For GSR, the VSAM control block structure is at the extended common storage area (ECSA). It can be accessed by task running programs in different address spaces in the same z/OS system.

For more information about NSR, LSR, GSR, and RLS, see Chapter 4, "VSAM performance" on page 127.

At open time, before creating a control block structure, VSAM verifies whether an *accessible* and *compatible* VSAM control block structure of this data set already exists:

- If the block structure exists, VSAM uses the same control block structure and implements a serialization mechanism (through locks that are in this same control block structure) to ensure data integrity. This process is done at the CI level (not at logical record level) by VSAM itself.
- If a block structure does not exist, VSAM creates a control block structure for the data set. Also, VSAM uses the SHAREOPTIONS value to determine the level of data set sharing to implement protection at data set level through enqueue (ENQ) in the major SYSVSAM Qname. The ENQ Rname is the name of the data set itself.
- If the block structure exists but is not compatible or not accessible, or it does not exist in other MVS in a Parallel Sysplex, VSAMs use the SHAREOPTIONS value to determine whether creating a control block structure would violate the SHAREOPTIONS rules. If so, the OPEN fails, otherwise the OPEN is successful. For more information, see "VSAM SHAREOPTIONS" on page 60.

  VSAM verifies whether a control block structure exists in other MVS in the Parallel Sysplex. It does so by issuing ENQ with the TEST option in the Qname SYSVSAM to resources that represent input and output control blocks' VSAM data set components. If the control block structure exists in another MVS system in the Parallel Sysplex, the SHAREOPTIONS rules apply, to determine whether the Open succeeds or fails.

Implementing VSAM data set integrity

There are five important ways to implement VSAM data set integrity according to how the data set is shared:

- Internal locks for non-RLS access, when you are using a single VSAM control block structure. The VSAM lock mechanism controls data sharing in a single control block structure. These locks are in the shared control block structure. The VSAM control block structure plays a decisive role in VSAM data sharing. When you share a VSAM data set using a *single and only* VSAM control block structure, VSAM ensures *read and write integrity* at CI level. This configuration is independent of the SHAREOPTIONS or JCL DISP parameter specification.

- Global lock for RLS access, a sophisticated mechanism that uses the coupling facility. RLS can be used when you share a data set throughout the Parallel Sysplex. SHAREOPTIONS does not apply to VSAM data sets in RLS mode because RLS uses a single control block structure through the sysplex. For more information about sharing VSAM data sets in RLS mode, see Chapter 5, “VSAM Record Level Sharing” on page 209.
- SYSVSAM ENQ issued by VSAM, according to the data set SHAREOPTIONS, for non-RLS access that involves multiple VSAM control block structures. At the same time, VSAM cannot ensure who has exclusive use of the data. This limitation is because the control block structure is not the same, and the locks are not as well. In this case VSAM applies SHAREOPTIONS rules, serializing at cluster or component level.
- The GRS serialization function issued (ENQ macro) by the data set allocation routine, as requested by the initiator according to the data set disposition at DD statement. The Qname is SYSDSN and the Rname is the data set name. The scope of the serialization is sysplex wide. The resource Status is shared for SHR data set disposition and exclusive for NEW, OLD, and MOD disposition.
- The ENQ serialization function can also be implemented by the user task application program. When the SHAREOPTIONS being used do not protect the data set, for example (3 3), the user application programs must provide data set integrity. They can do so by using ENQ or RESERVE macros.

Creating a single VSAM control block structure

You can use these methods to create a single VSAM control block structure for a VSAM data set while you are processing multiple concurrent requests:

1. A single access method control block (ACB) and an STRNO>1. For more information about STRNO, see *z/OS DFSMS Using Data Sets,* SC26-7410.
2. DD name sharing. Use this method when you have multiple ACBs all in the same address space, pointing to a single DD statement. This exists when multiple tasks, in the same address space, are running the same program concurrently. It also exists as in Example 2-1, in assembly language, when multiple tasks are running different programs, but pointing to same DD.

Example 2-1 DD name sharing that points to a single DD statement

```
JCL:
//DD1 DD DSN=ABC

task A executing PGM1:
PGM1:      OPEN ACB1
      ACB1 ACB DDNAME=DD1

task B executing PGM2:
PGM2:      OPEN ACB2
      ACB2 ACB DDNAME=DD1
```

3. Data set name sharing, with multiple ACBs pointing to multiple DD statements with different DDnames, but *with the same DSname*. The data set names are related with an ACB open specification (MACRF=DSN). This MACRF option means that subtask shared control block connection is based on common data set names as shown in Example 2-2.

Example 2-2 DD name sharing with different DD names

```
JCL:
//DD1 DD DSN=ABC
```

```
//DD2 DD DSN=ABC

task A executing PGM1:
PGM1:      OPEN ACB1
      ACB1 ACB DDNAME=DD1,MACRF=DSN

task B executing PGM2:
PGM2:      OPEN ACB2
      ACB2 ACB DDNAME=DD2,MACRF=DSN
```

For data sets with AIX, VSAM connects an ACB to an existing VSAM control block structure for data set name sharing *only* when the base of the sphere is the same for both ACBs. In Example 2-3, supposing no VSAM control block structures exist:

- When task A issues OPEN to the cluster, VSAM creates block structure for the cluster and for the alternate index. VSAM acquires the name of the AIX from the catalog.
- When task B issues OPEN to the PATH, VSAM adds the alternate index, and because the base of the sphere is the same, the cluster to the existing structure.
- When task C issues OPEN to the alternate index, VSAM creates a control block structure for the alternate index. VSAM does not add to the existing structure because the base of the sphere is not the same. SHAREOPTIONS are enforced for AIX_A data set name because multiple control block structures exist.

Example 2-3 Two VSAM control block structure to same data set

```
JCL:
//DD1 DD DSN=CLUSTER_A
//DD2 DD DSN=PATH CONNECTING AIX_A TO CLUSTER_A
//DD3 DD DSN=AIX_A

task A executing PGM1:
PGM1:      OPEN ACB1
      ACB1 ACB DDNAME=DD1,MACRF=DSN

task B executing PGM2:
PGM2:      OPEN ACB2
      ACB2 ACB DDNAME=DD2,MACRF=DSN

task C executing PGM3:
PGM3:      OPEN ACB3
      ACB3 ACB DDNAME=DD3,MACRF=DSN
```

In order for a new ACB use an existing VSAM control block structure, the new ACB must be compatible with the existing control block structure. If compatibility cannot be established, OPEN tries to build a new control block structure. If it cannot, OPEN fails. The reason for the failure might be the limitations of the share options that were specified when the data set was defined.

When you implement intra-address space sharing and attaching new subtasks, the *subpool zero must be shared* by mother and daughter tasks to share the control block structure. In this configuration, SZERO=YES in the ATTACH macro.

A subtask can have an opened ACB sharing a control block structure that has been previously used. In this case, if this subtask issues the POINT macro to obtain the position (through the place holder) for the data set, do not assume that positioning is at the beginning of the data set.

When VSAM cannot use an existing control block structure for a data set, VSAM uses the SHAREOPTIONS. For more information, see “VSAM SHAREOPTIONS” on page 60.

2.2.5 Sharing data in a single VSAM control block structure

After you create a shared control block structure, you can use it for sharing a data set. If the sharing of a VSAM data set is done using a single and only VSAM control block structure throughout the MVS systems of the Parallel Sysplex, VSAM ensures read and write integrity. This is independent of the SHAREOPTIONS or JCL DISP specifications. The control block structure is accessible and compatible between all the program tasks.

The locks that are used for the serialization process are at the single and only VSAM control block structure.

Table 2-1 shows how to implement VSAM data sharing with read and write integrity in the z/OS systems of Parallel Sysplex.

Table 2-1 One VSAM control block structure

Buffering Technique	One Control Block structure in the Parallel Sysplex
NSR/LSR	All OPENs to data set issued in the same address space
GSR	All OPENs to data set issued in the same z/OS system
RLS	Anywhere

With NSR or LSR, many tasks (or subtasks) concurrently OPEN the same data set. These tasks can use only one VSAM control block structure because they are issued in the *same address space*.

The ability to perform multiple OPENs to the same data set within a task or from different tasks in a single address space that shares *a single VSAM control block structure* is called inter-address space sharing. It is also called *subtask sharing*. Each task uses at least one string to access data in the data set. Subtask sharing allows many logical views of the data set while it maintains a single VSAM control block structure. With a single control block structure, you can ensure that you have exclusive control of the buffer when you update a CI. An example of the use of single VSAM control block structure is in Figure 2-4 on page 52. The data set “Tita” in CICS-A, is accessed by ACB1 and ACB3, both pointing to same data set.

When you use GSR, the same VSAM control block structure can be used by program tasks in *different address spaces in the same z/OS system*. This configuration is possible because the control block structure is in the ECSA. In Figure 2-4 on page 52, JOB 1 and JOB 2 are using GSR in the same z/OS system and access the same data set. They are using the same VSAM control block structure for access the data set “Teka.” That is the only VSAM control block structure in all z/OS systems in the Parallel Sysplex. Before you use GSR, see “Global shared resource (GSR)” on page 162, for its disadvantages. Generally, use RLS instead of GSR because of these disadvantages.

Integrity with multiple strings in several tasks

VSAM also ensures read and write integrity in multiple string processing that is implemented through several tasks. In this processing environment, you can have multiple independent and concurrently request parameter lists (RPLs) for the same data set, by using a single

VSAM control block structure. Each I/O access is requested by the application to VSAM through an RPL control block:

- When your program issues a GET non-update request, VSAM first tries to locate the control interval in buffers that are attached to its string. It either obtains a copy of the CI (with NSR) or shares the copy in the buffer (LSR). If VSAM does not find the CI in buffers, it reads the CI from DASD. In this case, no CI lock request is done by VSAM. There is no read integrity for such type of access because no lock is required.
- For GET update requests (meaning a read and afterward an update in place write), the CI is obtained in exclusive control through VSAM lock mechanism. The CI is then read from the device for the latest fresh copy of the data. If the buffer is already in exclusive control of another string, the request fails with an exclusive control feedback code. Exclusive control has these rules:

  a. If a string obtains a record with a GET for update request, the CI is not available for update or insert processing by another string. However, reads are allowed.

  b. If a string is in a control area split caused by an update with length change or an insert, that string obtains exclusive control of the entire control area. Other strings cannot process insert or update requests against this control area until the split is complete.

An exclusive control conflict happens when a request is made for a resource that is in exclusive control of another request. VSAM treats in two different ways, according to the buffering mode you are using:

- With LSR or GSR, VSAM handles the request according to the information in the MACRF parameter in ACB:
  - LEW: The request is queued, placing the requiring task in wait until the resource become available. This configuration is the default.
  - NLW: VSAM returns exclusive control error return code X’14’ to the application program, and the application program determines the next action.
- With NSR, VSAM does not queue requests that have exclusive control conflicts. VSAM always returns a logical error return code, and you must stop activity and clear the conflict. If the RPL that caused the conflict had exclusive control of a control interval from a previous request, the application program should issue an ENDREQ before attempting to clear the conflict and avoid a dead lock. The conflict can be solved in one of three ways, all of them at the application program level:

  a. Queue the current request internally until the RPL holding exclusive control of the control interval releases that control, and then reissue the request.

  b. For assembly language, issue an ENDREQ against the RPL holding exclusive control to force it to release control immediately.

  c. When you are using LSR and Assembly source program, release the buffer by running the macro MRKBFR with the option MARK=RLS.

  If the RPL includes MSGAREA and MSGLEN, the address of the RPL holding exclusive control is provided in the first word of the MSGAREA. The RPL field, RPLDDDD, contains the RBA of the requested control interval.

Sometimes this locking on a Control Interval basis done by VSAM might cause contention and even deadlocks. There are no deadlock detection and prevention algorithms that are implemented in VSAM, except in an RLS environment. If you are facing these drawbacks, you can use fewer logical records in a data control interval, also called better lock granularity. You can also move your application to RLS buffering where the serialization granularity is at logical record level.

2.2.6 Sharing data with many VSAM control block structures

This section addresses sharing data sets throughout the MVS systems of a Parallel Sysplex when *more than one* VSAM control block structure exists. This configuration happens when an OPEN to a data set cannot use an existing VSAM control block structure.

VSAM SHAREOPTIONS

The VSAM SHAREOPTIONS installation option plays an important role in VSAM integrity. It specifies whether and to what extent a data set is to be shared among tasks in one or multiple z/OS systems in the Parallel Sysplex. When you define VSAM data sets, you specify how the data is to be shared by using the SHAREOPTIONS parameter of DEFINE command.

The serialization mechanism that is used by VSAM to serialize data sets, as indicated by SHAREOPTIONS, is the GRS ENQ in the name SYSVSAM.data_set_name, with the scope systems. Make sure that the entity to be serialized is the data set itself, and not the control interval or even the logical record.

The share options are specified as `SHAREOPTIONS(x,y)`. The *x* and *y* parameters are referred as cross-region and cross-system. However, the scope of the ENQ is systems. SHAREOPTIONS is a *sysplex-wide* serialization mechanism. The names cross-region and cross-system are used to avoid confusion and be compatible with such terminology, but keep in mind that they do not mean that.

The basic sharing is still controlled throughout VSAM control block structures. However, VSAM uses the SHAREOPTIONS values to determine whether creating a control block structure would violate the SHAREOPTIONS rules. If so, the OPEN fails. Otherwise, the OPEN is successful.

VSAM uses GRS enqueue services to implement the level of sharing requested in SHAREOPTIONS when you are creating a control block structure at OPEN time. VSAM issues enqueues to the major name SYSVSAM with the scope systems that protects the data set through GRS in the Parallel Sysplex.

When a cluster has an alternate index, the following processes occur:

- If the data set opened is the *path or the base cluster*, VSAM creates control block structure and issues ENQ, for these components:
  - The base cluster components. For example, if the data set is KSDS, VSAM creates control block structure for the data and for the index components.
  - The alternate index components.
- If the data set opened is the alternate index, VSAM issues enqueue *only* to the alternate index components (for example, data and index).

For each new VSAM control block structure that is created, VSAM issues enqueue to enforce SHAREOPTIONS.

Cross-region options

The following options are valid for serializing the data set among MVS systems in the Parallel Sysplex:

(**1**): This SHAREOPTIONS means that can be only one VSAM OUTPUT control block structure throughout the Parallel Sysplex or any VSAM INPUT control block structures in the Parallel Sysplex. With SHAREOPTIONS, the granularity of the serialization is at data set level. The value of *one* is the most restrictive toward serialization.

Except for those OPENs that use the same VSAM control block structure, VSAM ensures only one OPEN for output or many OPENs for input only. Any other OPEN fails with a return code in ACB. If the data set is open for input, other OPENs creating a control block structure can open the same data set for input successfully. If the intention is to open the data set for output, the open fails with a return code in ACB. VSAM thus ensures total *read and write integrity*. The intention of input or output is declared in the ACB MACRF parameter, not in the OPEN input or output options.

For *output*, VSAM issues *exclusive* enqueue (in SYSVSAM major name and the data set name as the minor name) for the input and output VSAM control blocks structure for each VSAM component. This configuration ensures that there is just one control block structure in the whole Parallel Sysplex. If another OPEN is issued against the same data set that is not using the same control block structure, VSAM does not queue the open. Instead, VSAM fails it with a return code.

For *input*, VSAM issues *shared* enqueue (in SYSVSAM major name and the data set name as the minor name) only for input VSAM control blocks structure for each VSAM component. OPENs for input are allowed. If an OPEN for *output* is issued against the same data set that is not using the same control block structure, VSAM does not queue the open. Instead, VSAM fails it with a return code.

When the VSAM data set is empty and is being open for OUTPUT, no matter the SHAREOPTIONS, *only one* VSAM OUTPUT control block structure can exist throughout the Parallel Sysplex. VSAM processes the data set as having SHAREOPTIONS(1,x).

When a data set is allocated with OLD disposition at JCL, VSAM treats the data set as having SHAREOPTIONS(1,x).

(**2**): VSAM ensures that there is only one VSAM control block structure for output in the MVS systems in the Parallel Sysplex and many VSAM control block structures for input. VSAM thus ensures write integrity. For OPEN output, the enqueue exclusive is issued, but there are not enqueues for OPEN input. If you require read integrity (with a better performance than cross-region (1)), use the ENQ and DEQ macros to provide that read integrity.

VSAM fails the OPEN output when an output VSAM control block structure to the same data set in the MVS systems in the Parallel Sysplex already exists.

(**3**): VSAM allows *many* OPENs to the data set for output or input. The user application programs must ensure both read and write integrity through their own ENQs. VSAM does not issue the enqueue at Open time.

(**4**): VSAM allows *many* OPENs to the data set for output or input. The user application programs must ensure both read and write integrity through their own ENQs. VSAM does not issue the enqueue at Open time just like option 3. Furthermore, with option 4, VSAM refreshes (through I/O operations) the *data and index* components buffer pools for direct processing. However, the sequence set is not refreshed. This refresh ensures the coherency of the data in the buffer pool. Coherency in this case means that the task gets the most updated contents of the requested record. Recall that such data can be modified by other MVS in the Parallel Sysplex and the most update copy is at the data set in storage.

Cross-system options

As mentioned before, the term cross-system is incorrect. Cross-system options provide the same capability as cross-region options 3 and 4. They are just a way to provide data sets that are using cross-region options 1 and 2 with the capabilities of 3 or 4.

You can use cross-system options to have SHAREOPTIONS 1 or 2 with (4) or without (4) buffers refreshment. Cross-system options can also be used to run the VSAM function named Control Block Update Facility (CBUF):

(**3**): VSAM does not refresh the buffer pools for direct processing. When `SHAREOPTIONS(3,3)` or `SHAREOPTIONS(4,3)` is used, VSAM provides a CBUF.

CBUF function is active, whenever a data set is opened with `DISP=SHR`, and `SHAREOPTIONS(3,3)` or `SHAREOPTIONS(4,3)`. With CBUF, VSAM record management maintains a copy of the data set critical control block data in z/OS common storage (CSA). This control block data is called VSAM shared information (VSI) Obviously, this common storage is available only to address spaces within this MVS. Passing this information to another MVS is a responsibility of the application. Passing such information across other MVS systems eliminates the restriction that prohibits control area splits with share options 3. However, under share options 4 these restrictions still exist.

Cross-system sharing that allows CA splits can be accomplished (with integrity) by sending the VSI blocks to the other MVS systems at the conclusion of each output request. Every time a data set is opened on a system for CBUF processing, a VSI is built for the data set and added to the VSI chain. This control block is then updated by the user to communicate information from one address space to another. Generally, the VSI sent to other z/OS images is not changed, and only a check occurs. For more information and an example about how to get the VSI, see "Accessing the VSAM Shared Information (VSI)" on page 383.

When you send the VSI to the other z/OS image, you can choose one of these options:

- XCF APIs, as IXCJOIN, IXCMSGO, IXCMSGI macros
- APPC IBM VTAM® LU 6.2, as RECEIVE_AND_WAIT and SEND_DATA commands.

Remember that you must continue to provide read and write integrity. VSAM ensures that tasks have correct control block information if serialization is done correctly. Also, option 3 does not cause buffer pool invalidation, as option 4 does.

Because programs in many address spaces (regions) can share a data set, an error that occurs in one address space can affect programs in other address spaces that share that data set. If a logical error (return code 8 at register 15) or physical error (return code 12 at register 15) is detected, any control block changes made before the error was detected are propagated to storage.

For examples about how to implement VSAM sharing with integrity, see "Techniques of Data Sharing" in *z/OS DFSMS Using Data Sets,* SC26-7410.

2.2.7 General share options considerations

User tasks running user programs that ignore the write integrity guidelines in share options 3 and 4 can cause these problems:

- VSAM program checks
- Lost or inaccessible records
- Incorrect data set failures
- Other unpredictable results.

This option places responsibility for data integrity on each user application program that shares the data set. For more information, see 7.3.2, "Broken VSAM data set" on page 362.

User programs that ignore the read integrity guidelines in share options 2, 3, and 4 result in down-level data records and erroneous no-record-found conditions.

In cross-region option 4 and cross-system option 4, buffers for direct processing are refreshed by VSAM for each request. Buffering does not save I/O operations. Option 4 has the following considerations:

- Each PUT request results in the appropriate CI buffer being written immediately into the VSAM data set's storage device. VSAM writes out the CI buffer in the user's address space that contains the new or updated data record. The data and sequence set control interval buffers are marked invalid following the I/O operation to storage.
- Each GET request results in all the user's input CI buffers being refreshed. The contents of each data and index buffer that are used by the user's program are retrieved from the VSAM data set's storage device through an I/O operation.
- For GRS serialization, the Open/Close/EOV routines use ENQ/DEQ in SYSVSAM.dsn.catname.I|O|B, to implement serialization when processing a VSAM data set. This configuration ensures correct sharing that is based on share options. To avoid integrity exposures, *never* add SYSVSAM Qname to the GRS RNL exclusion list. This enqueue resource must be global, that is, scope equal to systems.
- During initial load mode processing (as in a Repro process), you cannot share data sets. Share options are overridden during load mode processing to (1 3). For more information, see 4.4.11, “Initial load option” on page 145.
- *Data and sequence sets* buffers for direct processing (for reads and writes) are refreshed for each request. Index buffers are not. Output processing is limited to update and add records that do not change the high used RBA (HURBA). HURBA is the RBA of the high key data control interval. The control area then splits. The addition of a new high-key record for a new control interval that results from a control interval split is not allowed. VSAM returns a logical error to the user's program when this condition occurs. If the task that is running your program does not satisfy the requirements described above, you require cross-system option 3. In option 3, CA splits and addition of a new high-key record are allowed because of CBUF.

Table 2-2 contains a summary of the share options, not including cross-system options 1 and 2.

Table 2-2 Relationship between share options and VSAM functions

Share Options (DISP=SHR)	**VSAM function that is provided**
(3 3)	CBUF (CA splits) and no buffer refresh
(3 4)	Data/Seq Set buffers invalidate. No CBUF (no CA splits)
(4 3)	CBUF and Data/Index buffers invalidate
(4 4)	Data/Seq Set/Index buffers invalidate. No CBUF (no CA splits)

2.2.8 Protecting VSAM data set through DISP parameter

When a data set is allocated, the data set disposition in the DD statement starts an enqueue (issued by the z/OS Allocation routine requested by the Initiator) in the major name SYSDSN. The minor name is the data set name. The scope of the enqueue is throughout the Parallel Sysplex unless the data set name is the GRS exclusion list. The enqueue is SHARED for shared data set disposition, and EXCLUSIVE for OLD, MOD, or NEW data set disposition.

Remember these considerations if you protect your VSAM data sets by using its disposition:

- The enqueue is done only for the cluster name, not for the sphere components.
- When a data set disposition of OLD is used, it is practically equivalent to setting SHAREOPTIONS(1,x).

An AIX example is shown in Example 2-4. The enqueues in SYSDSN are issued on behalf of the two Jobs (J1 and J2) because of they are allocated in two different resource names. The resource names are PATH_OF_CLUSTER_A (the base cluster), and CLUSTER_A (the sphere).

With the disposition OLD in JOB J1, VSAM treats the base cluster as SHAREOPTIONS(1,x). The OLD disposition in J1 does not prevent allocating CLUSTER_A in JOB J2. However, while JOB J1 is running with DD1 opened, if JOB J2 starts and tries to open the data set, it receives an OPEN error. This error occurs because one control block structure for output already exists.

Example 2-4 VSAM data set that is allocated with DISP=OLD

```
SPHERE: CLUSTER_A (Base of the sphere)
        AIX_OF_CLUSTER_A
        PATH_CLUSTERA (Connecting the alternate index to the base cluster)

JOB J1 OPEN FOR OUTPUT:
//DD1 DD DSN=PATH_OF_CLUSTER_A,DISP=OLD

JOB J2:
//DD2 DD DSN=CLUSTER_A,DISP=SHR
```

Before you use DISP=OLD to protect VSAM data sets, see 2.2.6, “Sharing data with many VSAM control block structures” on page 60. This section addresses how VSAM creates control block structure according to the component that is being opened. If you intend to use DISP=OLD, do not forget to allocate the VSAM data set components with DISP=OLD as well.

2.3 Extended format

Extended format is a technique that affects the way count key data (CKD) is stored in a 3390/3380 logical track. Extended format was first introduced to implement data striping. It increases the performance and the reliability of an I/O operation.

Generally, convert your data sets to extended format for better performance, more function, and improved reliability. A good time to convert to extended format is when you reorganize your VSAM data set.

All VSAM data set organizations can be defined in extended format, including KSDS, ESDS, RRDS, VRRDS, and LDS. Extended format data sets have the following benefits:

- Data striping
- Data compression
- VSAM extended addressability
- Partial space release
- System-managed buffering

2.3.1 Extended format characteristics

When you create a VSAM component with the extended format option, there are two major differences from those not in extended format:

- If a data set is allocated as an extended format data set, 32 bytes are added to each physical block. This physical block suffix might increase the amount of space that is needed for the data set, depending on the CI size.

For example, in a 3390 device with a CI size of 18432 bytes, you need 12.5% more space. The 32-bytes suffix contains:

  – A relative record number
  – A 3-byte field to detect controller invalid padding, thus improving the availability of the I/O operation

- The length of the data portion of the physical block is 4K B, a type of fixed block architecture.

The block format and the suffix are transparent to the application. The application does not require internal or external modifications to create and use the extended format data set.

Extended format data sets must be system-managed. A system-managed data set must have an associated storage class and be in a system-managed volume. Extended format data sets are described in the catalog as striped data sets with a stripe count of one. When a data set is allocated as an extended format data set, the data and index are extended format. Any alternate indexes that are related to an extended format cluster are also extended format.

Certain types of key-sequenced data set types cannot be allocated as extended format:

- Catalogs
- System data sets (SMS is not active at early IPL stages)
- Temporary data sets

If a data set is allocated as an extended format data set, 32 bytes are added to each physical block. This physical block suffix might increase the amount of space that is needed for the data set, depending on the CI size. For example, in a 3390 device with a CI size of 18432 bytes, you need 12.5% more space.

2.3.2 Creating an extended format data set

To convert a non-extended format data set to extended format, or to allocate an extended format data set, you must create an SMS data class (DC) with the DATASETNAMETYPE field equal to EXT. You must then assign the data sets to that data class.

2.4 Extended addressability (EA)

Starting with DFSMS/MVS 1.5, support for extended addressability is extended to all VSAM record and buffering organizations. With DB2 V1.6 and later, EA can be used for very large DB2 table spaces. DB2 uses a VSAM Linear Data Set for storing data and DB2 indexes, dictionaries, and catalogs.

With EA, the 4-GB architectural limit for data set size that is imposed by using the 4-byte field for the relative byte address (RBA) is eliminated. With EA, the RBA contents field now has 8 bytes.

Extended addressability and extended format are not the same concept. Extended format is a way of storing data in a 3390 track volume. Extended addressability allows larger VSAM data sets. However, extended format is a pre-prerequisite for extended addressability.

Using EA, the size limit for a VSAM data set in one of these ways:

- CI size is multiplied by 4 GB because the CI number within VSAM control blocks is still a 4-byte field
- The 3390 volume size is multiplied by 59 because DSCB limitation in VTOC

A 4-KB CI size yields a maximum data set size of 16 TB. A 32-KB CI size yields a maximum data set size of 128 TB. A 4-KB CI size is preferred by many applications for performance reasons (for random and sequential accesses). No increase in processing time or performance deterioration are expected for extended format data sets that grow beyond 4 GB.

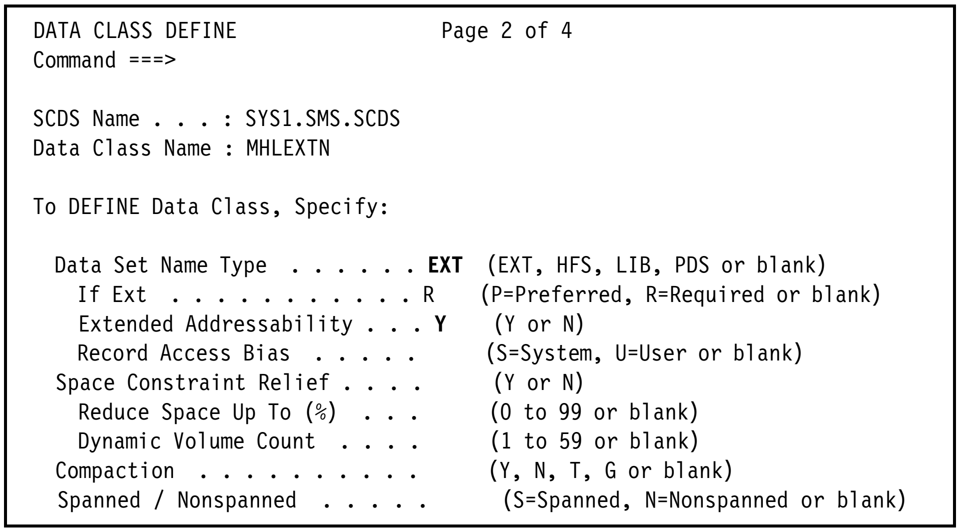
To use EA, the data set *must* have these characteristics:

- SMS-managed
- Defined as extended format

EA is available to a data sets associated to a data class defined with these characteristics:

- DSNTYPE=EXT and
- EXTENDED ADDRESSABILITY=Y

Figure 2-5 shows the ISMF panel where you specify EF and EA when you define or alter a data class.



```
DATA CLASS DEFINE                 Page 2 of 4
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : MHLEXTN

To DEFINE Data Class, Specify:

  Data Set Name Type  . . . . . . EXT  (EXT, HFS, LIB, PDS or blank)
    If Ext  . . . . . . . . . . . R    (P=Preferred, R=Required or blank)
    Extended Addressability . . . Y    (Y or N)
    Record Access Bias  . . . . .      (S=System, U=User or blank)
  Space Constraint Relief . . . .      (Y or N)
    Reduce Space Up To (%)  . . .      (0 to 99 or blank)
    Dynamic Volume Count  . . . .      (1 to 59 or blank)
  Compaction  . . . . . . . . . .      (Y, N, T, G or blank)
  Spanned / Nonspanned  . . . . .      (S=Spanned, N=Nonspanned or blank)
```

Figure 2-5 DATA CLASS DEFINE ISMF panel

After you create a data class with these attributes, you can code the DATACLAS value on your DD statements. Or you can allow the ACS routines to assign the appropriate data class for your eligible data sets. Applications that access a VSAM extended format KSDS through a user-specified key can have very large VSAM components without making JCL or code changes.

Another method in which JCL can be used to create a KSDS with extended addressability is through the DD statement keyword LIKE.

With EA, the format of the index entry in the index CI changes. The data RBA pointer for a EA data component grows from 4 to 8 bytes.

For a normal application not that does not refer to a specific RBA, as in most applications, there is total compatibility when migrating the data set to EA.

To support EA, many DFSMS macros and commands have been changed. To take advantage of extended addressability, new macro parameters and subparameters that are related to RBA have been added:

- RPL macro, added the XRBA sub parameter to the OPTCD parameter to indicate that extended addressability is to be used. It must be used when you are processing a data set by its RBA. For example, OPTCD=(ADR,DIR,XRBA) instead of OPTCD=(ADR,DIR,RBA).
- TESTCB macro, added XADDR as a possible value to ATRB. It can be used to test if the data set is in extended addressability format.
- SHOWCB macro, added:
  - XAVSPAC parameter to obtain the amount of available space in the data component or index component, in bytes.
  - XENDRBA, to obtain the HURBA
  - XHALCRBA, to obtain the high allocated RBA (HARBA)

  These fields use 8 bytes to return the information, instead of 4 bytes used for non-EA data sets.

Applications that process an EF data set by RBA must provide an 8-byte RBA when the data set is defined for EA. Special provisions are allowed for certain types of requests (for example, GET SEQ,ADR and GET ADR,DIR,LRD).

EA design is intended to make applications, such as backup/restore, transparent to the function. That is, the extended field (8-bytes) XRBA is not needed unless it is a positioning request. For a backup when the data set is to be read sequentially from the beginning, which requires no positioning, the extended field is not required. Also, RLS processing does not support any use of RBA or XRBA access to an EA data set.

DB2 data warehousing projects that require table spaces larger than 4 GB can use the EA support for linear data sets. To do so, assign a data class with the extended addressability attribute to the data set when it is defined. The data class must have the attributes specified for it as shown in Figure 2-6.

```
Data Set Name Type = Extended
IF Extended = Required
Extended Addressability = Yes
Recorg = LS
```

Figure 2-6 Extended Addressability Data class attributes

Make sure that your data class ACS routine for DB2 allows the use of the data class created.

The message `IDC3351I` *`** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS 116`* is displayed when you try to go beyond 4 GB in a VSAM data set without EA.

2.5 Catalog Search Interface

The Catalog Search Interface (CSI) is shipped as a component of DFSMS. It has several advantages over other methods of obtaining catalog information. The following section describes the CSI setup, including the following topics:

- Some CSI highlights
- Supplied sample program

2.5.1 CSI highlights

The CSI is a general-use programming interface for obtaining information from ICF catalogs. It provides great flexibility in specifying the selection criteria for the data that is to be returned. The CSI can be started by assembler programs, high-level language programs, and REXX execs. For a complete description of the interface, see the appendix in *Managing Catalogs*, SC26-4914.

Much of the information you can obtain from the CSI can also be obtained by using an IDCAMS LISTCAT command. However, there are some advantages when using CSI that you might want to consider when you are accessing catalog information:

- Using a Generic Filter Key:

  When requesting information from the CSI for specific catalog entries, you can specify a generic filter key. This key can contain the following symbols to filter the entry names:

*	A single asterisk represents one or more characters within a qualifier
**	A double asterisk represents zero or more qualifiers
%	A percent sign represents one alphanumeric or national character
%%...	Up to eight characters can be specified in one qualifier

- Using selection criteria fields

  When requesting information from the CSI for specific catalog fields, you can specify a list of field names. If you are only interested in the volume and the file sequence number for specific data sets, specify the catalog field names VOLSER and FILESEQ in the field name list when calling the CSI. Obtaining this information from IDCAMS requires you to use the IDCAMS LISTCAT ALL command and scan the output to retrieve the wanted information.

- Performance benefits

  Using the CSI generally results in better performance compared to using IDCAMS LISTCAT, which does a catalog call for each entry processed. The flexibility in requesting only the information that you are interested in using the CSI results in more performance improvements by eliminating the retrieval of unneeded information.

- CSI programming considerations

  The CSI is distributed as load module IGGCSI00 in SYS1.LINKLIB. It is reentrant and reusable. It can be started in 24-bit or 31-bit addressing mode, in any PSW key, and in either problem or supervisor state.

  CSI requires three parameters to process your request:

  – A 4-byte *reason area that is used* to return error or status information
  – A variable length *selection criteria list (for input)*
  – A *work area* that is used to return the requested catalog data

Supplied sample programs

IBM provides three sample assembler programs and one REXX exec in SYS1.SAMPLIB. These programs and exec have the following functions:

- IGGCSILK produces output similar to that of an IDCAMS LISTCAT CAT(catname) command.
- IGGCSIVG identifies unused space at the end of VSAM data sets defined in a catalog. This space is calculated as the difference of the high-allocated and the high-used relative byte address (HARBA-HURBA)

- IGGCSIVS produces a list of data set names that are defined in a catalog on a specific volume. Such a list might be helpful in a volume recovery situation.
- IGGCSIRX is a REXX exec that produces a list of data set names that match a generic filter key. When you call it from a TSO/E session, it prompts you for the filter key, and returns matching data set names, their type, and volume definition.

  For a sample REXX that uses CSI, see “SMFLSR sample program” on page 395.

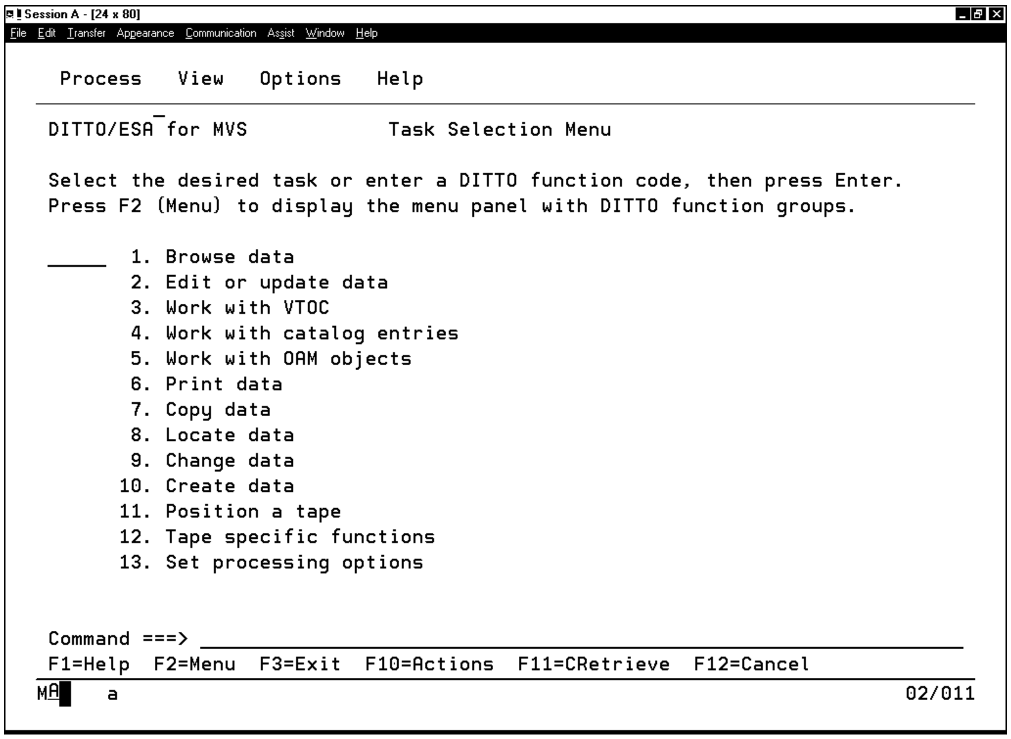
2.5.2 DITTO/ESA

DITTO is a powerful utility product that you can use to browse, edit, and delete VSAM records. You can start DITTO in full-screen mode from a TSO terminal. Check with your system programmer how to start DITTO because the procedures can vary with the installation.

In full-screen mode, you can use menus, online help, and interactive browse and update functions. Full-screen mode is often the most convenient way to run DITTO, especially if you are a new DITTO user.

You can also run DITTO in line, command, or batch modes. For more information about using DITTO, see *DITTO/ESA V1R3 User’s Guide,* SH19-8221.

Figure 2-7 shows the DITTO Task Selection menu. Select option 1 and then option 3 to browse a VSAM data set.

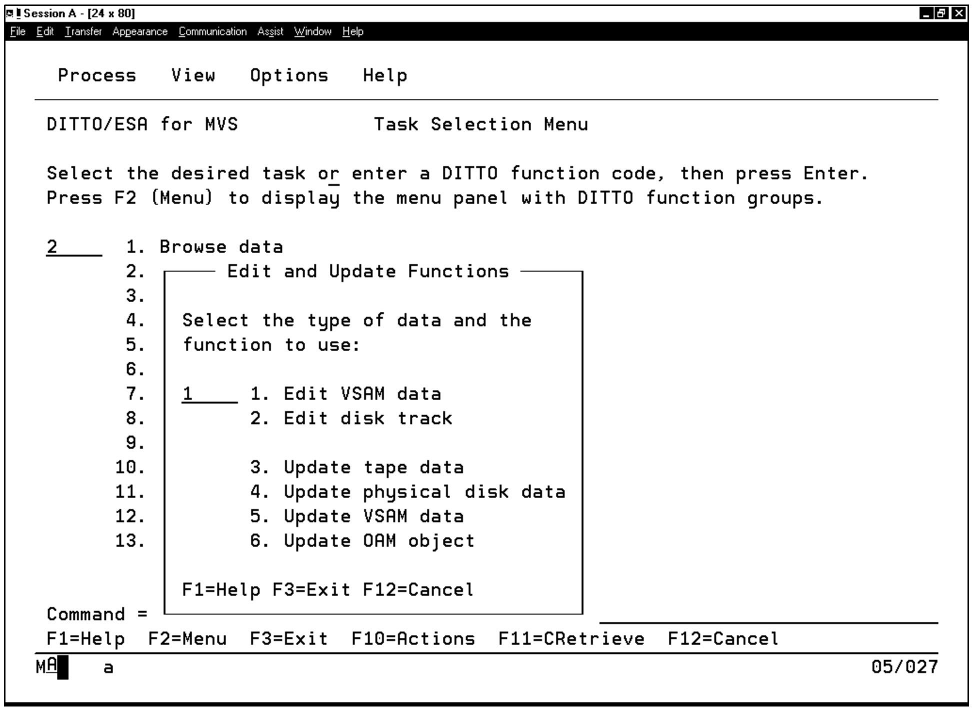

Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help

```
  Process   View   Options   Help
 ----------------------------------------------------------------------------
 DITTO/ESA for MVS            Task Selection Menu

 Select the desired task or enter a DITTO function code, then press Enter.
 Press F2 (Menu) to display the menu panel with DITTO function groups.

 _____    1. Browse data
          2. Edit or update data
          3. Work with VTOC
          4. Work with catalog entries
          5. Work with OAM objects
          6. Print data
          7. Copy data
          8. Locate data
          9. Change data
         10. Create data
         11. Position a tape
         12. Tape specific functions
         13. Set processing options


 Command ===> _________________________________________________________________
 F1=Help  F2=Menu  F3=Exit  F10=Actions  F11=CRetrieve  F12=Cancel
MA     a                                                              02/011
```

Figure 2-7 DITTO selection panel

Select option 2 on the Task Selection menu and then option 1 to edit a VSAM data set.

Figure 2-8 shows the DITTO edit menu.

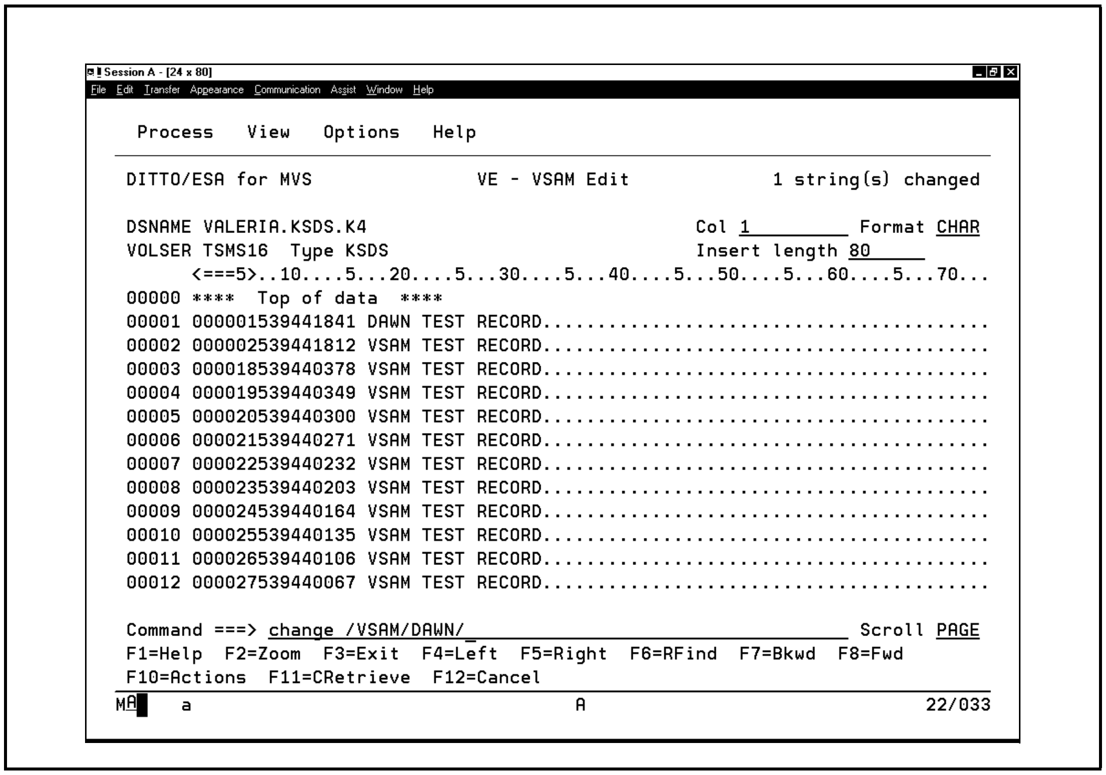


Session A - [24 x 80]
File Edit Transfer Appearance Communication Assist Window Help

```
  Process   View   Options   Help
 ------------------------------------------------------------------------------
 DITTO/ESA for MVS            Task Selection Menu

 Select the desired task or enter a DITTO function code, then press Enter.
 Press F2 (Menu) to display the menu panel with DITTO function groups.

 2____   1. Browse data
         2. ┌──── Edit and Update Functions ────┐
         3. │                                    │
         4. │ Select the type of data and the    │
         5. │ function to use:                   │
         6. │                                    │
         7. │ 1____  1. Edit VSAM data           │
         8. │        2. Edit disk track          │
         9. │                                    │
        10. │        3. Update tape data         │
        11. │        4. Update physical disk data│
        12. │        5. Update VSAM data         │
        13. │        6. Update OAM object        │
            │                                    │
            │ F1=Help F3=Exit F12=Cancel         │
 Command =  └────────────────────────────────────┘ _______________________
 F1=Help  F2=Menu  F3=Exit  F10=Actions  F11=CRetrieve  F12=Cancel
MA     a                                                              05/027
```

Figure 2-8 DITTO edit panel

Figure 2-9 shows an example of editing a record by using DITTO.



```
Session A - [24 x 80]
File  Edit  Transfer  Appearance  Communication  Assist  Window  Help

   Process   View   Options   Help
 ------------------------------------------------------------------------------
 DITTO/ESA for MVS               VE - VSAM Edit            1 string(s) changed

 DSNAME VALERIA.KSDS.K4                              Col 1_________ Format CHAR
 VOLSER TSMS16  Type KSDS                            Insert length 80_____
      <===5>..10....5...20....5...30....5...40....5...50....5...60....5...70...
 00000 ****  Top of data  ****
 00001 000001539441841 DAWN TEST RECORD........................................
 00002 000002539441812 VSAM TEST RECORD........................................
 00003 000018539440378 VSAM TEST RECORD........................................
 00004 000019539440349 VSAM TEST RECORD........................................
 00005 000020539440300 VSAM TEST RECORD........................................
 00006 000021539440271 VSAM TEST RECORD........................................
 00007 000022539440232 VSAM TEST RECORD........................................
 00008 000023539440203 VSAM TEST RECORD........................................
 00009 000024539440164 VSAM TEST RECORD........................................
 00010 000025539440135 VSAM TEST RECORD........................................
 00011 000026539440106 VSAM TEST RECORD........................................
 00012 000027539440067 VSAM TEST RECORD........................................

 Command ===> change /VSAM/DAWN/_________________________________ Scroll PAGE
 F1=Help  F2=Zoom  F3=Exit  F4=Left  F5=Right  F6=RFind  F7=Bkwd  F8=Fwd
 F10=Actions  F11=CRetrieve  F12=Cancel
MA     a                                A                                22/033
```

Figure 2-9 VSAM edit panel

2.6 Major sources of VSAM processing options

The VSAM cluster processing options and characteristics come from different sources. For example, JCL, SMS data classes, and application parameters are in control blocks. The source of the processing options can be confusing because the same parameter can be specified in more than one source. In this case, there is an order of precedence that, if not understood, can give logically unexpected results. Table 2-3 lists the processing options and the characteristics and their sources, indicating the precedence by the highest number. The underlined value in MACRF indicates the default. The *Type* indicates whether the parameter is related to a data set characteristic (C) or processing option (P).

Table 2-3 VSAM data set parameters and processing options

Parameter	Type	ACB	DD	Define	SMB DC
ACCBIAS	P	-	2	-	1[a]
AVGREC (RECORDS)	C	-	2	2	1
BUFSP	P	2[b]	3[b]	1[c]	-
BUFND,BUFNI	P	2	3	1	-
BWO	P	-	-	2	1
CATALOG	C	-	-	1	-
COMPACTION	C	-	-	-	1

Parameter	Type	ACB	DD	Define	SMB DC
CISZ	C			2	1
DATACLAS (DATACLASS) [d]	C	-	1	1	-
DDNAME	P	1	-	-	-
DYNAMIC VOLUME COUNT	P	-	-	-	1
ERASE/NOERASE	P	-	-	1	-
EXCEPTIONEXIT	P	-	-	1	-
EXLST	P	1	2[e]	-	-
EXPDT	C	-	2	2[f]	1,3[g]
EXTENDED ADDRESSABILITY	C	-	-	-	1
EXTENDED FORMAT	C	-	-	-	1
FREESPACE	C	-	-	2	1
FRLOG	P	-	3	2	1
LIKE	C	-	1	1[h]	-
KEYLEN	C	-	2	2	1
KEYOFF	C	-	2	2	1
LOG	P	-	-	2	1
LOGSTREAMID	P	-	2	2[i]	1
LRECL (RECORDSIZE) [j]	C	-	2	2	1
MGMTCLAS (MANAGEMENTCLASS) [d k]	P	-	1	1	-
MACRF=[NSR I LSR I GSR I RLS]	P	1	[l]	-	-
MACRF=[ADR,CNV,KEY]	P	1	-	-	-
MACRF=[CFX I NFX]	P	1	-	-	-
MACRF=[DIR][,SEQ][,SKP]	P	1	-	-	-
MACRF=[DFR I NDF]	P	1	-	-	-
MACRF=[DDN I DSN]	P	1	-	-	-
MACRF=[IN,OUT]	P	1	-	-	-
MACRF=[ICI I NCI]	P	1	-	-	-
MACRF=[LEW I NLW]	P	1	-	-	-
MACRF=[NIS I SIS]	P	1	-	-	-
MACRF=[NRM I AIX]	P	1	-	-	-
MACRF=[NRS I RST]	P	1	-	-	-
MACRF=[NUB I UBF]	P	1	-	-	-
OWNER	C	-	-	1	-
RECATALOG/NORECATALOG	C	-	-	1	-

Parameter	Type	ACB	DD	Define	SMB DC
RECORG	C	-	1	1	-
RETPD	C	-	2	2[m]	1,3[n]
REUSE/NOREUSE	P	-	-	2	1
RLS CF Cache Value	P	-	-	-	1
RLSREAD={NR |CR NORD}]	P	1	[l]	-	-
RMODE31	P	1	2	-	-
SHAREOPTIONS	C	-	-	2	1
SPACE (CYL, KB, MB, REC, TRK)	P	-	2	2	1
SPACE CONSTRAINT RELIEF	P	-	-	-	1
SPANNED/NONSPANNED	C	-	-	2	1
SPEED/RECOVERY	P	-	-	2	1
STORCLAS (STORAGECLASS) [d]	P	-	1	1	-
STRNO	P	1	2	-	-
VOLUME	P	-	2[o]	2	1
WRITECHECK	P	-	-	1	-

a. Corresponds to Record Access Bias in Data Class
b. Specifies the maximum space for buffers
c. Specifies the minimum space for buffers
d. Can be overridden by the ACS routines
e. Through the AMP, SYNAD parameter
f. TO(*x*) FOR(*x*) of DEFINE command
g. When specified in MANAGEMENT CLASS, cannot be overridden
h. Correspond to MODEL in DEFINE command
i. LGSTREAM JCL parameter
j. LRECL does not apply to LINEAR data sets
k. Overrides Data Class parameters EXPDT RETPD
l. RLS=(NRI or CR) when in ACB RLSREAD=NORD or does not specify RLS
m. TO(*x*) FOR(*x*) of DEFINE command
n. When specified in MANAGEMENT CLASS, cannot be overridden
o. Only when Storage Class with Guaranteed Space= yes

2.6.1 Access method Control Block (ACB)

The ACB is a control block within the application program (written in any language). The ACB is to VSAM as the DCB is to other access methods. The ACB contains logical information about the VSAM data set in its fields. The ACB information is used by Open, Close, and other VSAM routines. The way that the ACB is created depends on the source language of the application program:

- In Assembly through the ACB macro at assemble time or GENCB at execution time
- In Cobol through the File Description statement
- In PL/I through a DCL statement.

The ACB fields can have these sources:

- From keywords in macro ACB at compile or assembler time
- At Open time, with ACB parameters that come from AMP parameter, in the DD statement
- From keywords in macro GENCB at execution time by the application program.

The following is a list of the major ACB fields. The ones that start with an asterisk indicate that they can also be entered in the JCL DD statement:

- DDName, which is the link with the DD statement. The DD statement, besides its optional complementary ACB fields, describes the physical characteristics of the data set. These characteristics include VOLSER which is where the data set is located, its device type, physical block size, and the space to be allocated.
- The size (or the maximum size) of the I/O buffer virtual storage space and the number of I/O buffers to process data and index control intervals.
- The chosen buffering technique, such as NSR, LSR, GSR, and RLS.
- The defer write option, which delays the buffer destage (move to the storage) of updated records.
- The processing options that you plan to use:
  - Search type arguments: Keyed, RBA, RRN
  - Access type: Sequential, direct, or skip sequential access, or a combination of them. This field is just an intention. It might be used by VSAM SMB for picking up the best buffer pool algorithm.
  - Action: Retrieval, storage, or update (including deletion), or a combination. It is used for requesting locks in an intra address space sharing situation.
  - If the resource pool is shared between data sets (NSR or LSR).
- The address of an exit list for your exit routines. Use the EXLST macro to construct the list.
- What to do when lock contention arises.
- Read integrity options for RLS GETs.
- If you are processing concurrent requests, and, if yes, which is the number of such requests (STRNO).
- The address and length of an area for error messages from VSAM,
- RMODE of the buffers and VSAM control block.
- Address of the VSAM GET/PUT routine. This field is filled by the OPEN macro function.

2.6.2 DD statement keywords

For more information about all the generic DD Statement keywords that apply to VSAM, see *MVS JCL Reference*, SA22-7597. This section covers the VSAM-specific parameters not present in the ACB:

- Options for controlling system-managed buffering (SMB) named ACCBIAS in the AMP keyword
- Logstream option for RLS and Transactional VSAM. For more information about this product, see Chapter 6, “DFSMStvs” on page 311.

2.6.3 Catalog BCS and VVDS entries

The catalog entry for a VSAM cluster or a component has the attributes usually needed to create this VSAM entity. Those attributes are entered in the catalog through the IDCAMS DEFINE/ALLOCATE/ALTER commands:

- Primary and secondary space information
- CI size
- Free space percentages
- Logical record size
- Cluster organization data set (KSDS, RRDS,ESDS, LDS, VRRDS)
- Key length and key offset
- Name of SMS constructs such as DC, SC, MC
- Retention period
- Share options: Cross region and cross system
- Spanned records
- Backup-while-open (BWO) options
- The minimum buffer space size
- Expiration date
- Initial load options (SPEED and RECOVERY)

2.6.4 SMS constructs

The following SMS constructs affect VSAM attributes.

Data Class construct

The data class construct has many of the options declared at IDCAMS DEFINE command and the DD statement. The practical difference is that, with a data class, you do not need to define all the IDCAMS keywords. You just point to the data class name. Also, these options are centralized and controlled through the SMS storage administrator. The following options in data class are not present in the IDCAMS DEFINE, and are not stored in the catalog:

- Whether to use the primary size in a secondary volume
- Maximum number of volumes
- Extended format option
- Extended addressability option
- Options for controlling SMB called RECORD_ACCESS_BIAS
- Space constraint relief
- Data compression
- RLS use of the CF cache structure

Storage Class

The storage class SMS construct has these parameters to control:

- Performance:
  - How the I/O operation uses the controller cache
  - Are the CIs stripped?
  - CF structure name to be used as a cache by VSAM RLS
- Availability:
  - Ensure the VSAM cluster a controller with RAID implementation.
  - Ensure the VSAM cluster a controller with T0 copy and remote copy capabilities.

All modern controllers support RAID, T0 Copy, and remote copy capabilities.

Management Class

The management class SMS construct has parameters to control VSAM clusters expiration, partial release, GDG, backups, and migration.

2.7 Media Manager, Open, Close, EOV in VSAM

This section addresses the Media Manager, Open, Close, and EOV functions in VSAM.

Allocation function

Before you can open the data set, it must be allocated. Data set allocation is done by a z/OS component named Allocation Routine started by the SVC 99 instruction.

First, differentiate between data set allocation and data set creation. The expression “allocate a data set” does not imply creating a data set. You can allocate an already existing data set. Allocating a data set to a task means to establish a logical connection between the task (through a program that runs underneath) with the data set. However, for a DISP=NEW data set, allocation includes creating the data set by generating a DSCB entry in the VTOC.

This connection is represented by an entry in the Task Input/Output Table (TIOT**), which is** another MVS control block. Each task has a TIOT. A TIOT occupied entry describes a data set that is allocated to such a task. Each entry connects the DNAME with the UCB (another MVS control block) representing the 3390 device that contains the data set. DDNAME is a name declared within the application program at the ACB control block. The purpose of this name is to link the logical attributes of the data set as described internally at the program in the ACB, with the physical attributes described at the DD card that has the same DDNAME.

OPEN macro

Before an application program can access a data set, it must first issue the OPEN macro to open the data set (after allocation) for processing. The OPEN is issued against a user ACB. Opening a data set causes VSAM open function to perform these tasks:

- Verify that the data set matches the description that is specified in the ACB or GENCB macro. For example, MACRF=KEY implies that the data set is KSDS.
- Construct the internal control blocks and buffer pools that VSAM needs to process your requests for access to the data.
- Load the correct access method routine (based on the ACB information) placing its address in the ACB for the next GET or PUT.
- Check your program task security authorization for reading and for writing through RACROUTE service (RACF).
- Fill out the ACB with options declared by the JCL DD statement (AMP parameters)
- Construct OPEN control blocks (such as Data Extent Block (DEB)), which are allocated at private area protected area (subpool 230, storage key 5). The exception is for GSR and ICI that are getmained at CSA and SQA.
- Initialize a protected field for the high allocated CI, for being later updated by the EOV routine when you extend the data set.
- Perform an implicit VERIFY. If the VERIFY is not successful, VSAM OPEN passes a return code and reason code. For more information about the VERIFY function, see “VERIFY” on page 346.

Usually a failed Open does not imply in a task abend. Then, it is key that the application tests afterward the Open return code to ensure that the data set Open function was succeeded.

Media manager

Media Manager is the VSAM I/O driver code. It runs in the supervisor state that is started by VSAM Record Management. It stands logically between the access method itself and the Input/Output Supervisor (IOS).

Media Manager runs all these functions:

- Writes a Channel Programs with virtual addresses (this was originally done by VSAM).
- Runs the Page-Fix and Page-Free functions toward the pages that contain the channel program and the I/O buffers. During the execution of an I/O operation, a page steal cannot happen.
- Full support for modern CCWs such as Read Track Data, Write Track Data, and Write Full Track.
- Translates virtual addresses to and from real addresses in the channel program. During this translation process Media Manager might convert the CCWs channel program that is written by VSAM into zHPF TCW and TCCB control blocks. This conversion is only done if you set ZHPF=YES in the Parmlib member IECIOSxx or zHPF is activated through the SETIOS MVS command.
- Verifies that buffers are accessible in user PSW key.
- Validates that RBA is within the data set.
- Pass the control to IOS through the STARTIO macro, which expands in just a branch instruction. IOS is the z/OS component in charge of issuing the SSCH instruction to trigger SAP to run the I/O operation by the FICON channel.
- Provides for DCME statistics.
- Starts SMFIOCNT.

End-of-Volume (EOV) routine

EOV function is started by VSAM Record Management when a VSAM data set requires more secondary space. The lack of this additional space is perceived in these conditions:

- HURBA is equal to HARBA
- The RBA of next CI or CA greater than HARBA during initial load
- No extent in the current volume contains a specific searched RBA

EOV then acquires new extents by interfacing with DADSM, a z/OS component that takes care of the VTOC. It updates the VSAM control block structure for the data set with the new extent information, and updates the critical control block data in common storage and in the catalog. This process ensures that the new space is accessible by all address spaces that use this VSAM data set.

CLOSE macro

The CLOSE macro disconnects your program task from the data set. VSAM runs the following processes during CLOSE:

- Writes any unwritten data or index records whose contents have changed
- Writes SMF record 64
- Updates the catalog entry for the data set.
- Updates the data set's HURBA

- Restores control blocks to the status they had before the data set was opened
- Releases virtual storage that was obtained during OPEN processing for more VSAM control blocks and VSAM routines
- For extended format KSDS, releases all space between HURBA and HARBA if partial release was specified at OPEN time

If an abend happens during CLOSE, the HURBA that is updated during CLOSE processing might be incorrect because the catalog was not updated.

You can issue a CLOSE TYPE=T, also called a temporary CLOSE. This command causes VSAM to complete any outstanding I/O operation, update the catalog, and write any required SMF records. Processing can then continue without issuing an OPEN macro.

2.8 VSAM and 64 bits in real storage

VSAM supports 64 bits of *real* storage. This capability means that virtual pages that contain I/O buffers can be in central storage frames with a real address above 2 GB. Do not confuse this function with the use of virtual storage above the bar (2 G addresses) for I/O buffers, as implemented in VSAM RLS.

Some VSAM functions were modified to allow the use of these real storage (or central storage) addresses:

- Media Manager: Performs I/Os in central storage above 2 GB to all types of VSAM data sets, except when you use ICI processing. For ICI processing, it supports 64-bit real addresses frames that are used to back 31-bit virtual addresses pages.
- OPEN routine: Obtains buffers in 64-bit *real* storage in all cases.

2.9 Special considerations for COBOL users and SMB

This section provides SMB considerations for COBOL users.

Keep in mind is that COBOL is written for NSR processing. If anything is done to switch the processing to LSR (whether OEM, SMB, or BLSR), the program or COBOL might not be able to handle it. However, SMB can be disabled for a particular program by coding an override on the DD card and allowing other files that access the data set continue to use SMB.

2.9.1 Positioning error

The most common problem is a positioning error. With NSR processing, implicit positioning by VSAM to the first record in the file is done on the first sequential GET. Many programs (especially older ones) take advantage of this process and do not explicitly position to the first record of the file. With LSR, no implicit positioning is done, so if the program does not position, the same GET that worked with NSR is returned with an rc58. This code translates into a COBOL SK30. Receiving this error should be minimal because SMB will decide against DO processing on a file that is opened with sequential.

2.9.2 Waits and hangs

Another common problem is waits/hangs. This problem is also because of the inherent differences between NSR and LSR. With NSR, you can have multiple requests in the pool that are accessing the same data CI. This sharing is possible because a copy of the CI is given to each request. In LSR, you can have multiple requests that are reading the same CI. They all share a buffer rather than getting their own copy. However, a request for update of that CI must wait until all the readers are finished with it. This process can cause hangs.

There are a number of possible solutions for this problem. Which one you should use depends on what the program is doing and in what order. If, for example, the program has two ACBs, it might open the sequential ACB first, and then DO would not be selected.

2.9.3 ISAM data set programs

Another possible problem might arise with old programs that were written for ISAM data sets. If you have been using the VSAM/ISAM interface to continue using these programs, you must continue to use NSR. The effect should be minimal because the ACB probably has sequential specified.

Some SMB users, not just COBOL, are finding that they must increase their region size when they are running SMB. The extra storage is required to optimize the buffering. This need is especially true on large files or programs that open many SMB VSAM files. In general, SMB is efficient and you should see a good deal of performance benefits from using SMB. Keep in mind that some applications might need minor JCL or coding changes to use SMB.

3

Guidelines for ICF catalog and VSAM

The ICF catalog is a VSAM data set with a specific function. It has some specifics needs that are addressed in this chapter along with guidelines that are based on customer experiences.

This chapter includes the following sections:

3.1 ICF catalog basics

A catalog is a data set that contains information about other data sets. It provides you with the ability to locate a data set by name, without knowing where the data set is. A catalog consists of two separate kinds of data sets:

- Basic catalog structure (BCS)
- VSAM volume data set (VVDS)

3.1.1 Basic catalog structure (BCS)

BCS is a VSAM KSDS. The key is the data set name, used to store and retrieve data set information. It can be considered *the* catalog, and contains volume, ownership, and association information for the data sets. For VSAM data sets, it also contains security information.

BCS catalogs can be SMS-managed. SMS-managed catalogs can be extended format, which provides better reliability and availability to the I/O operations.

Catalogs can be shared for all systems in the sysplex. BCS are functionally divided in two kinds of catalog: Master catalog and user catalog. There is no structural difference between them. What makes a master catalog different is how it is used, and what data sets are cataloged in it.

The BCS can have as many as 123 extents on one volume. One volume can have multiple catalogs on it. All the necessary control information is recorded in the VVDS on that volume.

Master catalog

Each system has one active master catalog. It must contain entries for system data sets required for IPL. It must also contain all the user catalogs that are used on the system and the aliases that point to them. A master catalog from one system image can be defined as a user catalog in another systems in the same sysplex.

Example 3-1 shows the execution of a command that connects a user catalog to a master of another system.

Example 3-1 Connecting a user catalog to a master of another system

```
IMPORT CONNECT OBJECTS((CATALOG.MHLRES2 -
                  DEVT(3390) VOLUME(TST057))) -
         CATALOG(MCAT.ZOSR04.VZ04CAT)
IDC0603I  CONNECT FOR USER CATALOG  CATALOG.MHLRES2 SUCCESSFUL
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

User catalogs

User catalogs only have information about others data sets, unless it is a master catalog from another system image. You create a user catalog with DEFINE USERCATALOG IDCAMS command. Unless you specify otherwise, the user catalog that is created is connected to the master where the command is run.

Example 3-2 shows a DEFINE USERCATALOG command.

Example 3-2 Defining a user catalog

```
//DEFCAT    EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD  *
  DEFINE USERCATALOG  -
        (NAME(CATALOG.MHLRES2) -
         CYLINDER(50 30)  -
         VOLUME(TST057)  -
          STRNO(3)) -
     DATA(CISZ(4096) BUFND(4)) -
     INDEX(CISZ(4096))
```

3.1.2 VSAM volume data set (VVDS)

VVDS is a VSAM ESDS data set and can be considered an extension of the VTOC. There is one VVDS on every volume that contains VSAM data set or any SMS-managed data set. The catalog information that requires the most frequent updates is physically in the VVDS on the same volume as the data set, allowing faster access. For VSAM data sets, it contains the data set characteristics, extent information, and the volume-related information. For SMS-managed non-VSAM, it contains data set characteristics and volume-related information. Temporary VSAM data sets that are SMS-managed also have VVDS entries, although they do not have BCS entries.

VVDS has two types of records:

- VSAM volume records (VVRs) contain information about VSAM data sets. Because a BCS is a VSAM data set, it also has a VVR in the VVDS.
- Non-VSAM volume records (NVRs) contain information about SMS-managed non-VSAM data sets. If a non-VSAM data set spans volumes, its NVR is in the VVDS of the data set's first volume.

SMS-managed volumes have larger VVDS because they can have VVR and NVR records, whereas non-SMS volumes have only VVR records.

Every catalog consists of one BCS and one or more VVDSs. A BCS does not “own” a VVDS: More than one BCS can have entries for a single VVDS. Every VVDS that is connected to a BCS has an entry in the BCS.

VVDS is created automatically the first time a VSAM data set or a non-VSAM SMS-managed data set is allocated in that volume. You do not need to allocate a VVDS unless you want to define a VVDS with a space allocation different from the default. In such a case, you must allocate the VVDS first.

The default space allocation is TRACKS(10 10). Example 3-3 shows how you can use the F CATALOG,VVDSSPACE command to change the default. The amount is in track units. The values that are specified with this command are not preserved across an IPL. You can see the default values by using the F CATALOG,REPORT command.

Example 3-3 Changing the default space allocation for VVDS

```
F CATALOG,VVDSSPACE(15,15)
```

3.2 Factors affecting catalog performance

As the number of systems that share resources grow in the parallel sysplex environment, contention for resources can increase, resulting in slower response times or reduced throughput. One of the resources that might increase contention is a catalog. Elongation of catalog requests cannot only affect the performance of work within the system performing the request, but can affect other systems in the sysplex as well. This section addresses the factors that can affect catalog performance.

3.2.1 Catalog configuration

Catalogs have an important role in availability and overall system performance. Having adequate catalog configuration is key for achieving this goal. During catalog configuration, analyze and plan for catalog protection, size, attributes, recovery procedures, volumes they will be in, documentation, and monitoring. When you create the catalog configuration, keep in mind that it must allow easy recovery of damaged catalogs with the least system disruption time. There are several options that you can choose to improve catalog performance without affecting the recoverability of a catalog.

If SMS is used, decide which catalogs are SMS-managed, and create data classes, management classes, and storage classes to be used by BCS and VVDS data sets. Simplify the definition of BCSs and VVDSs by creating data classes with the appropriate default size, performance attributes, and characteristics. For more information about sizing the catalog, see *z/OS DFSMS Managing Catalogs,* SC26-7409.

Plan the aliases and the user catalogs so that the master catalog contains only entries for catalogs, catalog aliases, and system data sets. This configuration improves master catalog performance and speed during IPL. During system initialization, the entire master catalog is read into main storage. If the master catalog is properly used, it is rarely updated and its performance is not appreciably affected by I/O requirements. Do not put entries for application or user data sets in a master catalog. Place these entries in user catalogs that are pointed to by the appropriate aliases in the master catalog.

Guideline: Define aliases to ensure that the master catalog in every system holds only connector records, records to user catalogs, systems data sets, and aliases that point to user catalogs.

3.2.2 BCS definition

You can enhance catalog performance by carefully selecting VSAM attributes when you define the catalog. Some catalog characteristics cannot be altered after the catalog is defined, so pay careful attention when you select catalog attributes.

You can estimate some catalog characteristics, such as strings number (STRNO) and index buffers (BUFNI), at creation time. You can adjust them by using IDCAMS ALTER command.

You define a catalog by using the IDCAMS DEFINE USERCATALOG command. Use this command even when you are defining a master catalog. Even if you code, the CATALOG parameter is not used and the entry goes to master catalog of the system where the command is run.

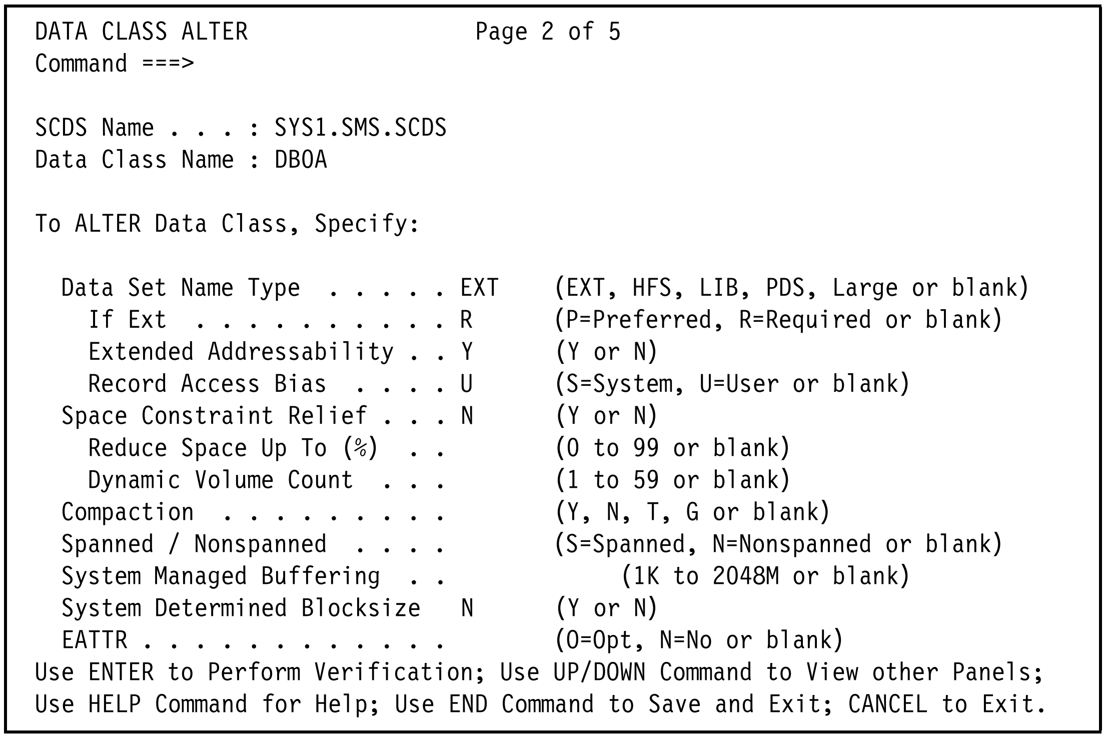
Some restrictions apply to catalogs:

- It cannot span volumes, so it cannot be defined as a striped data set
- Since z/OS V1R12, it can be extended format
- It can have up to 123 extents

A BCS is limited to 4 GB unless you define it as SMS-managed, extended format BCS with extended addressability. In that case, its size limit become the control interval size multiplied by 4 GB. For example, a control interval size of 4-KB yields a maximum data set size of 16 TB. No increase in processing time is expected for extended format data sets that grow beyond 4 GB. Extended format provides better reliability and availability to the I/O operations.

To share a catalog by using extended addressability, all systems in sysplex must be in R12 or higher. APAR OA30000 must be on down level systems to keep them from accessing catalogs by using extended addressability. Otherwise, the catalogs will break.

Create a BCS with extended format and extended addressability by setting a DATACLASS with DSNTYPE=EXT parameter and subparameter R (meaning required). Use R to ensure that the BCS is extended. The Extended Addressability value must be set to Y (Yes). You create or alter an SMS data class by using ISMF tool, as shown in Figure 3-1.



```
DATA CLASS ALTER                  Page 2 of 5
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : DBOA

To ALTER Data Class, Specify:

  Data Set Name Type  . . . . . EXT    (EXT, HFS, LIB, PDS, Large or blank)
    If Ext  . . . . . . . . . . R      (P=Preferred, R=Required or blank)
    Extended Addressability . . Y      (Y or N)
    Record Access Bias  . . . . U      (S=System, U=User or blank)
  Space Constraint Relief . . . N      (Y or N)
    Reduce Space Up To (%)  . .        (0 to 99 or blank)
    Dynamic Volume Count  . . .        (1 to 59 or blank)
  Compaction  . . . . . . . . .        (Y, N, T, G or blank)
  Spanned / Nonspanned  . . . .        (S=Spanned, N=Nonspanned or blank)
  System Managed Buffering  . .             (1K to 2048M or blank)
  System Determined Blocksize   N      (Y or N)
  EATTR . . . . . . . . . . . .        (O=Opt, N=No or blank)
Use ENTER to Perform Verification; Use UP/DOWN Command to View other Panels;
Use HELP Command for Help; Use END Command to Save and Exit; CANCEL to Exit.
```

Figure 3-1 Data Class with extended format and extended addressability

Since z/OS DFSMS V1R12, BCS and VVDS can explore EAV capability. They can be above the 64k cylinders area (the cylinder managed area), which is also called Extended Addressing Space (EAS). The EATTR JCL keyword and IDCAMS DEFINE parameter are used to control a data set being created in EAS. You can set EATTR attribute on the ISMF Data Class Define panel.

Figure 3-2 shows an example of defining a user catalog. EATTR=OPT means that the data set can have extended attributes (format 8 and format 9 DSCBs) and can be in EAS. When you define a catalog by using EAV, be sure that all systems that share the catalog are able to share an online EAV. The minimum level that is required is z/OS V1R10.

```
DEFINE USERCATALOG  -
      (NAME(CATALOG.MHLRES2) -
       CYLINDER(42 21)  -
       VOLUME(TST057)  -
       EATTR(OPT) -
        STRNO(3)) -
   DATA(CISZ(4096) BUFND(4)) -
   INDEX(CISZ(4096))
```

Figure 3-2 Defining a BCS that can be at the cylinder managed area

For more information about EAV, see *z/OS V1R11 DFSMS Release Guide*, SG24-7768.

Guideline: All systems must be on z/OS R12 or higher for sharing catalogs that use extended addressability. Apply APAR OA30000 to prevent down-level systems from accessing them.

RECORDSIZE

Do not code RECORDSIZE. The record size information is ignored when a catalog is defined, and is always set to RECORDSIZE(4086 32400). The default maximum value should be used if large records might be generated. Large catalog records are generated for GDGs that have many GDSs. Master catalogs with user catalog connector records that contain many aliases also have large catalog records.

Control interval size

All considerations about VSAM CI size also apply to catalogs. The unit of transfer between DASD and storage is the CI. Large CI size favors sequential processing mode, whereas small CI size is better for direct processing. Most access to catalogs is direct. Catalogs records can span CIs. Selecting a CI size that is too small can cause another I/O operation to retrieve a record, which is affects performance.

For the data component, defining 4-KB CI size provides a compromise between minimizing data transfer time and reducing the occurrence of spanned records.

In the index component, each sequence set CI maps one CA data component. Each record maps one CI in the CA data component. If the index CI size is too small, there is no room for records to map all CIs. This limitation makes part of the CA unusable, wasting space. For the index component, use a minimum Index CI size of 3584 if you are using a 4-KB data component CISIZE.

Guidelines: For the data component, use 4-KB CISIZE. For the index component, do not use the default. Use at least 3584 CISIZE if you are using a 4-KB data component CISIZE.

Primary and secondary space allocation

Optimize the control area size by selecting the correct size and number of units when allocating a catalog. CA does not span cylinders, so its maximum size is one cylinder, the best size to minimize splits. Always allocate space in cylinder units. This configuration makes the CA size be one cylinder size.

Make secondary space greater than one cylinder to avoid going to allocation routines each time a split occurs. Also, making secondary space too small can reach 123 extents, which requires reorganizing the catalog.

Guidelines: Always use cylinder as allocation unit. Specify secondary space allocation higher than one cylinder.

STRNO

Strings are controls used to allow parallel access to data. Each string is related to one request. Requests to ICF catalogs are served in catalog address space (CAS). You can have from 2 to 255 concurrent read requests. Only one write request is allowed at a time. Defining STRNO=3 is a good start. You can adjust this number as needed.

You can monitor if there are requests that are waiting for strings with IBM RMF™. Monitoring these requests provides information about resources that periodically build up queues. Look for enqueues in the resource major name SYSZRPLW and minor name as the catalog name. An indication of enqueue contention is given by the number of jobs that are waiting to use the resource and average contention time. For more information about the steps to use RMF III monitor to detect enqueue problems, see “Catalog contention” on page 116. For more information about RMF Reports, see *z/OS V1R12.0 Resource Measurement Facility (RMF) Report Analysis* SC33-7991-18.

You use the F CATALOG,REPORT,CATSTATS(*catname*) console command to see I/O statistics and BUFNI, BUFND and STRNO values for a catalog that is allocated to CAS. If you omit (*catname*), information is displayed for all catalogs that are allocated to CAS, as shown in Figure 3-3 on page 88. You can also use the IDCAMS LISTCAT command, but you must add the CATALOG parameter, as shown in Example 3-4.

Example 3-4 Using LISTCAT command to obtain information about a catalog

```
LISTC ENT(CATALOG.MHLRES2) ALL CAT(CATALOG.MHLRES2)
```

You can change the STRNO attribute with the access method services ALTER command. This process takes effect after a close and subsequent reopen of the catalog. You can close the catalog by using the F CATALOG,CLOSE console command. The next request to the catalog causes the catalog to be reopened.

Guideline: Generally, start with STRNO=3. For catalogs with high activity, start with 6 or 7.

See Figure 3-3 for a catalog report that shows STRNO.

```
F CATALOG,REPORT,CATSTATS
IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
IEC359I CATALOG I/O STATS REPORT
*CAS*************************************************************
*     ADDS  UPDATES     GETS   GETUPD  DELETES  BUFNI  BUFND STRNO *
*                                                                  *
* CATALOG.MHLRES2                                                  *
*       1       2       13        1        0      5     11     3 *
* UCAT.ZOSR1A                                                      *
*       0       0        2        0        0     10     16     8 *
* UCAT.ZOSMF                                                       *
*       0       0        2        0        0     10     16     8 *
* UCAT.ZFSFR                                                       *
*       0       0        2        0        0      5     11     3 *
* UCAT.VTFMTAPE                                                    *
*       0       0        2        0        0     10     16     8 *
* UCAT.VSBOX11                                                     *
*       0       0        2        0        0     10     16     8 *
* UCAT.VSBOX02.TEST01                                              *
*       0       0        2        0        0      4      4     2 *
* UCAT.DB9D                                                        *
*       0       0      395        0        0     10     16     8 *
* UCAT.VSBOX01                                                     *
*       6       3    1,664        3        3     11     16     8 *
* MCAT.SANDBOX.Z1C.SBOX00                                          *
*      18      18   33,742       17        8     11     16     8 *
*CAS*************************************************************
IEC352I CATALOG ADDRESS SPACE MODIFY COMMAND COMPLETED
```

Figure 3-3 Obtaining I/O statistics, STRNO, BUFNI, BUFND for all catalogs that are allocated to CAS

BUFNI, BUFND, and BUFFERSPACE

Do not define BUFFERSPACE. It is the minimum buffer size in bytes, and applies to the whole cluster. Accept the default to avoid errors in miscalculations.

Specify the number of buffers for index and for data component by using BUFNI and BUFND. Most accesses are direct, so set the buffers number to favor direct access.

In direct access, data buffers and sequence set buffers are not shared among strings. Only one data buffer is used for each parallel request (string). Additional data buffers are used for CA split and spanned records. There is no gain from having more data buffers.

For index buffers, more buffers are used for index set and are shared among strings. Therefore, a better performance is achieved when all CIs of index set are in buffers and each string has one buffer per sequence set. You can calculate how many CIs are used by an index set only when the data set is loaded. Figure 3-4 on page 89[1] shows the LISTCAT command output and where to find the information you need for the calculation:

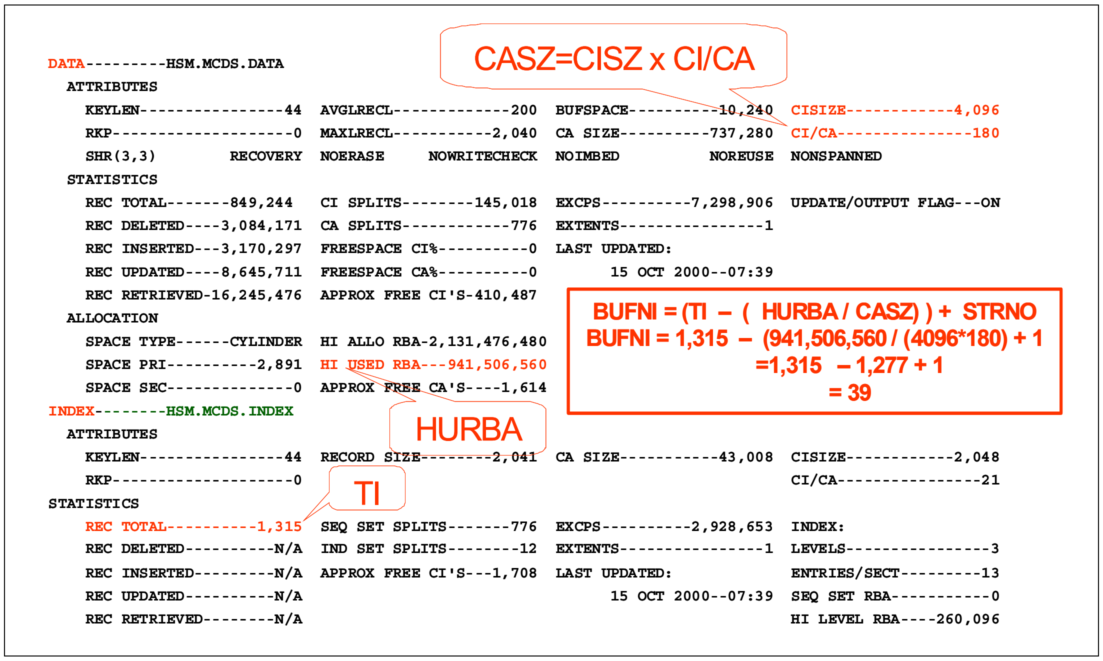
1. Each sequence set CI maps one data CA.
2. In the index set, each CI is occupied by only one index record.

[1] Courtesy Stephen Branch, IBM Corporation and Janet Sun, Rocket Software

3. HURBA points to the next available byte for inclusion.
4. Data CISIZE multiplied by data CI/CA value gives the data CA size.
5. Dividing the data HURBA value by the data CA size gives the number of CAs being used in the data component. If the result is not an integer, the last CA that is not totally used is rounded up. This is the number of CI in the sequence set (see step 1).
6. The number of index records minus the numbers of CIs used for sequence set gives the number of CIs used for the index level. For the optimal index buffers number, add one buffer for each string to hold the sequence set CI.

The formula is:

```
BUFNI= STRNO + TI - (HURBA / CASZ)
WHERE:
STRNO means 1 non-shared buffer for the sequence set for each string
The rest are shared index buffers used for index set
```



```
DATA---------HSM.MCDS.DATA
  ATTRIBUTES
    KEYLEN----------------44  AVGLRECL-------------200  BUFSPACE----------10,240  CISIZE------------4,096
    RKP--------------------0  MAXLRECL-----------2,040  CA SIZE---------737,280  CI/CA---------------180
    SHR(3,3)        RECOVERY  NOERASE     NOWRITECHECK  NOIMBED          NOREUSE  NONSPANNED
  STATISTICS
    REC TOTAL-------849,244   CI SPLITS--------145,018  EXCPS---------7,298,906  UPDATE/OUTPUT FLAG---ON
    REC DELETED----3,084,171  CA SPLITS------------776  EXTENTS----------------1
    REC INSERTED---3,170,297  FREESPACE CI%----------0  LAST UPDATED:
    REC UPDATED----8,645,711  FREESPACE CA%----------0        15 OCT 2000--07:39
    REC RETRIEVED-16,245,476  APPROX FREE CI'S-410,487
  ALLOCATION
    SPACE TYPE------CYLINDER  HI ALLO RBA-2,131,476,480
    SPACE PRI----------2,891  HI USED RBA---941,506,560
    SPACE SEC--------------0  APPROX FREE CA'S----1,614
INDEX--------HSM.MCDS.INDEX
  ATTRIBUTES
    KEYLEN----------------44  RECORD SIZE--------2,041  CA SIZE-----------43,008  CISIZE------------2,048
    RKP--------------------0                                                       CI/CA----------------21
STATISTICS
    REC TOTAL----------1,315  SEQ SET SPLITS-------776  EXCPS---------2,928,653  INDEX:
    REC DELETED----------N/A  IND SET SPLITS--------12  EXTENTS----------------1  LEVELS----------------3
    REC INSERTED---------N/A  APPROX FREE CI'S---1,708  LAST UPDATED:             ENTRIES/SECT---------13
    REC UPDATED----------N/A                                  15 OCT 2000--07:39  SEQ SET RBA-----------0
    REC RETRIEVED--------N/A                                                       HI LEVEL RBA----260,096
```

CASZ=CISZ x CI/CA
BUFNI = (TI – (HURBA / CASZ)) + STRNO
BUFNI = 1,315 – (941,506,560 / (4096*180) + 1
=1,315 – 1,277 + 1
= 39
HURBA
TI

Figure 3-4 Calculating index buffers for direct access, STRNO=1

Guidelines:

- Do not code BUFFERSPACE
- Use BUFND = STRNO + 1
- Use BUFNI= STRNO + TI - (HURBA / CASZ)

FREESPACE and SHAREOPTIONS

BCS has unevenly distributed record insertion activity. The VSAM FREESPACE option distributes space evenly and provides little help for BCS. Splits, both CI and CA, are the best technique available to handle this uneven distribution. Keep in mind that splits are not bad. Splits are how VSAM deals with lack of space for insertion between records, and they generate free space for new insertions. Allow splits to obtain free space where it is needed. Do not reorganize, unless the data set is reaching the 123 extents limit.

Guideline: Use FREESPACE(0 0).

3.2.3 VVDS definition

The VVDS can hold a maximum of 65535 control intervals (CIs). This maximum limits defining of primary and secondary allocation sizes to 5460 tracks or less, or 364 cylinders or less. This limitation is also in effect when you extend the VVDS. Thus, if extending the VVDS by the secondary allocation amount causes it to exceed the CI limit, the extend fails. If you know the average number of VVRs per CI in the VVDS, you can estimate the maximum number of data sets the VVDS can support on the volume. IDCAMS PRINT of the VVDS can help you estimate the average number of VVRs per CI. Note that track allocations can be rounded up to the next cylinder value. For example, a define request for 5455 tracks will likely allocate 5460 tracks, which are evenly divisible by 15 and are the next cylinder boundary.

With z/OS DFSMS V1R12, BCS and VVDS can use EAV capability and be allocated at cylinder managed area. For more information about EAV, see *z/OS V1R11 DFSMS Release Guide*, SG24-7768.

3.2.4 Caching catalogs

The simplest method of improving catalog performance is to use cache to maintain catalog records within main storage or data space. After a data set's catalog records are in cache, future references to the data set can be processed by using the information in the cache. this process eliminates I/Os to the BCS. For the master catalog, records read sequentially or by key are cached. Alias and catalog connector records are built into tables at IPL time. For user catalogs, only records that are read directly are cached. Although the entire CI is read into the buffer, only the record you asked for is cached.

Two kinds of cache are available exclusively for catalogs:

- In-storage Catalog (ISC) cache
- Catalog Data Space Cache (CDSC)

Selecting the wrong cache type for your catalog can negatively affect performance. A single catalog can be cached in just one cache. It cannot use both at the same time. Both types of cache are optional. In addition, both can be canceled and restarted without an IPL.The best way to choose which cache type to use is to understand how the caches are managed.

Cache management spends processor cycles. Using cache become advantageous when the processor effect of maintaining the cache becomes less than the cost of doing the I/O to the catalog. Generally, the breakeven point for caching is a 20% hit rate. Anything less than that, and the processor usage while managing the cache is more than just doing the I/O.

You can see which cache each catalog is using and its hit rate by issuing the F CATALOG,REPORT,CACHE operator command. Figure 3-5 on page 91 shows an output of this command:

- The SHR column shows the cache type being used (ISC or VLF for CDSC). The F CATALOG,CLOSE command does not clear the PURGE statistic but the other values are reset.
- The PURGE count displays the number of times a particular catalog cache has been purged. This statistic is reset at IPL. Catalog purges commonly occur for two reasons:
  - ISC-cached catalog purges occur when a change is made from a sharing system
  - CDSC, the number of changes within a system exceeds the maximum that can be recorded between two catalog access events

For an ISC cache, look for PURGE information. If the cache is being purged frequently for a catalog, it is not a good candidate for this type of cache. For both kinds, a low hit ratio (below 20%) is not a good number. Caching does not help.

```
F CATALOG,REPORT,CACHE
IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
IEC359I CATALOG CACHE REPORT
*CAS*****************************************************************
*   HIT% -RECORDS- -SEARCHES --FOUND-- -DELETES- -SHR UPD- --PURGE-- *
*                                                                    *
* UCAT.VTFMTAPE                                  (ISC)               *
*    72%       1       581       422        90         0         0  *
* UCAT.VTFM                                      (ISC)               *
*    99%       6   491,596   491,590         0         0         1  *
* UCAT.VTFM                                      (ISC)               *
*    99%       6   491,596   491,590         0         0         1  *
* UCAT.TATERS2                                   (ISC)               *
*    99%       7     4,575     4,568         0         0         1  *
* UCAT.DB9J                                      (ISC)               *
*    99%       7     2,688     2,681         0         0         2  *
* UCAT.DB9CMISC                                  (ISC)               *
*    99%       9     6,105     6,096         0         0         0  *
* UCAT.DB9CLOGS                                  (ISC)               *
*    66%       1         3         2         0         0         2  *
* MCAT.SANDBOX.Z1C.SBOX00                        (ISC)               *
*    60%      89     1,569       942         0         0       117  *
*CAS*****************************************************************
```

Figure 3-5 F CATALOG,REPORT,CACHE command output

In-storage Catalog (ISC) cache

ISC cache is the default cache, and is in the CAS private storage. Each user catalog has its own cache with a fixed amount of space for cached records. When all allocated space is used, the LRU algorithm is used to make room for the new entry.

The master catalog does not have a fixed amount of cache space, and so uses as much as it needs. All eligible records in the master catalog are cached in the ISC as they are read. This process is another good reason to keep in the master only the entries needed, avoiding using too much main storage.

Sharing catalogs also affects cache management for ISC. If a catalog uses the ISC and a sharing system updates any record, catalog management releases the *entire* ISC for the catalog and creates an ISC for the catalog. This process occurs even if the record is not cached in this system's ISC. A catalog with a high update rate affects the performance because the cache is always being invalidated. Catalogs that are not frequently updated or shared with other systems use the ISC most effectively. ISC is a good choice for master catalog when it has only system data sets, aliases, and connector records to user catalogs because it is rarely updated.

As said before, ISC is the default cache, but you can alter it by using the F CATALOG command to manipulate the cache location. The following command can be used to stop an allocated catalog from using ISC:

```
F CATALOG,NOISC(UCAT.TESTFR)
```

Catalog Data Space Cache (CDSC)

CDSC is in a VLF managed data space. The catalogs are defined in the COFVLFxx PARMLIB member. There are considerable differences from ISC. Catalogs are not limited to a set amount of storage. They can use as much as they need up to the maximum amount of space that is set by the MAXVIRT parameter in the COFVLF member. When sharing catalogs, CDSC can identify individual records that a sharing system has updated and CDSC space used by the catalog is not necessarily released. CDSC space for a catalog is only "invalidated" if there are excessive changes that occur and the catalog management cannot keep up with the rate of changes. Invalidated means the space is marked unusable and given back to the CDSC as "free space" that any catalog can use. Otherwise, all changes that are made by the sharing system can be made to the CDSC record by record. When a change is detected, only the changed records are released.

When the data space approaches full, the least recently used records are removed. VLF does trimming of unreferenced records when utilization of MAXVIRT approaches 90% to try to keep some space available. Small amounts of trimming are considered normal. A continuous large amount of trimming might indicate that VLF effectiveness is constrained. In this case, address the trimming. To lessen the chance of trimming, specify a large enough value for MAXVIRT. Specifying a MAXVIRT value that is too large can, in some cases, cause high processor utilization in VLF. The value that you insert in MAXVIRT represents 4-KB blocks. For IGGCLASS, 60 KB is generally a good value. Subtype 3 of SMF record type 41 can be used to examine any trimming that might be occurring.

Master catalog can use CDSC but, when you follow the recommendation to keep the master to system data sets, user catalog connectors, and aliases, you do not need to cache. The user catalog connectors and aliases are already kept in the multi-level alias tables in main storage. The systems data set is mostly accessed during IPL and never referred to again.

Catalogs can be added only by restarting the VLF address space, which releases all of the existing CDSCs and data that is cached in VLF. If you turn off VLF caching, the catalog reverts to ISC caching unless you also turn that off. Figure 3-6 shows an example of defining catalog to VLF.

```
CLASS NAME(IGGCAS)      /* IGGCAS is the class name for catalog data space */
      EMAJ(UCAT.TESTFR)      /* catalog name */
      EMAJ(UCAT.PROD2)       /*catalog name */
      MAXVIRT(15)            /* A rule of thumb suggests  15 blocks of 4k = 60KB*/
```

Figure 3-6 COFVLFxx: Defining catalog caching

You also must tell CAS to stop using ISC and start using CDSC. Perform these steps to change a catalog cache management from ISC to CDSC:

1. Update the COFVLFxx parmlib member to add the catalog EMAJ name to the IGGCAS class and recycle VLF.
2. Issue the F CATALOG,CLOSE(catname) command. This command releases the storage used.
3. Issue the F CATALOG,NOISC(catname) command.
4. Issue the F CATALOG,VLF(catname) command.

Guideline: CDSC is a better caching choice in most instances because it has better granularity for invalidation.

3.2.5 GRS configuration

The GRS configuration can affect the overall system performance. This section provides recommendations for using GRS and the catalog address space. For more information about GRS, see *z/OS MVS Planning: Global Resource Serialization*, SA22-7600.

CAS uses the SYSIGGV2 reserve while serializing access to catalogs. The SYSIGGV2 reserve is used to serialize the entire catalog BCS component across all I/O, and to serialize access to specific catalog entries. The SYSZVVDS reserve is used to serialize access to associated VVDS records. The SYSZVVDS reserve along with the SYSIGGV2 reserve provide an essential mechanism to facilitate cross-system sharing of catalogs.

You can prevent such deadlocks by always converting the SYSIGGV2 reserves to SYSTEMS ENQUEUEs by using global resource serialization (GRS) or an equivalent product. The resources SYSZVVDS or SYSVTOC should be either both converted or both excluded in the GRS RNL lists.

The SYNCHRES option of GRS allows an installation to specify whether a system obtains a physical hardware reserve before returning control to the program that issued the ISGENQ or RESERVE macro. This feature helps prevent lockouts by eliminating the window between the time the RESERVE macro is issued and the time the first I/O operation to the device is successfully started. When SYNCHRES=NO, the system places a reserve channel command word (CCW) on the front of the next channel program that runs. When SYNCHRES=YES, the system ensures that the reserve CCW is issued and successful before returning.

Guidelines:

- Use the GRS Star configuration because of its advantages in availability, real storage consumption, processing capacity, and response time.
- If you use GRS Star configuration, ensure that ISGLOCK structure size is large enough to minimize false contention.
- Use SYNCHRES(YES) in the GRSCNFxx PARMLIB member to avoid resource deadlock.
- Use RESMIL(OFF), ACCELSYS(2) in the GRSCNFxx PARMLIB member for improved performance.
- GRS QNAMES SYSZVVDS and SYSZVTOC must be treated the same way to avoid resource deadlocks.

3.3 Catalog problem determination

Correct catalog functioning requires that the structure of the BCS remains healthy and that the data set information contained in the BCS, VVDS, and VTOC is synchronized. That is, the characteristics of a data set must be the same in the data set's entries in the BCS, VVDS, and VTOC. For more information about catalog problems, see *ICF Catalog Backup and Recovery A Practical Guide,* SG24-5644. This section provides only a quick reference of commands you can use, and the new commands and functions that can help you in such matter.

Use the IDCAMS commands to diagnose problems. You must have READ authority for the RACF STGADMIN.IDC.EXAMINE and STGADMIN.IDC.DIAGNOSE profiles, if defined, to run these commands against catalogs.

3.3.1 Using the EXAMINE command

Use this command to check structural integrity in a BCS. A structural error is a problem with the BCS as a VSAM KSDS, not as a catalog. Possible causes might be an interrupted control interval (CI) or control area (CA) split or improper sharing. Check the structural soundness of a catalog if non-zero return codes are issued during backup. Check it as well if a catalog is causing jobs to fail and you cannot trace the failure to unsynchronized entries.

To run this command against a catalog, you must have READ authority for the RACF STGADMIN.IDC.EXAMINE profile, if defined. To issue this command under your TSO session, the EXAMINE command must be included in AUTHCMD, in IKJTSOxx PARMLIB member.

The EXAMINE INDEXTEST command ensures that sequential and key direct access are accurate. It evaluates the index set and sequence set. It also checks vertical and horizontal pointers from one index record to the next and correlates index records with data control areas. Example 3-5 shows an output of the command where no errors were detected.

Example 3-5 Issuing EXAMINE command

```
IDCAMS  SYSTEM SERVICES                                           TIME: 22:53:57

   EXAMINE NAME(MCAT.OS3R4V01.VTOTCAT) INDEXTEST
IDC01700I INDEXTEST BEGINS
IDC11773I           268 KEYS PROCESSED ON INDEX LEVEL   1, AVERAGE KEY LENGTH:
IDC11773I             2 KEYS PROCESSED ON INDEX LEVEL   2, AVERAGE KEY LENGTH:
IDC11774I CURRENT INDEX CISIZE IS  4096, RECOMMENDED MINIMUM INDEX CISIZE IS  35
IDC01724I INDEXTEST COMPLETE - NO ERRORS DETECTED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

   EXAMINE NAME(MCAT.OS3R4V01.VTOTCAT) DATATEST
IDC01700I INDEXTEST BEGINS
IDC11773I           268 KEYS PROCESSED ON INDEX LEVEL   1, AVERAGE KEY LENGTH:
IDC11773I             2 KEYS PROCESSED ON INDEX LEVEL   2, AVERAGE KEY LENGTH:
IDC11774I CURRENT INDEX CISIZE IS  4096, RECOMMENDED MINIMUM INDEX CISIZE IS  35
IDC01724I INDEXTEST COMPLETE - NO ERRORS DETECTED
IDC01701I DATATEST BEGINS
IDC01709I DATATEST COMPLETE - NO ERRORS DETECTED
IDC01708I 268 CONTROL INTERVALS ENCOUNTERED
IDC01710I DATA COMPONENT CONTAINS 5233 RECORDS
IDC01711I DATA COMPONENT CONTAINS 0 DELETED CONTROL INTERVALS
IDC01712I MAXIMUM LENGTH DATA RECORD CONTAINS 10621 BYTES
IDC01722I 92 PERCENT FREE SPACE
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

The EXAMINE DATATEST command reads all data CIs to ensure structural integrity. It evaluates the index sequence set and the data component by sequentially reading all data control intervals. Tests are conducted to ensure record and control interval integrity, including the connection between sequence set entries and data control interval contents. Such conditions as duplicate records or records out of sequence can be detected. Example 3-5 on page 94 shows an output of the command where no errors were detected.

When EXAMINE identifies the problem with a BCS, you *cannot* use the same procedure that you use with a normal KSDS:

- Deleting and recovering from a recent backup is not enough. You must apply all the changes that occurred after the backup. You need forward recovery using SMF data.
- Unloading data component directly as an ESDS, sorting data, and then loading into new KSDS *does not work* for a BCS.

If a BCS index component is damaged, perform these steps:

- Use IDCAMS REPRO NOMERGECAT to copy catalog records to a new, empty catalog. FROMKEY and TOKEY can be used to get around bad index records. NOMERGECAT is the default. So it is just a normal copy, bypassing the damaged index records, as shown in Example 3-6.

  Example 3-6 Using REPRO to get around the damaged records

  ```
  REPRO -
  IDS(damaged_ucat) -
  ODS(empty_new_ucat) -
  FROMKEY(initial_good_record_key) -
  TOKEY(final_good_record_key)
  ```

- Severe damage requires recovery from backup and forward recovery by using SMF data. For more information, see “Integrated Catalog Forward Recovery Utility” on page 110.
- Repair can be accomplished with IBM Tivoli Advanced Catalog Management for z/OS.

3.3.2 Using DIAGNOSE command

DIAGNOSE is a tool that you use to see synchronization problems between the BCS and VVDS record structure. Catalog entries might become unsynchronized, so that information about the attributes and characteristics of a data set are different in the BCS, VVDS, and VTOC. These differences can make a data set inaccessible or otherwise unusable.

DIAGNOSE assumes that a BCS is structurally correct. It is possible for DIAGNOSE to complete with a return code of zero when EXAMINE shows structural errors. Because the DIAGNOSE command checks the content of the catalog records, and the records might contain damaged length field values, the job might abend.

You need READ authority for the RACF profiles STGADMIN.IDC.DIAGNOSE.CATALOG and STGADMIN.IDC.DIAGNOSE.VVDS to use this command for catalog and VVDS. To issue this command under your TSO session, DIAGNOSE command must be included in AUTHCMD, in IKJTSOxx PARMLIB member.

Use DIAGNOSE ICFCATALOG, without the compare parameter, to check information integrity within each BCS record (inside-the-BCS only) as shown in Example 3-7.

Example 3-7 DIAGNOSE ICFCATALOG showing content errors

```
DIAGNOSE ICFCATALOG INDATASET(UCAT.CRPLUS2)
IDC21364I ERROR DETECTED BY DIAGNOSE:
ICFCAT ENTRY: YCJRES2.CRPLUS.TESTE.E000026.DATA (T)
RECORD: YCJRES2.CRPLUS.TESTE.E000026.DATA /00
OFFSET: X'0036'
```

```
REASON: 26 - CELL TYPE INVALID IN CONTEXT
IDC21365I ICFCAT RECORD DISPLAY:
RECORD: YCJRES2.CRPLUS.TESTE.E000026.DATA /00
*.!..T..D.YCJRES2.CRPLUS.TESTE.E0*
*00026.DATA ........YCJ*
*RES2.CRPLUS.TESTE.E000026. *
IDC21364I ERROR DETECTED BY DIAGNOSE:
ICFCAT ENTRY: YCJRES2.CRPLUS.TESTK.K000026.DATA (D)
RECORD: YCJRES2.CRPLUS.TESTK.K000026 /00
OFFSET: X'004A'
REASON: 23 - TRUENAME LOOP FAILURE
IDC21365I ICFCAT RECORD DISPLAY:
RECORD: YCJRES2.CRPLUS.TESTK.K000026 /00
*....C....YCJRES2.CRPLUS.TESTK.K0*
*00026 ...........*
*............D.$..YCJRES2.CRPLUS.*
*TESTK.K000026.DATA..............*
*...........SBOX75........}......*
*.......I.*..YCJRES2.CRPLUS.TESTK*
*.K000026.INDEX..................*
*.......SBOX75........}..........*
IDC21365I ICFCAT RECORD DISPLAY:
RECORD: YCJRES2.CRPLUS.TESTK.K000026.DATA /00
*.!..T..D.YCJRES2.CRPLUS.TESTK.K0*
*00026.DATA ........YCJ*
*RES2.CRPLUS.TESTZ.K000026. *
IDC21364I ERROR DETECTED BY DIAGNOSE:
ICFCAT ENTRY: YCJRES2.CRPLUS.TESTK.K000026.DATA (T)
RECORD: YCJRES2.CRPLUS.TESTK.K000026.DATA /00
OFFSET: X'0036'
REASON: 20 - ASSOCIATION NOT FOUND
IDC01371I RECORD DISPLAY SUPPRESSED, ALREADY DUMPED.
IDC21363I THE FOLLOWING ENTRIES HAD ERRORS:
YCJRES2.CRPLUS.TESTE.E000026.DATA (T) - REASON CODE: 26
YCJRES2.CRPLUS.TESTK.K000026.DATA (D) - REASON CODE: 23
YCJRES2.CRPLUS.TESTK.K000026.DATA (T) - REASON CODE: 20
```

Use DIAGNOSE VVDS, without the compare parameter, to check information integrity within each VVDS record (inside-the-VVDS only).

When DIAGNOSE identifies the problem that might indicate an incomplete catalog entry, perform these fixes:

- If it is an entry in a BCS, delete the catalog record and attempt to recatalog it:
  a. DELETE *entry_name* NOSCRATCH
  b. DEFINE *entry_name* RECATALOG
- If the truename exists without the associated cluster records:
  – DELETE *entry_name* TRUENAME
- If it is an entry in a VVDS, delete the record type:
  – DELETE *entry_name* VVR
  – DELETE *entry_name* NVR

- It might be possible to recatalog the data set:
  - DEFINE CLUSTER(NAME(*cluster_name*) … RECATALOG)

3.3.3 Locking Catalogs

When you are performing certain maintenance or recovering a catalog, it is a good idea to LOCK the catalog to prevent undesired access. Locking a catalog makes it inaccessible to all users without read authority to RACF FACILITY class profile IGG.CATLOCK. For protected catalogs, locking an unlocked catalog requires also ALTER authority for the catalog that you are locking.

Lock a catalog by using IDCAMS ALTER LOCK command, as shown in Example 3-8.

Example 3-8 Locking a catalog

```
//LOCKCAT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
    ALTER SYS1.ICFCAT.PROJECT1 LOCK
/*
```

Be sure to UNLOCK the catalog as soon as maintenance is complete as shown in Example 3-9.

Example 3-9 Unlocking a catalog

```
//LOCKCAT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
    ALTER SYS1.ICFCAT.PROJECT1 UNLOCK
/*
```

Guideline: If you lock a shared master catalog from another system, make sure that it is unlocked as soon as possible. If the master catalog is locked, the system cannot be IPLed.

3.4 Catalog management

This section addresses managing your catalogs. The following topics are covered:

- Catalog sharing
- Creating a balanced catalog environment
- Reorganizing catalogs
- Using the Catalog Search Interface (CSI)
- SYSZTIOT catalog contention detection
- Catalog autotuning function
- Catalog backup and recovery
- BCS forward recovery
- RACF and REPRO MERGECAT command
- Catalog IMBED/REPLICATE health check
- Helpful frequently asked questions
- Diagnosing prolonged catalog ENQ times
- Monitoring the catalog address space

3.4.1 Catalog sharing

A catalog is shared when eligible to be used by more than one system image. There are two requirements for using the sharing protocol be used for a catalog:

1. The catalog must have SHAREOPTIONS(3 4). The cross-region 3 value (multiple reads and writes with the user providing read and write integrity) is possible because CAS address space is the only real user of the catalog. CAS does the serializing. The second parameter, 4, is cross-system allowing multiple reads and writes with user responsible for integrity. However, VSAM assists by invalidating buffers and forcing it to read again rather than reuse contents of data and sequence set buffers. Most broken catalogs occur because of improper sharing values.
2. The catalog must be allocated in a volume that is shared by all systems that access the catalog. The volume is defined as shared by using the hardware configuration definition (HCD) facility. If the volume is not defined as shared in some system that access the catalog, it will be corrupted. Corruption occurs because the CAS address space in that system does not use the share protocol when updating the catalog. Catalogs that are defined as unshared and on shared volumes become damaged if referred to by another system.

There is information necessary to communicate changes in a BCS to other systems that are sharing the catalog. Before each physical access to a shared catalog, CAS address space runs special checks to ensure that the cache contains current information. It also ensures that the access method control blocks for the catalog are updated if the catalog is extended, or otherwise altered from another system. This checking maintains data integrity. It also affects performance because the VVR for a shared catalog must be read before using the cache version of the BCS record. There are two protocols that can be used to share that information:

- VVDS mode
- Enhanced Catalog Sharing (ECS) mode

ECS and VVDS protocols cannot be used simultaneously for a catalog. This restriction is enforced by the catalog address space. If you attempt to use a catalog that is ECS-active from a non-ECS system in the sysplex, the associated catalog request fails with return code RC228 and reason code RSN26.

VVDS mode

VVDS mode is the default protocol. The information that is needed to communicate catalog changes to other systems that share the catalog is stored in a special record in the VVDS of the volume the catalog is on. This record is called the *integrity* or time-stamp VVR. This VVR is used by all systems to communicate changes that are made to the BCS. Catalog management checks this VVR at each reference. To read this record, I/O to the VVDS is required. This I/O load can become significant and have a noticeable effect on system or sysplex performance.

Enhanced Catalog Sharing (ECS) mode

In ECS mode protocol, the integrity VVR record is copied into a cache structure that is defined in the coupling facility. Sharing systems retrieve and update information from this location, eliminating the I/O to the VVDS at each catalog reference. This configuration results in better sysplex-wide performance. The SYSZVVDS RESERVE is avoided. If you attempt to use a catalog that is ECS-active from a system outside the sysplex, the request might break the catalog.

For more information about setting the ECS mode, see *z/OS DFSMS Managing Catalogs,* SC26-7409.

Catalog sharing scenarios and GRS definitions

APAR II14297 covers the various combinations of GRS and catalog definitions. In all cases, the GRS resource names SYSVVDS and SYSVTOC are treated in the same way to avoid deadlocks.

Guidelines:

- In Parallel Sysplex environment, use ECS.
- Use SHAREOPTIONS(3 4) for shared catalogs and allocate them in volumes that are shared by all system that access the catalog.
- To prevent potential catalog damage, never place a catalog with share options (3 3) on a shared device.
- For non-shared catalogs, use SHAREOPTIONS(3 3) and place them in non-shared volumes.

3.4.2 Creating a balanced catalog environment

Most z/OS systems have hundreds of thousands to millions of data sets cataloged, and typically have from 25 to 100 catalogs on a system. Assume an environment with 1 million cataloged data sets and 25 catalogs, which is a fairly common ratio. If the spread of data sets across catalogs is even, that works out to 40,000 data sets in each catalog. Therefore, if any one of the 25 catalogs suffers an outage, access to 40,000 data sets is lost until the catalog is recovered. This figure can be much worst. Data sets are rarely distributed evenly across catalogs. More frequently, a handful of catalogs contain a high percentage of the application data sets. If one of those catalogs suffers an outage, access to a far higher number of data sets is affected.

Avoid this situation by analyzing your z/OS environment to determine whether your data sets are concentrated in a few user catalogs, and assess the risk to your critical business functions. If they are concentrated, initiate a project to spread your cataloged data sets across more of your user catalogs.

3.4.3 Reorganizing catalogs

It is common to schedule catalog reorganization that is based on time or when the number of splits is high. This is not a good practice for catalogs. In general, catalogs have unevenly distributed record insertion activity. Splits, both CI and CA, are the best technique available to handle this uneven distribution. Splitting is the way VSAM deals with lack of space for insertion between records, and it generates free space for new insertions. Allow splits to obtain free space where it is needed.

If you are worried about wasting space for empty CAs, since z/OS DFSMS V1R12, the VSAM CA reclaim function is available, even for catalogs. By default, the function is not enabled. For more information, see Chapter 4, “VSAM performance” on page 127.

When reorganizing catalogs, use the IDCAMS EXPORT command followed by IMPORT. You can also reorganize catalogs without taking applications out of service by using Tivoli Advanced Catalog Management for z/OS.

Guidelines: Reorganize catalog only in these circumstances:

- The catalog is approaching the maximum of 123 extents
- There is not enough room on the catalog's volume to allow more extents to be taken
- An attribute of the catalog must be changed

3.4.4 Using the Catalog Search Interface (CSI)

CSI is a valuable tool that you can use to obtain information about entries in catalogs. This section provides a basic overview of the tool. For more information about CSI, see Chapter 11, "Catalog Search Interface User's Guide", in *z/OS DFSMS Managing Catalogs,* SC26-7409.

Study the samples that are provided in the SYS1.SAMPLIB data set. For a description, see 2.5, "Catalog Search Interface" on page 67. The samples are also documented in *z/OS DFSMS Managing Catalogs,* SC26-7409. You can use them to learn how to use CSI services by using them as model to adapt and get the information that you need.

If you are not skilled in coding assembler, do not worry. The assembler samples come with JCL. You just need to complete the JOB card. The JCL does not save the load module in a permanent load library. You can add and customize the overrides that are shown in bold in Example 3-10 to save the load module in your load library.

Example 3-10 Compiling assembler samples

```
//STEP1    EXEC HLASMCLG,COND.G=((0,EQ),(0,NE))
.
.
.
//L.SYSLIN DD
//       DD DSN=SYS1.LINKLIB(IGGCSI00),DISP=SHR <---
//L.SYSLMOD DD DSN=YOUR_LOAD_LIBRARY_NAME(module_name),DISP=SHR
```

You can use generic filter keys to identify the data sets you want information about. Use as many generic filter keys as you need to make a filter. For more information about their meaning and examples, see Chapter 11, "CSIFILTK, Generic Filter Key", in *z/OS DFSMS Managing Catalogs,* SC26-7409.

The information that you can obtain, the field name you must inform to CSI to get the information, and size are listed in chapter 11, "Field Name Directory", in *z/OS DFSMS Managing Catalogs,* SC26-7409.

The information returns in a work area you pass to the CSI. This work area can have from 1 KB to 1 MB. Generally, the best size is 64 KB.

The CSI module name is IGGCSI00, and is in SYS1.LINKLIB. Example 3-12 on page 102 shows a sample REXX that starts CSI to list data sets with IMBED, REPLICATE, and KEYRANGE attributes. Figure 3-7 on page 104 shows the JCL that you can use, given that you saved the REXX as IRK.

The input is a catalog name. If this catalog is a master catalog, IRK gets the user catalog names connected to this master and searches both them and the master.

In the first part, CSI brings the user catalogs names:

- CSIFILTK set to bring all data set names. The generic filter key that is used is **.
- CSIDTYPS set to ‘U’ to search only connector records.
- CSIS1CAT set to search just the catalog that is informed in the variable CSICATNM.
- CSINUMEN is zero because no additional information is requested because you want only the user catalogs name connected.
- WORKLEN set the work area size to 64000 bytes. This work area is the variable DWORK, whose first 4 bytes contains its size.
- RESUME is a variable set to return information from CSI in the CSIFIELD. CSI set to Y when there is more information to return in the work area and CSI must be restarted to bring this information.
- The user catalog names are stored in a data stack, which is accessed by the UCAT.N stem variables.

In the second part, CSI searches for VSAM data sets names and information about the VSAM type of data set:

- The catalog searched in the first part is included in the stack to be searched for VSAM data sets having the searched attributes.

  If no user catalogs were connected to the catalog informed in the PARM, just this catalog if searched. This configuration is a way to search only specific catalogs. However, you must run one execution for each catalog, and you must know the catalog name.
- CSICATNM is set with user catalog name found in the first part. The DO WHILE loop, which is controlled by the L variable, runs CSI for each user catalog.
- CSIDTYPS is set to C to bring information about VSAM cluster. CSI brings information about the cluster and each VSAM component.
- When CSIOPTNS is set to F, the length fields are returned in 4 bytes. Otherwise the length is 2 bytes, as in this example.
- The wanted information is in the field VSAMTYPE. For more information about this field, see Chapter 11, “Field Name Directory”, in *z/OS DFSMS Managing Catalogs,* SC26-7409. You need just this field and its name is set in the CSIFLD1. So CSINUMEN is set to 1.
- When you want more than one returning field, put the field names in CSIFLD1, adjust the size to 8 bytes for each field name and adjust the CSINUMEN value as shown in Example 3-11.

  Example 3-11 Informing more than one field name

  ```
  CSIFLD1  = SUBSTR('VSAMTYPEHURBA   LRECL   ',1,24)
  CSINUMEN = '0003'X
  ```

- The return code is 4 if CSI found any catalog entry error. This error might be a VSAM data set name in the catalog, but in the VVDS. These errors are skipped.
- The information that you want is in the VSAM components. So the cluster entry is skipped, C just before the cluster name. See the topic Work Area Format Table for CSIETYPE field, in *z/OS DFSMS Managing Catalogs,* SC26-7409.

- The information that you want is in the bits of VSAMTYPE field. List only the data sets that have IMBED (I), REPLICATE (R), and KEYRANGE (K) bits on. The output is in DDNAME SYSTSPRT and looks like Example 3-12.

Example 3-12 IRK: Finding VSAM data sets with IMBED, REPLICATE, and KEYRANGE attributes

```
/* REXX */
/*TRACE R*/
 /*****************************************************************/
 /* FUNCTION:                                                     */
 /* SEARCH CATALOG FOR IMBED REPLICATE KEYRANGE VSAM ATTRIBUTES     */
 /* CALLING INFORMATION:                                          */
 /*       EXEC REXX_NAME 'MASTERCAT_NAME'                         */
 /*****************************************************************/
 /*****************************************************************/
 /* FIRST PART:  SEARCH MASTER FOR THE CONNECTOR RECORDS, TO GET THE */
 /* USER CATALOGS NAMES AND SAVE INTO UCAT. STEM VARIABLES  */
 /*****************************************************************/
 PARSE UPPER ARG CAT
 IF CAT = '' THEN
    DO
    SAY "NO CATALOG INFORMED. "
    EXIT
    END
MODRSNRC = SUBSTR(' ',1,4)
CSIFILTK = SUBSTR('**',1,44)
CSICATNM = SUBSTR(CAT,1,44)
CSIRESNM = SUBSTR(' ',1,44)
CSIDTYPS = SUBSTR('U',1,16)         /*JUST CONNECTOR RECORDS */
CSICLDI  = SUBSTR(' ',1,1)
CSIRESUM = SUBSTR(' ',1,1)          /*   RESUME FLAG       */
CSIS1CAT = SUBSTR('Y',1,1)          /*   SEARCH 1 CATALOG  */
CSIOPTNS = SUBSTR(' ',1,1)
CSINUMEN = '0000'X                  /* NO INFORMATION FIELDS */
CSIFLD1  = SUBSTR('        ',1,8)
CSIOPTS  = CSICLDI || CSIRESUM || CSIS1CAT || CSIOPTNS
CSIFIELD = CSIFILTK || CSICATNM || CSIRESNM || CSIDTYPS || CSIOPTS
CSIFIELD = CSIFIELD || CSINUMEN || CSIFLD1
WORKLEN = 64000                     /*WORK AREA LENGTH */
DWORK = D2C(WORKLEN,4)|| COPIES('00'X,WORKLEN-4)
NUMERIC DIGITS 16
RESUME = 'Y'
N=0            /* COUNT USER CATALOGS */

DO WHILE RESUME = 'Y'
   ADDRESS LINKPGM 'IGGCSI00  MODRSNRC  CSIFIELD  DWORK'
   LCC = RC
   IF LCC ¬= 0 THEN
      DO
      REASON = C2X(MODRSNRC)
      SAY 'IGGCSI00 NOT OK RC:' LCC 'REASON:' REASON
      EXIT
      END
   RESUME = SUBSTR(CSIFIELD,150,1)  /* RESUME FLAG LOOP CTL */
   USEDLEN = C2D(SUBSTR(DWORK,9,4)) /* USED WORK AREA LENGTH  */
   POS1=15                          /* AREA INFO ENTRY  */
   /****************************************************************/
   /*WORK AREA DATA: GET USER CATALOGS NAMES                      */
   /****************************************************************/
   DO WHILE POS1 < USEDLEN            /* PROCESS  WORK AREA */
      IF SUBSTR(DWORK,POS1+1,1) = '0' THEN POS1 = POS1 + 50
      ELSE
         DO
           CSIEFLAG = SUBSTR(DWORK,POS1,1)
           X = BITAND(CSIEFLAG,'20'X)            /*  VALID? */
           IF X /= '20'X THEN  POS1 = POS1 + 50
           ELSE
             DO
             N = N + 1
             UCAT.N = SUBSTR(DWORK,POS1+2,44)
             POS1 = POS1 + 50
             END
         END
   END
END
 /*******************************************************************/
 /* SECOND PART:  SEARCH EACH USER CATALOG. LIST VSAM DATA SETS WITH */
 /* IMBED (I) AND/OR REPLICATE(R) AND/OR KEYRANGE ATTRIBUTES        */
 /*******************************************************************/
L = 0
N = N + 1
UCAT.N =  CAT
DO WHILE L < N
   L = L + 1
   MODRSNRC = SUBSTR(' ',1,4)
   CSICATNM = SUBSTR(UCAT.L,1,44)
   CSIRESNM = SUBSTR(' ',1,44)
   CSIDTYPS = SUBSTR('C',1,16)      /*CLUSTERS*/
   /*  OPTIONS */
   CSICLDI  = SUBSTR(' ',1,1)
   CSIRESUM = SUBSTR(' ',1,1)          /*   RESUME FLAG  */
   CSIS1CAT = SUBSTR('Y',1,1)          /*   SEARCH 1 CATALOG  */
   CSIOPTNS = SUBSTR(' ',1,1)
   CSINUMEN = '0001'X
   CSIFLD1  = SUBSTR('VSAMTYPE',1,8)

   CSIOPTS  = CSICLDI || CSIRESUM || CSIS1CAT || CSIOPTNS
   CSIFIELD = CSIFILTK || CSICATNM || CSIRESNM || CSIDTYPS || CSIOPTS
   CSIFIELD = CSIFIELD || CSINUMEN || CSIFLD1
   RESUME = 'Y'
   SAY   'SEARCHING ==>' UCAT.L
   DO WHILE RESUME = 'Y'
      ADDRESS LINKPGM 'IGGCSI00  MODRSNRC  CSIFIELD  DWORK'
      LCC = RC
      REASON = C2X(MODRSNRC)
      IF LCC = 0 | ( LCC = 4 & (SUBSTR(MODRSNRC,4,1) = '64'X)) THEN
      DO
        RESUME = SUBSTR(CSIFIELD,150,1)  /* RESUME FLAG LOOP CTL */
        USEDLEN = C2D(SUBSTR(DWORK,9,4)) /* USED WORK AREA LENGTH  */
        POS1=15                          /* AREA INFO ENTRY  */
        /**********************************************/
```

```
         /*WORK AREA DATA: GET VSAM IMBED REPLICATE KR */
         /*********************************************/
         DO WHILE POS1 < USEDLEN
            IF SUBSTR(DWORK,POS1+1,1) = '0' THEN POS1 = POS1 + 50
            ELSE
               DO
               CSIEFLAG = SUBSTR(DWORK,POS1,1)
               X = BITAND(CSIEFLAG,'60'X)   /* FLAG ENTRY ERROR*/
               IF X > '20'X THEN  POS1 = POS1 + 50   /*SKIP*/
               ELSE
                 DO
                 TOTLEN = C2D(SUBSTR(DWORK,POS1+46,2))
                 IF SUBSTR(DWORK,POS1+1,1) /= 'C' THEN
                   DO
                   VSAMTYPE = SUBSTR(DWORK,POS1+52,1)
                   INFO = ' '
                   IMBED = BITAND(VSAMTYPE,'20'X)
                   KEYRANGE = BITAND(VSAMTYPE,'04'X)
                   REPLICATE = BITAND(VSAMTYPE,'10'X)
                   IF IMBED = '20'X THEN INFO = 'I'
                   IF REPLICATE = '10'X THEN INFO = 'R'||INFO
                   IF KEYRANGE = '04'X THEN INFO = 'K'||INFO
                   IF INFO /= ' ' THEN
                     SAY '    '||SUBSTR(DWORK,POS1+2,44) ' - ' INFO
                   END
                 POS1 = POS1 + 46 + TOTLEN
                 END
               END
         END
       END
       ELSE RESUME = ' '
    END
END
```

Figure 3-7 shows the JCL you use to start IRK REXX in a batch environment.

```
//IRJJOB JOB .....
// NOTIFY=&SYSUID,TIME=1440,REGION=6M
//*************************************************
//IRXJCL EXEC PGM=IRXJCL,PARM='IRK catalog_name'
//SYSTSPRT DD  SYSOUT=*
//SYSEXEC  DD  DSN=pds_rexx_source,DISP=SHR
//SYSTSIN  DD  DUMMY
```

Figure 3-7 JCL to start IRK REXX in batch environment

Figure 3-8 shows the output of the IRK REXX batch job.

```
SEARCHING ==> CATALOG.DB2ICF2.VTOTCAT
    CATALOG.DB2ICF2.VTOTCAT                    -  I
    CATALOG.DB2ICF2.VTOTCAT.CATINDEX           -  RI
SEARCHING ==> CATALOG.HCD.USERCAT
    CATALOG.HCD.USERCAT                        -  I
    CATALOG.HCD.USERCAT.CATINDEX               -  I
SEARCHING ==> CATALOG.HSM
SEARCHING ==> CATALOG.MHLRES2
SEARCHING ==> CATALOG.SHRICF1.VIODFPK
    CATALOG.SHRICF1.VIODFPK                    -  I
    CATALOG.SHRICF1.VIODFPK.CATINDEX           -  I
SEARCHING ==> CATALOG.TOTICFM.VTOTCAT
SEARCHING ==> CATALOG.TOTICF1.VTOTCAT
    ASNA.SMPCSI.CSI.DATA                       -  I
    ASNA.SMPCSI.CSI.INDEX                      -  I
    ASNA.V5R1M0.MSGS.DATA                      -  I
    ASNA.V5R1M0.MSGS.INDEX                     -  I
    ASNL.V5R1M0.MSGS.DATA                      -  I
    ASNL.V5R1M0.MSGS.INDEX                     -  I
    CICSTS52.REXX.FP01.DIR.DATA                -  I
    CICSTS52.REXX.FP01.DIR.INDEX               -  I
    CICSTS52.REXX.FP01.FILE01.DATA             -  I
    CICSTS52.REXX.FP01.FILE01.INDEX            -  I
```

Figure 3-8 REXX IRK output

The following are some highlights to help you with the work area that contains the returning information. For more information, see Chapter 11, “Catalog Search Interface User's Guide” in *z/OS DFSMS Managing Catalogs,* SC26-7409.

- The Work Area Format Table contains the description of the returned fields.
- The catalog information is always listed more than once when more than one catalog is searched (CSIS1CAT not ‘Y’).
- After the catalog information, comes the entry identification, which is preceded by 2 bytes. First byte is CSIFLAG that identifies if the entry is in error or if there is data for this entry. If error, instead of returning data, comes return and reason code for the error, after de entry name. The second byte identifies the entry type. Then comes the entry name.
- When you do not request any data field, next 4 bytes are zeros or contain error reason and return code entry error. Then starts information for the next entry or catalog, if more than one catalog is being searched and catalog information changed.
- If you requested that field information and data is returned, it comes after the entry name. You can view this by checking CSIFLAG bits. The way the data is returned is shown in Figure 3-9 on page 106.
- The meaning of the returned data that you find in “Field Name Directory”.
- Then comes the information for the next entry or catalog, if more than one catalog is being searched and catalog information changed.

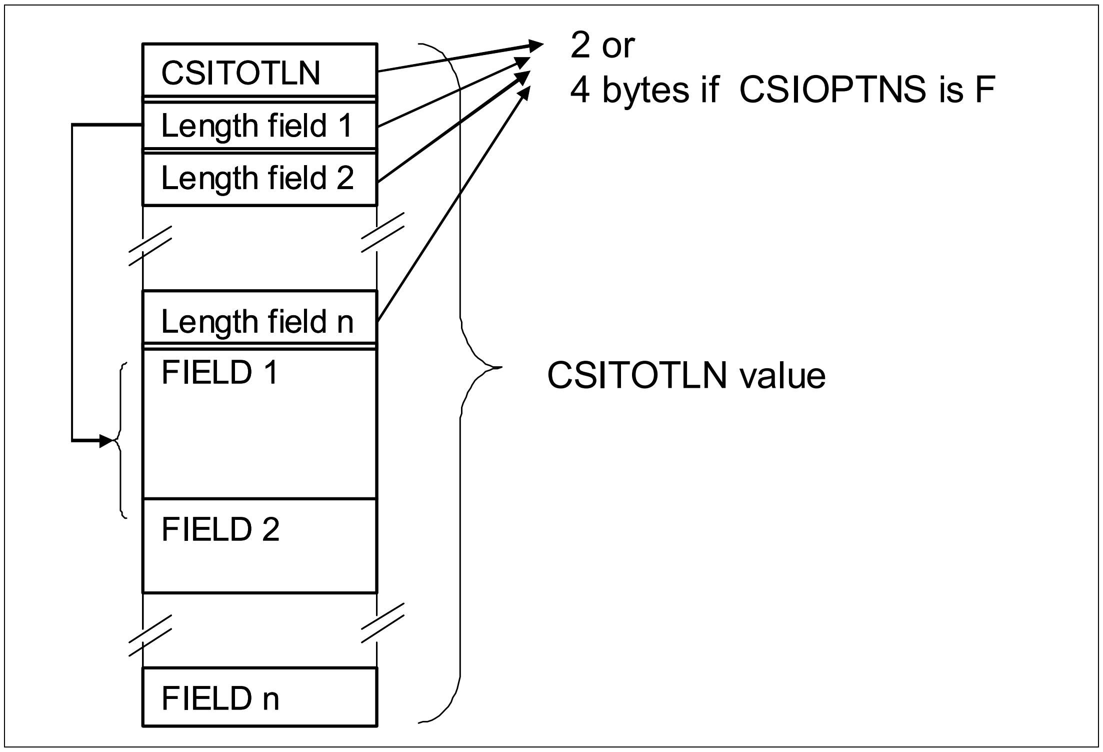


Figure 3-9 Work area returning data for each entry that matches the filter

3.4.5 SYSZTIOT catalog contention detection

When a process or task, holding the SYSZTIOT shared or exclusive, calls CATALOG, it might lead to dead-locks:

- If the caller holds it shared and CATALOG in some task that must get SYSZTIOT EXCLUSIVE, CATALOG will be stuck waiting on the calling tasks SHARED.
- If the calling task already holds SYSZTIOT EXCLUSIVE, CATALOG will not be able to get SYSZTIOT SHARED. Getting SYSZTIOT shared is fairly typical. The EXCLUSIVE requirement is not typical, but does exist.

Since z/OS V1 R12, Catalog Contention Detection runs as part of the existing Catalog Address Space Analysis Task. the Analysis Task runs every 30 seconds or when a specific analysis is requested. Contention Detection is run every time the Analysis Task runs.

SYSZTIOT Catalog Contention Detection was created to notify the operator that SYSZTIOT contention is occurring. It notifies you which JOB,ASID, and TASK might be causing contention using message IEC393I. The job/task can then be canceled to release the dead-lock. Also, reevaluate why the task needs to hold the SYSZTIOT resource while it is calling CATALOG.

Contention Detection scans the Catalog Service Tasks list and runs these processes:

- Checks for service tasks that are active and waiting on the Task input/output table (SYSZTIOT) enqueue. It sets timer for each waiting task.
- On subsequent checks, if the service task timer exceeds the allowed interval wait time (default 10 minutes):
  - Symptom records (SYMREC) are created.
  - Messages are written to the console on the first occurrence. If wait persists, another message is issued at 5 minutes and then at 15-minute intervals thereafter.

You can use the F CATALOG,CONTENTION(*reason class*, *wait-time*) to set the wait time. The command has these variables:

- *Reason class* specifies the group of events that the wait-time field applies to. Currently, the only reason class available is SYSZTIOT.
- *Wait-time* specifies how long a CAS Service Task can wait on an event before notification by a symptom record. If time is zero, contention detection is disabled. The default value for this field is 10 minutes. The valid values are 0, >=5, <=999. The F CATALOG,REPORT console command output shows the wait time set, as shown in Figure 3-10.

```
F CATALOG,REPORT
IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
IEC359I CATALOG REPORT OUTPUT
*CAS*************************************************************
*  CATALOG COMPONENT LEVEL   = HDZ1C10                           *
*  CATALOG ADDRESS SPACE ASN = 0037                              *
*  SERVICE TASK UPPER LIMIT  =  180                              *
*  SERVICE TASK LOWER LIMIT  =   60                              *
*  HIGHEST # SERVICE TASKS   =   25                              *
*  # ATTACHED SERVICE TASKS  =   25                              *
*  MAXIMUM # OPEN CATALOGS   = 1,024                             *
*  ALIAS TABLE AVAILABLE     = YES                               *
*  ALIAS LEVELS SPECIFIED    = 1                                 *
*  SYS% TO SYS1 CONVERSION   = OFF                               *
*  CAS MOTHER TASK           = 007FF368                          *
*  CAS MODIFY TASK           = 007A2DC8                          *
*  CAS ANALYSIS TASK         = 007A2968                          *
*  CAS ALLOCATION TASK       = 007A2B98                          *
*  CAS ECS TASK              = 007A2738                          *
*  VOLCAT HI-LEVEL QUALIFIER = SYS1                              *
*  NOTIFY EXTENT             =   80%                             *
*  DEFAULT VVDS SPACE        = ( 11, 11) TRKS                    *
*  CONTENTION SYSZTIOT TIME  =    10                             *
*  ENABLED FEATURES          = DSNCHECK DELFORCEWNG SYMREC       *
*  ENABLED FEATURES          = UPDTFAIL                          *
*  DISABLED FEATURES         = VVRCHECK AUTOTUNING BCSCHECK      *
*  INTERCEPTS                = (NONE)                            *
*CAS*************************************************************
IEC352I CATALOG ADDRESS SPACE MODIFY COMMAND COMPLETED
```

Figure 3-10 F CATALOG,REPORT output that displays SYSZTIOT Contention Detection wait time

IEC393I is an informational message warning that at least one CAS task has exceeded the wait time that is associated with the reason class. Currently, only the SYSZTIOT reason class is implemented. The message can contain up to five lines. These lines display a JOBNAME and number of tasks within the job past the current limit, or a JOBNAME and the TCB of the single task in the job beyond the limit.

3.4.6 Catalog autotuning function

This function is disabled from the system by the APAR OA25072. As such, the MODIFY CATALOG,ENABLE(AUTOTUNING) operator command will not enable the Auto Tuning function, but also will not return an error. The *z/OS DFSMS Managing Catalogs,* SC26-7409, V1R12 edition, still refers to this feature as active it but it should be removed from the next edition.

This function was designed to improve BUFND, BUFNI, and STRNO parameters for catalogs. However, the implementation of Autotuning support is better done in a development release environment.

3.4.7 Catalog backup and recovery

Catalogs hold information about how to locate hundreds of thousands or even millions of data sets. The catalog is therefore a sensitive part of the system. They are shared across multiple systems, where in-storage control blocks and buffers are maintained and accessed by each system during open, close, and extending data sets. A fail in software code, an improper sharing setting, or other problem can cause a catalog to break, affecting your business applications availability. When this happens, you must restore your catalog information as soon as possible. The recovery procedure is accomplished by restoring the last backup and applying the changes that were made after the backup were done.

This section present some rules to follow so you can have a faster and more secure recovery from a catalog crash:

- Rule #1: Back up as often as you can
- Rule #2: Make sure all your catalogs are being backed up
- Rule #3: Double check the backups
- Rule #4: Verify that you can recover

Rule #1: Back up as often as you can

The first decision is how often should you back up your catalogs. Ensure that your backup frequency meets your business resiliency requirements, but is not so frequent as to negatively affect your day-to-day operations. How often is enough depends on how often the catalogs are updated. At least once a day for all catalogs is a basic rule. Back up more often for highly volatile, critical, and shared catalogs. Catalogs that generate lots of SMF records, indicating heavy data set DEFINEs, DELETEs, and allocation extensions should also be backed up often. The more frequently backups are made, the less SMF data you must apply, and faster the catalog can be recovered.

The master catalog is not updated often. Most updates are TSO userid aliases. Nevertheless, back it up just like any other user catalog, and at the same frequency as your user catalogs. A failure of your master catalog causes a total outage on that system. If a single master catalog is shared across a sysplex and has a failure, you will have an outage of all systems in the sysplex. Because the master catalog is critical to your z/OS system, recovery from a failure in the shortest time possible is critical. The fastest way to recover from a failure is to set up and maintain an alternate master catalog.

Rule #2: Make sure all your catalogs are being backed up

Periodically perform a backup audit of all user catalogs that are defined on each system in your environment (for example, production, batch, online, test, development). Ensure that all catalogs and all aliases are being backed up for each catalog. If aliases are not backed up, you risk loss of all aliases for the catalog if a recovery is performed. When the catalog is deleted, the connector record with all of the aliases in the master catalog is also deleted.

Obtain a list of connected catalogs in all master catalogs and compare the list to your catalog backup job to ensure that all are backed up. During the recovery procedure, the worst thing is to discover that the catalog that you need to restore is not being backed up.

Figure 3-11 is output from a sample LISTC of a user catalog.

```
LISTC UCAT
LISTING FROM CATALOG -- MCAT.SANDBOX.Z1C.SBOX00
 USERCATALOG --- CATALOG.DB2ICF2.VTOTCAT
 USERCATALOG --- CATALOG.HCD.USERCAT
 USERCATALOG --- CATALOG.HSM
 USERCATALOG --- CATALOG.SHRICF1.VIODFPK
 USERCATALOG --- CATALOG.TOTICFM.VTOTCAT
 USERCATALOG --- CATALOG.TOTICF1.VTOTCAT
 USERCATALOG --- CATALOG.TOTICF2.VTOTCAT
 USERCATALOG --- CATALOG.USER
 USERCATALOG --- MCAT.OS3RSA.VOS3CAT
```

Figure 3-11 Listing the user catalogs entries from the master

Rule #3: Double check the backups

Double-check that your backups are actually running successfully. Have automation in place to notify you of a problem, or have a production control group that lets you if something happens. Plan to periodically check the return codes from your backups jobs yourself. Automation can, and has, failed. If there are several things going on, you might not get timely notification from your production control group. Schedule a periodic “manual” check of your catalog backup return codes if you are depending on automation to notify you.

Establish a regular method to check catalog backup return codes. Ensure that you run EXAMINE INDEXTEST IDCAMS commands on each catalog and check the output. Consider duplexing your backups and create a third copy for your disaster recovery (DR) site. At least run an IDCAMS EXAMINE INDEXTEST on every catalog at the time it is backed up. You will learn at the first sign of problem if you have any structural errors that are creeping into the index.

Consider creating two copies of your catalog backup to be retained on site to reduce the likelihood of encountering a bad backup file. Put at least one copy of the backup on DASD if possible. The size of the backup file might be a constraint. Being on DASD makes the most recent backup copy more readily available.

Rule #4: Verify that you can recover

Being able to locate your backups in the event of a catalog failure is critical. Ensure that you know which catalog the backup files are cataloged in. If this is the failing catalog, you likely will not be able to access it to locate the catalog backup file. To resolve this issue, make two copies of your catalog backups, and make sure that each copy is registered in a different catalog. Make sure that each backup data set name uses a different high-level qualifier that points to a different user catalog.

Ensure you can quickly and accurately locate the SMF dump data sets from all systems in your environment that share access to the catalog being recovered. If the SMF data is in the catalog that has had the failure, it might be difficult to locate.

Test the recovery procedure to make sure that it works.

3.4.8 BCS forward recovery

When a catalog breaks, you can recover it from a backup and apply the changes that were made since the backup was taken. For every update to the ICF catalog environment, the catalog management system routines create an SMF record. SMF records are the only way for BCS forward recovery. The following SMF record types are required:

- Type 61 – data set define
- Type 65 – data set delete
- Type 66 – data set alter

Recovery tips

Records that are written between the time of backup and restore identify all new data sets created, deleted, and extended. Follow these hints to improve your catalog recovery:

- Make sure that SMF records are being captured and retained on all your systems. Check the SMFPRMxx PARMLIB member for the jobs that dump SYS1.MANx data sets with IFASMFDP. Ensure that the SMF you require for catalog forward recovery are not somehow being discarded.
- Time can be saved during forward recovery processing if a copy of the SMF records required for catalog forward recovery are written to their own unique data set. This way, only the catalog SMF records must be processed at the time of forward recovery. Create a separated data set containing only the required SMF records. They can be split out and created at the time the SYS1.MANx data is dumped with IFASMFDP.
- Create a GDG for the SMF records that are split out.
- Set the limit on each system to allow for the number of dump cycles of the SYS1.MANx data sets that might occur between the catalog backup jobs.
- Catalog recovery is faster if the SMF data that you must process is in a disk data set, rather than tape. When you create an extract of only the needed SMF records, this data set is a more manageable size, You might be able to retain it on DASD. It might be practical to create two copies of this data and catalog them in different user catalogs with different high levels as suggested for catalog backup files.
- Test to make sure that you can recover the catalog. If you have never tried to perform a catalog forward recovery before a catalog failure occurs, you might lose time. You lose time determining where the catalog backups are, where the SMF data dumps are, and how to set up the JCL. Do all of this in advance and set up a process that you practice regularly to test forward recovery.

Integrated Catalog Forward Recovery Utility

Integrated Catalog Forward Recovery Utility (ICFRU) is an IBM field developed product that is incorporated into z/OS DFSMS 1.7 and used for BCS Forward Recovery. ICFRU uses the IDCAMS EXPORT copy of the BCS and SMF records from all sharing systems as input. It creates an EXPORT format backup that is used as input to IDCAMS IMPORT to rebuild the catalog.

If you are using ICFRU or a similar product, there can be several steps that are involved in performing the forward recovery. This complexity makes it all the more critical that you become familiar with how to perform the forward recovery. ICFRU has two different programs:

- ICRURRSV: Record Selection and Validation. Processes dumped SMF data sets and extracts appropriate records
- ICRURRAP: Record Analysis and Processing. Processes the extracted and sorted SMF records. It then updates the backup file to create a EXPORT format backup file that can then be used to build a new catalog.

Whatever process you use, practice forward recovery to minimize the setup time that is required should an actual catalog failure occur.

3.4.9 RACF and REPRO MERGECAT command

IDCAMS REPRO MERGECAT command actually runs a series of commands:

- Locates gather data set information for each entry
- DEFINE RECATALOG for each data set entry
- DELETE NOSCRATCH for each data set entry

The storage administrator originally needed access to all data sets entries that were being moved from one catalog to another. Limit the authority that is required so administrators only need to have access to the catalogs themselves is addressed with the APAR OA33013. It introduced the RACF Facility Class STGADMIN.IGG.DEFINE.RECAT, which requires only the following authority:

- ALTER authority to the source and target catalogs
- READ authority to the following RACF Facility classes:
  - STGADMIN.IGG.DELETE.NOSCRTCH
  - STGADMIN.IGG.DEFINE.RECAT

Read authority to STGADMIN.IGG.DEFINE.RECAT bypasses RACF data set authority when checking for DEFINE RECATALOG. Therefore, no authorization checking is done to the data set if the user has READ authority to STGADMIN.IGG.DEFINE.RECAT class. However, UPDATE authority is required for the catalog.

Read authority to STGADMIN.IGG.DELETE.NOSCRTCH bypasses RACF data set authority when checking for DELETE NOSCRATCH.

3.4.10 Catalog IMBED/REPLICATE health check

Since z/OS V1 R11, the IBM Health Checker looks for catalogs that are defined with IMBED/REPLICATE parameters. It works only for catalogs. This function runs when Health Checker starts, and every 24 hours thereafter.

Health Checker has no input parameters, and reads all user catalog connectors in the master catalog by using the Catalog Search Interface (CSI). It issues a catalog call for each user catalog connector to get the Imbed/Replicate information. This call opens each catalog, which opened can cause messages if user catalog volumes are non-existent or offline. Activate and deactivate it only as needed.

The health check can issue these messages:

- IGGHC103I No catalogs were found defined with IMBED and REPLICATE attributes.
- IGGHC104E The health check has detected one or more catalogs defined with the IMBED and REPLICATE attributes.
- IGGHC106I Following catalog(s) were detected with the IMBED and REPLICATE attributes.

3.4.11 Helpful frequently asked questions

This section answers frequently asked questions about catalogs.

How large should an ICF catalog be?

Base the size of your catalogs on application and recovery needs. The larger a catalog, the more time it takes to recover. Consider the following concerns about your applications:

- How many entries do they require?
- What is the length of time to recover a catalog?
- How long can you tolerate an outage for the application?

Catalogs are now more resilient, and DASD hardware is much more reliable. You can have approximately 3500 aliases per catalog. This size limitation might also be a factor in your catalog's size.

The 4-GB size is no longer a limit. If you define a catalog as extended format, you can set extended addressability for the catalog. Its size limit is then the CI size multiplied by 4 GB. However, the catalog is still limited to 123 extents.

How do I alter BUFNI for a catalog index?

Use the ALTER IDCAMS command. Add the CATALOG parameter to specify the catalog you want to change, and also to indicate the component name. To change any index attribute, specify the catalog name that is appended with “CATINDEX” (Example 3-13).

Example 3-13 Altering BUFNI for a catalog

```
ALTER UCAT.PROD1.CATINDEX BUFNI(6) CAT(UCAT.PROD1)
```

When you do not use the CAT parameter, IDCAMS tries to update the catalog connector record in the master catalog instead of the catalog data set itself. This causes the command to fail.

When you add the CAT parameter with no component name, IDCAMS tries to change the data component. This process fails if the attribute you are trying to change applies only to index component.

How many aliases can a user catalog have?

All alias names are in the user catalog connector record in the master catalog. The number of aliases that a user catalog can have is a factor of the maximum catalog record size, which is 32400 bytes. The number of aliases in a user catalog is dependent on the length of the alias name. Therefore, there is no way to define an exact number.

If you have short alias names, the number is around 3500. if you are using multi-level aliases, the names are longer, so the number possible is reduced.

You delete all the alias records when you delete the user catalog connector record. The user catalog connector record contains these items:

- Name cell, which is 52 bytes in length
- Owner cell, 20-bytes length
- Volume cell, 28-bytes length
- Association cell, variable length and contains the alias names

Therefore, the number of aliases can be calculated by using this formula:

```
(32400-52-20-28-3)/(alias_name_length + 1) = (32297)/(alias_name_length + 1)
```

Assuming 8 is the average length for alias names, the number of aliases possible is 3588.

Is increasing the number of user catalogs per volume useful?

You can have two or more user catalogs on the same volume. With the reliability of RAID devices, vulnerability to crashes is typically not a concern.

When sharing, catalog management issues a RESERVE macro on the volume for each user catalog update access. Having many RESERVEs on the same volume might cause some contention, depending on the level of activity of the catalog involved. However, when you use ECS, RESERVE is not issued, so you can use ECS to get around the RESERVE contention.

Can I uncatalog a base VSAM cluster and not uncatalog the AIXs?

You cannot uncatalog a base VSAM cluster without uncataloging all of the associated AIXs.

Uncataloging means issuing the DELETE NOSCRATH command. The alternate indexes (AIXs) are in a subrecord within the base VSAM cluster record. When you delete the base VSAM cluster record, you also delete all the associated records, including AIX and PATH records. When the base VSAM cluster record and the associated subrecords are removed, the physical data sets still exist on the volumes and can be recataloged by using DEFINE RECATALOG. You must redefine any PATHs with DEFINE PATH command.

3.4.12 Diagnosing prolonged catalog ENQ times

When you are analyzing prolonged catalog ENQ times, verify these items:

- LPAR definition
- GRS configuration
- Catalog workload
- I/O response time and workload

Because elongation of catalog requests might result from increased workload, historical data is required to determine whether workload has increased. SMF records 70-79 and the `F CATALOG,REPORT,PERFORMANCE` command output can provide information about workload trends and system utilization. See Figure 3-20 on page 122 for an output of this command.

Before any diagnosis, perform these tasks to gather the data that are required for diagnosing a perceived catalog performance problem:

- Document LPAR configuration: This information can be obtained from the RMF Partition Data Report.
- RMF records 70-79: Ensure that SMFPRMxxXnn in PARMLIB includes SMF records 70-79. Include type 79 subtype (7) for SENQ reports. For more information, see *z/OS MVS System Management Facitilies (SMF)*, SA22-7630.
- Set up and start RMF reporting. For more information about setting up RMF Post Processor, Monitor II, and Monitor III sessions, see *RMF User's Guide,* SC33-7990.

  Specify ENQ(DETAIL) to report job names that own the resource, have the longest period of contention, and are waiting for a resource.
- Ensure that the dispatching priority for CAS and GRS is allowed to default.
- If GRS is in ring mode, perform these additional tasks:
  - Consider changing to star configuration. If not, review the recommendations for settings in “GRS configuration” on page 93.
  - Run GENQRESP found in SYS1.SAMPLIB(ISGNQRSP) to determine base response time for a GRS ENQ/DEQ request.

- Implement ENQ/RESERVE/DEQ Monitor as described in Chapter 3 of *z/OS MVS Planning: Global Resource,* SA22-7600.
- For catalog, periodically issue the following series of commands (possibly at the beginning of each shift):
  - `F CATALOG,REPORT` to display CAS specifications
  - `F CATALOG,REPORT,CACHE` to display catalog cache hits
  - `F CATALOG,LIST` to display waits, and so on
  - `F CATALOG,REPORT,PERFORMANCE` to record number of entries into catalog, number of ENQs, and so on
  - `F CATALOG,REPORT,PERFORMANCE(RESET)` to reset counters

Catalog LOCATE Flow

If you suspect that catalog contention is causing bad system performance, an understanding of how a catalog request is processed and the components involved is needed.

The following scenarios assume VVDS mode sharing and a GRS Ring environment, although GRS Star can be in use. Example 3-14 shows the output from a GTF trace during a catalog LOCATE request.

Example 3-14 Flow of a catalog LOCATE request

```
SVC 26
SVC 56 SYSZTIOT
SVC 48 SYSZTIOT
SVC 56 SYSIGGV2 CATALOG.MVSICFM.VMASTER
SVC 56 SYSZVVDS CATALOG.MVSICFM.VMASTER
SVC 56 SYSZVVDS VOLA
SSCH   72C6
IO     72C6
SVC 48 SYSZVVDS VOLA
SVC 48 SYSZVVDS CATALOG.MVSICFM.VMASTERSVC 48 SYSIGGV2 CATALOG.MVSICFM.VMASTER
SVC 56 SYSZPCCB PCCB
SVC 48 SYSZPCCB PCCB
SVC 56 SYSIGGV2 CAT1.UCAT
SVC 56 SYSZVVDS CAT1.UCAT
SVC 56 SYSZVVDS VOLB
SSCH   7526
IO     7526
SVC 48 SYSZVVDS VOLB
SVC 48 SYSZVVDS CAT1.UCAT
SVC 48 SYSIGGV2 CAT1.UCAT
```

The output includes the following trace entries:

- SVC 26: LOCATE
- SVC56: ENQ
- SVC48: DEQ
- SSCH: Start subchannel
- IO: I/O to the device

The trace shows the resources that are being serialized and when I/O is performed to obtain the requested catalog entry. The number of Locates issued and the length of time it takes to process ENQ requests are the primary factors that determine how long it takes to return a catalog request.

LPAR considerations

When systems in the parallel sysplex run in LPAR mode and there are multiple partitions in one CEC competing for resources, examine the LPAR definitions. Before CAS or GRS can do any work, they must be dispatched in their z/OS image and the partition they are running in must be dispatched to a physical processor. To ensure that the partitions are given enough access to the physical processors, perform these steps:

- Check the logical CP configuration:
  - Number of CPs available to the partition's operating system
  - Operating system dispatches work to a logical processor
  - LPAR associates a physical processor to a logical processor to run work
- Check weight and capping:
  - Weight specifies the amount of capacity that is guaranteed to partition at time of high processor utilization
  - Capping specifies the amount of capacity that the partition is not allowed to exceed even if there are processor resources available
  - Number of logical CPUs and weight are static parameters that are entered on LPAR definition panels
- Review RMF CPU and Partition reports for partition performance. The RMF Partition Data Report can be used to gather information about LPAR

A situation that might cause perceived bad performance is an LPAR environment referred to as having 'short' engines. For example, assume that you have a CEC that has 4 physical CPs, each having a capacity of 25 MIPs. There are two partitions defined each with three logical CPs sharing the four physical CPs:

- Partition 1 has a weight of 75
- Partition 2 is defined with a weight of 25 (the weight of the partition is less than the total MIPs of the logical processors that are assigned to it)

Assume that there is high processor activity for both partitions:

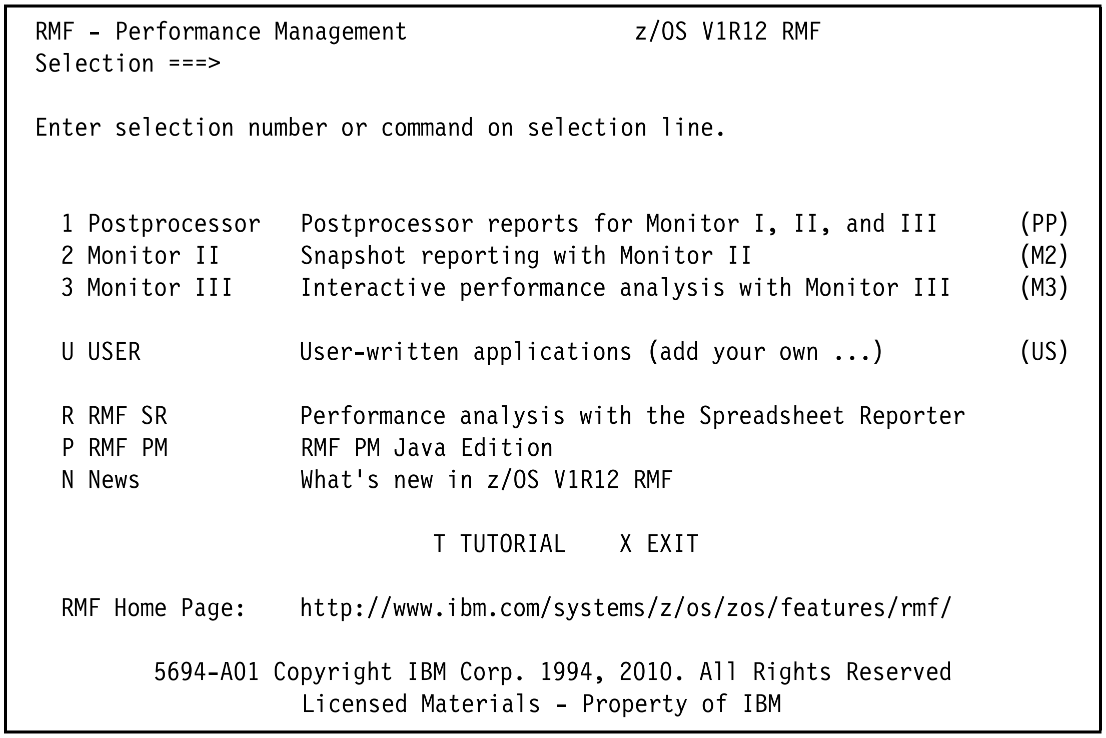
- Partition 2 is dispatched and GRS is dispatched to one of the physical CPs.
- Partition 2 reaches its weight limit and partition 1 requires a CP.
- The physical CP that had been assigned to partition 2 and had GRS dispatched is taken away from partition 2 and assigned to partition 1.

  The z/OS image that is running in partition 2 is not aware that the physical processor has been taken away from it. The work GRS was to perform must wait.

  The work GRS was to perform can be a serialization request from another system that was to be passed onto another system. This serialization request is not granted until the GRS in partition 2 is again assigned a physical processor. This process increases the time that it takes for a request to be granted. If it is a serialization request for a catalog or VVDS, it might increase the elapsed time of a LOCATE request.

Catalog contention

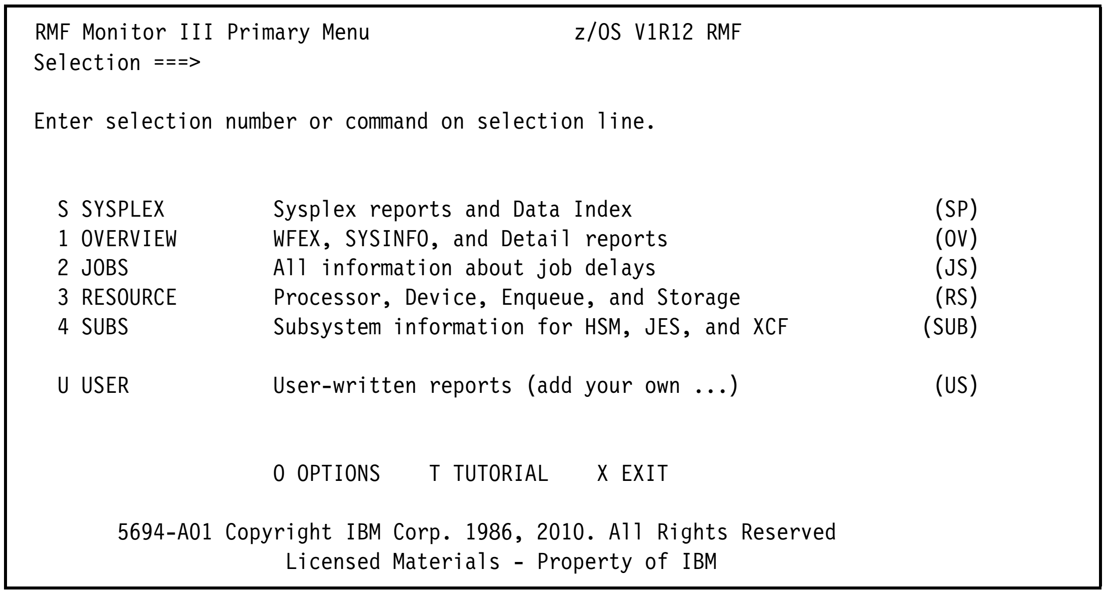
If system performance seems to be bad, use the RMF command to start with your investigation. From the ISPF TSO command shell (option 6), enter the RMF command. This command displays RMF reporting options as shown in In Figure 3-12.



```
RMF - Performance Management                 z/OS V1R12 RMF
Selection ===>

Enter selection number or command on selection line.


  1 Postprocessor   Postprocessor reports for Monitor I, II, and III      (PP)
  2 Monitor II      Snapshot reporting with Monitor II                    (M2)
  3 Monitor III     Interactive performance analysis with Monitor III     (M3)

  U USER            User-written applications (add your own ...)          (US)

  R RMF SR          Performance analysis with the Spreadsheet Reporter
  P RMF PM          RMF PM Java Edition
  N News            What's new in z/OS V1R12 RMF

                              T TUTORIAL    X EXIT

  RMF Home Page:    http://www.ibm.com/systems/z/os/zos/features/rmf/

         5694-A01 Copyright IBM Corp. 1994, 2010. All Rights Reserved
                    Licensed Materials - Property of IBM
```

Figure 3-12 RMF Monitor panel

If application work is running slowly, it might be caused by delays. These delays might be caused by a unit of work that is waiting for requests to be processed by DFSMShsm, JES, Operator Reply, GRS, or other system functions. From the RMF Monitor Panel, select option 3, for Monitor III. The Monitor III options are shown in Figure 3-13.



```
RMF Monitor III Primary Menu                 z/OS V1R12 RMF
Selection ===>

Enter selection number or command on selection line.


  S SYSPLEX         Sysplex reports and Data Index                        (SP)
  1 OVERVIEW        WFEX, SYSINFO, and Detail reports                     (OV)
  2 JOBS            All information about job delays                      (JS)
  3 RESOURCE        Processor, Device, Enqueue, and Storage               (RS)
  4 SUBS            Subsystem information for HSM, JES, and XCF          (SUB)

  U USER            User-written reports (add your own ...)               (US)


                    O OPTIONS    T TUTORIAL    X EXIT

       5694-A01 Copyright IBM Corp. 1986, 2010. All Rights Reserved
                    Licensed Materials - Property of IBM
```

Figure 3-13 RMF Monitor III Primary menu

If a particular job runs slowly, use option 2 to display information about what might be causing a job to be delayed. If it is because of catalog ENQ contention, this information will be in the RMF Job Report Selection report. Selecting option 2 from the RMF Monitor III Primary Menu displays the panel that is shown in Figure 3-14.

```
                RMF Job Report Selection Menu
Selection ===>

Enter selection number or command and jobname for desired job report.

  Jobname ===> MYJOB___

  1 DEVJ          Delay caused by devices                    (DVJ)
 1A DSNJ          .. Data set level                          (DSJ)
  2 ENQJ          Delay caused by ENQ                         (EJ)
  3 HSMJ          Delay caused by HSM                         (HJ)
  4 JESJ          Delay caused by JES                         (JJ)
  5 JOB           Delay caused by primary reason          (DELAYJ)
  6 MNTJ          Delay caused by volume mount               (MTJ)
  7 MSGJ          Delay caused by operator reply             (MSJ)
  8 PROCJ         Delay caused by processor                   (PJ)
  9 QSCJ          Delay caused by QUIESCE via RESET command   (QJ)
 10 STORJ         Delay caused by storage                     (SJ)
 11 XCFJ          Delay caused by XCF                         (XJ)
```

Figure 3-14 RMF III Job Selection menu

Select option 2 to find out whether ENQs are causing the delays for this particular job. If heavy contention for a catalog resource is suspected after using RMF, use D GRS commands to further investigate. You can issue commands to see whether there is contention and, if contention exists, which job is blocking other requesters for a particular resource.

By itself, resource contention is not a sign of a problem. However, contention that is held for a long time by the same resources and requesters might indicate a problem.

GRS provides diagnostic commands to help you determine the source of contention:

- **DISPLAY GRS,CONTENTION**

  Provides an alphabetized list of all visible ENQ resources that are in contention. Each resource is reported with the owners and waiters of the resource.

  The `DISPLAY GRS,CONTENTION` command reports only contention for SCOPE=SYSTEM resources that are allocated on the system where the command was run. It does *not* report contention for SCOPE=SYSTEM resources on other systems in the complex.

- **DISPLAY GRS,ANALYZE,WAITER**

  Provides a list of requesters that have been waiting the longest for ENQ resources. Each waiter is reported with these characteristics:

  – The resource name and scope
  – The count of waiters and blockers of the resource
  – The top blocker of the resource

  Each waiter is reported with its wait time, the system resource that it was ENQed on, and the type of access (shared or exclusive) requested. The counts of waiters and blockers are explicitly returned only when the count is greater than one.

- **`DISPLAY GRS,ANALYZE,BLOCKER`**

  Provides a list of the requesters that have been blocking ENQ resources for the longest time. Each blocker is reported with these characteristics:

  – Resource name and scope.
  – The count of waiters and blockers of the resource.

  The block time, system the blocker ENQed from, job name, and the type of access that was requested are also reported.

- **`DISPLAY GRS,ANALYZE,DEPENDENCY`**

  Provides resource allocation dependency analysis in these configurations:

  a. Starting with each of the longest ENQ waiters, an analysis is run, iteratively chaining from waiter to top blocker. The analysis runs until either a request that is not waiting is found, or a resource allocation deadlock is detected.
  b. Starting with the top blockers of a specified resource, an analysis is run, iteratively chaining from waiter to top blocker. The analysis runs until either a request that is not waiting is found, or a resource allocation deadlock is detected.

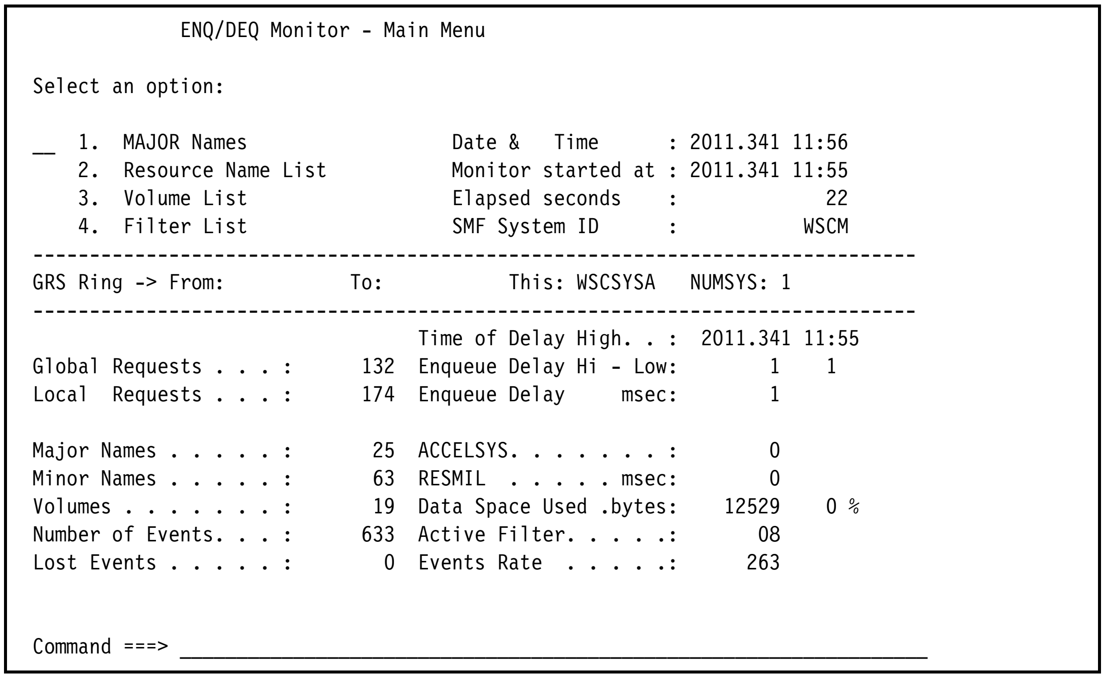
The various forms of the `DISPLAY GRS,ANALYZE` command are available. For more information about how these commands can be used to diagnosis suspected contention problems, see *z/OS MVS Planning: Global Resource,* SA22-7600.

After you determine that catalog contention exists, determine the reason for the contention:

- The length of time it takes to satisfy a catalog request
- An increase in CAS workload
- A combination of both

The GRS ENQ/RESERVE/DEQ Monitor and `F CATALOG,REPORT PERFORMANCE` commands are useful for determining the length of ENQs and how many are being issued. For more information about implementing the ENQ/RESERVE/DEQ Monitor, see Chapter 2 of *z/OS MVS Planning: Global Resource,* SA22-7600.

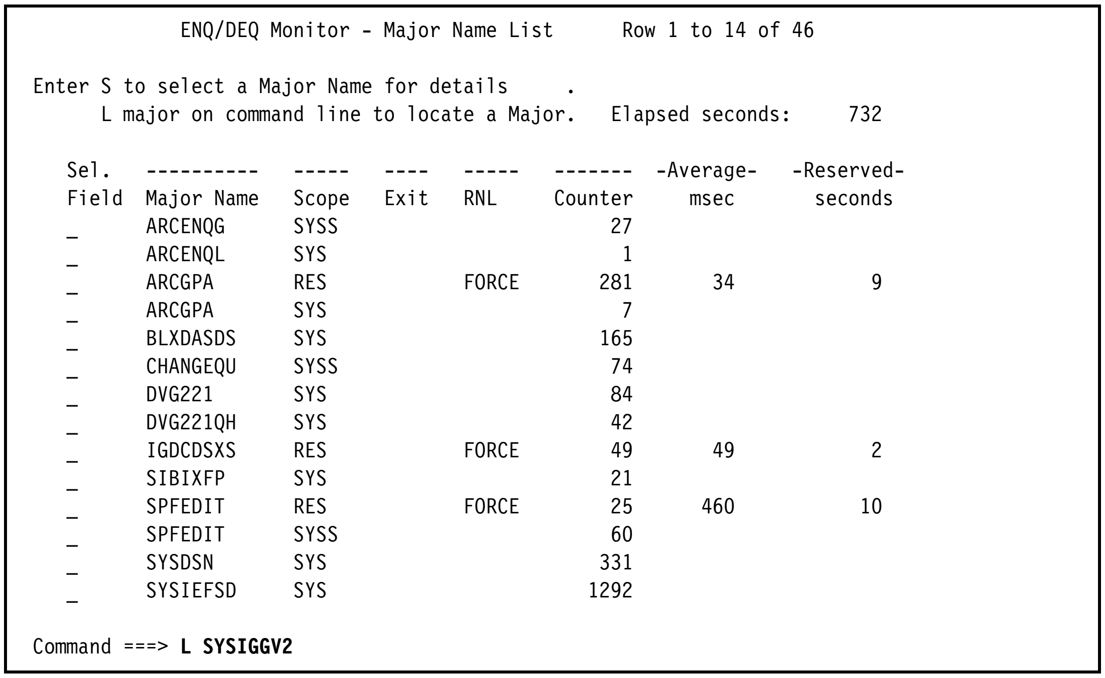
After it is started, the panel that is shown in Figure 3-15 is displayed.



```
             ENQ/DEQ Monitor - Main Menu

Select an option:

__  1.  MAJOR Names               Date &   Time      : 2011.341 11:56
    2.  Resource Name List        Monitor started at : 2011.341 11:55
    3.  Volume List               Elapsed seconds    :             22
    4.  Filter List               SMF System ID      :           WSCM
-------------------------------------------------------------------------------
GRS Ring -> From:          To:          This: WSCSYSA   NUMSYS: 1
-------------------------------------------------------------------------------
                               Time of Delay High. . :  2011.341 11:55
Global Requests . . . :      132  Enqueue Delay Hi - Low:        1    1
Local  Requests . . . :      174  Enqueue Delay     msec:        1

Major Names . . . . . :       25  ACCELSYS. . . . . . . :        0
Minor Names . . . . . :       63  RESMIL  . . . . . msec:        0
Volumes . . . . . . . :       19  Data Space Used .bytes:    12529    0 %
Number of Events. . . :      633  Active Filter. . . . .:       08
Lost Events . . . . . :        0  Events Rate  . . . . .:      263


Command ===> __________________________________________________________________
```

Figure 3-15   ENQ / DEQ Main Menu panel

The Main Menu panel summarizes the active GRS options and activity. Select option 1 to list the ENQ activities by major names, RNL action, and the number of global and local ENQs if global resource serialization is active (Figure 3-16).



```
             ENQ/DEQ Monitor - Major Name List      Row 1 to 14 of 46

Enter S to select a Major Name for details     .
      L major on command line to locate a Major.   Elapsed seconds:     732

   Sel.   ----------   -----   ----   -----   -------  -Average-   -Reserved-
   Field  Major Name   Scope   Exit   RNL     Counter     msec       seconds
   _      ARCENQG      SYSS                        27
   _      ARCENQL      SYS                          1
   _      ARCGPA       RES            FORCE       281       34             9
   _      ARCGPA       SYS                          7
   _      BLXDASDS     SYS                        165
   _      CHANGEQU     SYSS                        74
   _      DVG221       SYS                         84
   _      DVG221QH     SYS                         42
   _      IGDCDSXS     RES            FORCE        49       49             2
   _      SIBIXFP      SYS                         21
   _      SPFEDIT      RES            FORCE        25      460            10
   _      SPFEDIT      SYSS                        60
   _      SYSDSN       SYS                        331
   _      SYSIEFSD     SYS                       1292

Command ===> L SYSIGGV2
```

Figure 3-16   ENQ / DEQ Monitor: Major Name List option

After the Major Name List panel is displayed, use *L SYSIGGV2* to find if there are any ENQs for catalog resources. Select the Major Name SYSIGGV2 as shown in Figure 3-17.

```
ENQ/DEQ Monitor - Major Name List     Row 16 to 29 of 46

Enter S to select a Major Name for details     .
      L major on command line to locate a Major.   Elapsed seconds:     732

   Sel.   ----------   -----   ----   -----   -------  -Average-   -Reserved-
   Field  Major Name   Scope   Exit   RNL     Counter    msec       seconds
   s      SYSIGGV2     RES            FORCE      1362       8           10
   _      SYSIKJBC     SYS                         28
   _      SYSIKJPL     SYS                        431
   _      SYSVSAM      SYSS                        14
   _      SYSVTOC      RES            FORCE        96      27            2
   _      SYSZ#SSI     SYS            NO            5
   _     SYSZALCF     SYS                         3
```

Figure 3-17 ENQ / DEQ Monitor: Locating a major name

After you select major name SYSIGGV2 a minor name list is displayed, as shown in Figure 3-18.

```
ENQ/DEQ Monitor - Minor Name List       Row 1 to 8 of 24

Minor Name list for:              Major Name  : SYSIGGV2
                                  RNL . . . . : FORCED
                                  Scope . . . : RESERVE
                                  Reserved sec: 10       Avg msec: 8

Enter S to select a Minor Name for Jobnames    .
      L min.  on command line to locate a Minor.   Elapsed seconds:      732

  Interval
- --Rate- ----- ------ -----------------------  ------------ Time ------------
S   min.  Count Volume Minor name (max 24 char)  Avg ms  Min ms  Max ms Tot sec
s     0       4 DB2710 CATALOG.DB2710                 6       6       8       0
_     0       3 CICSTS ICFCAT.CICSTS                 24       6      61       0
_     5      56 SMSN01 ICFCAT.IBMBOOKS                6       5      30       0
_   164       1 IMS610 ICFCAT.IMS610                 62      62      62       0
_     1      29 SMS019 ICFCAT.SMSUCAT1                6       6       8       0
_    22     663 SMS011 ICFCAT.SMSUCAT2                8       6     147       5
_    10     257 SMS015 ICFCAT.SMSUCAT3               10       6     121       2
```

Figure 3-18 ENQ / DEQ Monitor: Minor Name List

Selecting a minor name displays the job and program names with an indication of exclusive or shared use as shown in Figure 3-19.

```
ENQ/DEQ Monitor - Jobname List            Row 1 to 1 of 1


List for Major Name  : SYSIGGV2
         Minor Name  : CATALOG.DB2710
         Minor Length: 20
   --------  --------   ----------       -------- ---       ----------
   Job_name  User_ID    Enqs x Job       Pgm_name E/S       Enqs x PGM
   CATALOG   *                   6       IGGPACDV  S                 6
******************************* Bottom of data ********************************
```

Figure 3-19 ENQ / DEQ Monitor: Jobname List

From these displays, you can find which catalogs have high activity and how long it takes to process ENQ serialization requests for these catalogs.

In addition to RMF and the ENQ/RESERVE/DEQ Monitor, the F CATALOG,REPORT,PERFORMANCE command can provide the following data:

- The number of entries in the catalog address space
- The number of serialization requests CAS has issued
- How long these requests take before being granted

Figure 3-20 on page 122 shows part of the output of the command.

This report provides the following information:

- How many entries have been made into CAS
- How many exclusive and shared ENQs have been issued to catalogs and VVDSs since the counters were last reset with F CATALOG,REPORT,PERFORMANCE(RESET)

If the command is issued over a time period, the report reveals workload and ENQ response time trends.

```
IEC359I CATALOG PERFORMANCE REPORT
*CAS***************************************************
*  Statistics since 15:44:22.18 on  05/26/2011       *
*  -----CATALOG EVENT----   --COUNT--  ---AVERAGE---  *
*  Entries to Catalog        35,102      4.947 MSEC  *
*  BCS ENQ Shr Sys           67,777      0.045 MSEC  *
*  BCS ENQ Excl Sys             298      0.259 MSEC  *
*  BCS DEQ                   74,122      0.030 MSEC  *
*  VVDS RESERVE CI           14,370      0.032 MSEC  *
*  VVDS DEQ CI               14,370      0.045 MSEC  *
*  VVDS RESERVE Shr         109,282      0.034 MSEC  *
*  VVDS RESERVE Excl            178      0.085 MSEC  *
*  VVDS DEQ                 109,460      0.042 MSEC  *
*  SPHERE ENQ Excl Sys          201      0.033 MSEC  *
*  SPHERE DEQ                   203      0.030 MSEC  *
*  CAXWA ENQ Shr                  6      0.013 MSEC  *
*  CAXWA DEQ                      6      0.008 MSEC  *
*  VDSPM ENQ                 68,293      0.006 MSEC  *
*  VDSPM DEQ                 68,293      0.004 MSEC  *
*  RPL ENQ                       13      3.327 MSEC  *
*  RPL DEQ                       13      0.026 MSEC  *
*  BCS Get                   42,404      0.072 MSEC  *
*  BCS Put                      105      0.571 MSEC  *
*  BCS Erase                     11      0.529 MSEC  *
*  VVDS I/O                 124,106      0.511 MSEC  *
*  VLF Define Major             109      0.002 MSEC  *
*  VLF Identify                 801      0.002 MSEC  *
*  RMM Tape Exit                  3      0.001 MSEC  *
*  OEM Tape Exit                  3      0.000 MSEC  *
*  BCS Allocate                 150     13.714 MSEC  *
*  BCS Deallocate                 5      2.255 MSEC  *
*  SMF Write                  6,092      0.042 MSEC  *
*  IXLCONN                        2    111.986 MSEC  *
*  IXLCACHE Read                  2      0.031 MSEC  *
*  DCME Write Stats              10      0.066 MSEC  *
*  MVS Allocate                 121     16.886 MSEC  *
*  MVS Deallocate                 5      2.200 MSEC  *
*  Capture UCB                   69      0.009 MSEC  *
*  Uncapture UCB                  4      0.018 MSEC  *
*  SMS Active Config              2      0.458 MSEC  *
*  RACROUTE Auth             28,842      0.069 MSEC  *
*  RACROUTE Define                3      0.103 MSEC  *
*  DADSM Scratch                  2     10.330 MSEC  *
*  DADSM Allocate                 3      7.367 MSEC  *
*  Obtain QuiesceLatch       68,095      0.000 MSEC  *
*  ENQ SYSZPCCB               6,074      0.003 MSEC  *
*  DEQ SYSZPCCB               6,074      0.002 MSEC  *
*  Release QuiesceLatch      68,095      0.000 MSEC  *
*  Capture to Actual          2,941      0.005 MSEC  *
*  ENQ SYSIGGV1                   2      0.009 MSEC  *
*  DEQ SYSIGGV1                   2      0.006 MSEC  *
*  ENDREQ                         3      0.003 MSEC  *
*CAS***************************************************
IEC352I CATALOG ADDRESS SPACE MODIFY COMMAND COMPLETED *
```

Figure 3-20 F CATALOG,REPORT, PERFORMANCE output

Catalog contention summary

After reviewing the RMF reports and the output of the D GRS and the F CATALOG commands, determine whether ENQ response times and CAS workload have increased. If efforts to tune GRS do not result in lower or acceptable ENQ response times, investigate the possibility of splitting the catalog in question. Splitting the catalog might also be needed if the GRS ring is operating as efficiently as possible and the elongation is because of increased CAS activity for a particular catalog.

Use automation to issue F CATALOG commands at specific intervals and saving the output in a data set. If you allocate the output data set with DISP=MOD, the command responses are appended to the existing data set so you have only one data set to analyze (Figure 3-21). Do not use RLSE in the SPACE parameter to preserve the remaining space for the next executions.

```
//MYJOBCARD ...
//*-----------------------------------------
//TSOCMD1 EXEC PGM=IKJEFT01,PARM='CATPERF'
//SYSEXEC  DD  DISP=SHR,DSN=MYPDS.REXX.DATA
//SYSTSPRT DD  SYSOUT=*
//OUTP     DD  DSN=MYOUTPUT.CATALOG.DATA,DISP=MOD,
//   LRECL=134,RECFM=FB,SPACE=(CYL,(2,2))
//SYSTSIN  DD  DUMMY
```

Figure 3-21 JCL to issue the F CATALOG,REPORT,PERFORMANCE batch

You can customize this JCL to start the REXX presented in Example 3-15 and use a scheduler such as OPC, to periodically run and collect catalog performance data.

Example 3-15 CATPERF REXX to issue F CATALOG command

```
/*                   REXX                               */
/* THIS IS A REXX EXEC TO ISSUE F CATALOG,REPORT,PERFORMANCE */
/* COMMANDS AND WRITE THE OUTPUT TO A DATA SET           */
/*                                                        */
/**********************************************************/
SOLD=0
UNSOLD=0
X=MSG("ON")
/*********************************************************/
/*                   REXX                               */
/* THIS WILL CHECK CONSPROF PROFILE AND ENTER CONSOLE    */
/* MODE TO ISSUE F CATALOG,REPORT COMMANDS.              */
/*********************************************************/
X=OUTTRAP('CONDIS.','*')
"CONSPROF"
X=OUTTRAP('OFF')
IF RC > 0 THEN DO
   SAY "ERROR IN CONSPROF."
   EXIT
END
PARSE VAR CONDIS.1 W1 W2 W3 W4 W5
IF SUBSTR(W1,1,3) = 'IKJ' THEN
  DO
      IF RIGHT(W2,4)="YES)" THEN
        DO
          ADDRESS "TSO" "CONSPROF SOLDISPLAY(NO)"
          SOLD=1
        END
      IF RIGHT(W4,4)="YES)" THEN
        DO
```

```
          ADDRESS "TSO" "CONSPROF UNSOLDISPLAY(NO)"
          UNSOLD=1
        END
  END
IF SUBSTR(W1,1,3) <> 'IKJ' THEN
  DO
      IF RIGHT(W2,4)="YES)" THEN
        DO
          ADDRESS "TSO" "CONSPROF SOLDISPLAY(NO)"
          SOLD=1
        END
      IF RIGHT(W4,4)="YES)" THEN
        DO
          ADDRESS "TSO" "CONSPROF UNSOLDISPLAY(NO)"
          UNSOLD=1
        END
  END
X=MSG('OFF')
Z=0
"CONSOLE ACTIVATE"
ADDRESS CONSOLE "F CATALOG,REPORT,PERFORMANCE"
DO WHILE RC = 0
  RC='GETMSG'(MVSCMD.,,,,5)
    IF RC = 0 THEN DO
    I = 1
      DO       I = 1 TO MVSCMD.0
       IF Z = 0 THEN
          DO
             DATE = MVSCMD.MDBGDSTP
             TIME = MVSCMD.MDBGTIMH
             QUEUE "DATE: "||DATE||"   TIME: "||TIME
           END
       QUEUE MVSCMD.I
       Z = 1
      END
    END
END
DO I = 1 TO QUEUED()
   ADDRESS MVS 'EXECIO 1 DISKW OUTP'
END
ADDRESS MVS 'EXECIO 0 DISKW OUTP (FINIS'
X=MSG('ON')
"CONSOLE DEACTIVATE"
IF SOLD=1 THEN ADDRESS "TSO" "CONSPROF SOLDISPLAY(YES)"
IF UNSOLD=1 THEN ADDRESS "TSO" "CONSPROF UNSOLDISPLAY(YES)"
EXIT
```

3.4.13 Monitoring the catalog address space

You can issue F CATALOG,REPORT command to monitor and obtain general information about CAS. This information can be used to evaluate your current catalog environment setup. If you determine your current setup is inadequate, you can change it with another F CATALOG command, or by changing the SYSCAT*xx* member of SYS1.NUCLEUS.

Figure 3-22 shows an output of this command. Consider the number of service tasks available to process catalog requests. Make sure that the upper limit set is adequate, considering the highest number of service task values.

The alias table available entry indicates whether there is a problem with the catalog alias table. This field should always say YES. If it says NO, try restarting the catalog address space. Performance is affected if catalog management does not have catalog aliases in the catalog alias table.

Remember: The alias table might not have been created if there was not enough virtual storage available during system initialization. Other messages are issued to indicate this problem. If restarting CAS fails to create the alias table, determine why there is not enough virtual storage, and make the necessary changes to your system.

```
F CATALOG,REPORT
IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
IEC359I CATALOG REPORT OUTPUT
*CAS*************************************************************
*  CATALOG COMPONENT LEVEL   = HDZ1C10                          *
*  CATALOG ADDRESS SPACE ASN = 0037                             *
*  SERVICE TASK UPPER LIMIT  =  180                             *
*  SERVICE TASK LOWER LIMIT  =   60                             *
*  HIGHEST # SERVICE TASKS   =   25                             *
*  # ATTACHED SERVICE TASKS  =   25                             *
*  MAXIMUM # OPEN CATALOGS   = 1,024                            *
*  ALIAS TABLE AVAILABLE     = YES                              *
*  ALIAS LEVELS SPECIFIED    = 1                                *
*  SYS% TO SYS1 CONVERSION   = OFF                              *
*  CAS MOTHER TASK           = 007FF368                         *
*  CAS MODIFY TASK           = 007A2DC8                         *
*  CAS ANALYSIS TASK         = 007A2968                         *
*  CAS ALLOCATION TASK       = 007A2B98                         *
*  CAS ECS TASK              = 007A2738                         *
*  VOLCAT HI-LEVEL QUALIFIER = SYS1                             *
*  NOTIFY EXTENT             =   80%                            *
*  DEFAULT VVDS SPACE        = ( 11, 11) TRKS                   *
*  CONTENTION SYSZTIOT TIME  =    10                            *
*  ENABLED FEATURES          = DSNCHECK DELFORCEWNG SYMREC      *
*  ENABLED FEATURES          = UPDTFAIL                         *
*  DISABLED FEATURES         = VVRCHECK AUTOTUNING BCSCHECK     *
*  INTERCEPTS                = (NONE)                           *
*CAS*************************************************************
IEC352I CATALOG ADDRESS SPACE MODIFY COMMAND COMPLETED
```

Figure 3-22 F CATALOG, REPORT

4

VSAM performance

This chapter addresses you can get most out of your VSAM data sets. It describes the VSAM functions that can enhance performance. Hints and tips are provided to help you implement these VSAM functions. However, the specific performance aspects of VSAM RLS, Catalog Management, and Transactional VSAM (TVS) are addressed in their respective chapters.

This chapter includes the following sections:

- Performance analysis
- VSAM performance management
- VSAM defaults and rule-of-thumbs
- Parameters affecting performance
- VSAM performance by scenarios
- Performance monitors

4.1 Performance analysis

Performance analysis is a discipline that enforces the performance goals as defined in the service level agreement (SLA) in your IT systems. Performance analysis consists of two components:

- Performance Administration
- Performance Management

In a data processing environment, the following disciplines can cause you to experience poor performance:

- Continuous availability (24 by 7)
- Security
- Integrity
- Functions that are related to disaster recovery
- Accounting data gathering
- Performance data gathering
- Compatibility at application level

The challenge is to maintain good performance while still fulfilling all these disciplines.

4.1.1 Service level agreement (SLA)

The human view of the performance of a system is often subjective, emotional, and difficult to manage. However, meeting the business needs of the business users is the reason that the system exists. To make the subject of performance more objective and related to specific business needs, the SLA was introduced.

The SLA is an agreement between Information Systems organization and user departments that describe these goals:

- Average transaction response time: total and optionally its components: network, I/O, CPU. It is becoming popular to also have a SLA for DASD I/O response time. Generally, correlate a response time SLA figure with a maximum transaction rate. For example, for less than 1000 transactions per second, ensure that the average response time is equal to one second.
- The distribution of these response times, a measurement about how erratic in relation to the average, they are. For example, 90% TSO trivial normally takes less than 0.2 of a second.
- System availability, which is the percentage of time that the system is available to the user. For example, ensure 99.5% of the time the required application (and its data) is available to be used.

There is a controversy if the throughput should be included in the SLA. The throughput is also called external throughput rate (ETR), and is measured in ended transactions per second of elapsed time (not CPU time). Generally, include the throughput as a qualifier of the average transaction response time. Sometimes, a high value of ETR might be a justification of a deviation of the average response time SLA figure. However, do not use the ETR as a metric by itself in SLA. The reason is that ETR varies with the number of users who send transactions (the workload), and also varies with the average user thinking time. Those variables cannot be controlled by the Information System organization in your company. For more information, see 4.1.2, “Transaction response time components” on page 130.

SLA is a fundamental concept for performance analysis and capacity planning disciplines, as shown in Figure 4-1. Another way of looking at SLA is to say that, SLA is the threshold at which the user has the right to complain.

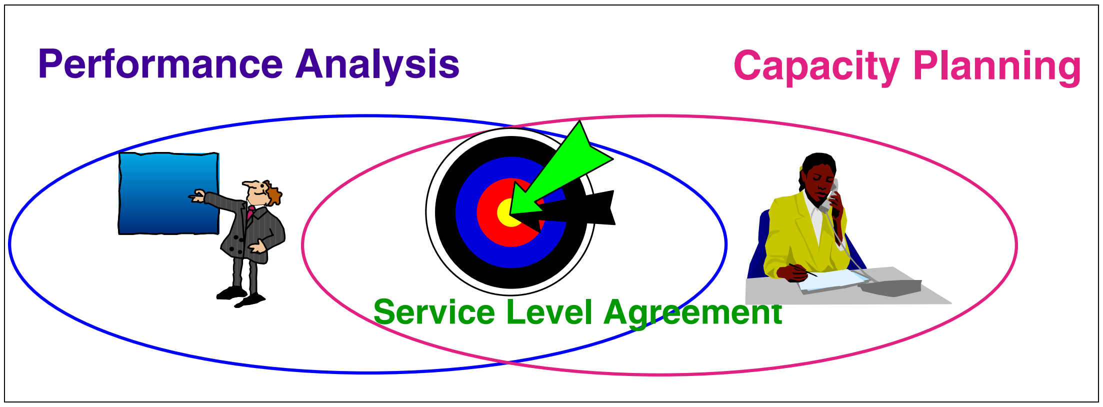


Figure 4-1 Disciplines that use SLA

Performance administration

Performance administration is one of the two components of performance analysis. It is run by the service level administrator with the objective to define the rank of goals by importance of the transactions. It is run later to analyze the reports to verify the performance results. The service level administrator is responsible for defining the installation's performance goals, which are based on business needs. This explicit definition of workloads and performance goals is called a service definition.

Performance management

Performance management is the other component of performance analysis. It is the activity in an installation that monitors and allocates data processing resources to transactions according to a SLA. PM has the following tasks:

- Management targeting allocation of data processing resources (processor, I/O, and storage) to transactions according to their SLA. The expression “allocating resources” implies determining transaction priority in queues that access such resources. This priority determines for how long transactions stay in such queues (the waiting time).
- Monitoring is the task of dynamically verifying whether objectives of the management task are reached.

  Performance management is run by the WLM z/OS software component or by system administrators, depending on the type of resource. For example, the processor resources are covered by WLM, and data set contention by system administrators.

Generally, there are three main ways to solve performance management problems, that is, conflicts between reality and goals as stated in the SLA:

- Buy: You can simply buy more resources (done by the installation).
- Tune: Tuning your system makes more effective and efficient use of those resources (done by installation or by WLM Resource Adjustment routine).
- Steal: You can “steal” the resources from a less critical transaction and delivering them to key transactions. This stealing can be done by modifying priorities in the resource queues. The resource allocation can be the WLM Policy Adjustment routine (at 10-second intervals) or the installation system programmers.

If none of these options are technically or financially possible, you must change the target expectations. You might complete an extensive performance analysis only to conclude that no further tuning or stealing of resources can be done. One of goals of this chapter is to assist you in determining whether you have reached such a turning point.

4.1.2 Transaction response time components

A transaction is a unit of work that is produced by an online or batch interaction with a user or a department within the system. It can be a CICS, TSO, a WEB, distributed DB2, or even a job batch interaction. The type of the transaction depends on the transaction manager software that receives the transaction. All transactions are monitored and accounted for by Workload Manager (WLM), a z/OS component in charge of the performance management tasks.

Transaction response time metrics

The two major metrics that characterize the performance of a transaction are average response time (related with quality) and ETR (related with quantity). This section is mostly concerned with the average response time.

To better analyze a transaction average response time, you must dissect its different components. A transaction response time consists of these components:

- Transmission time, when the transaction (contents of a window modified by the user) is traveling in the network
- Other servers time, when processing in an intermediate server as such firewalls, DNS, and MQ
- Host time, when processing in the mainframe

Host response time

This book only cover the host response time, which is called *response time* for short. The transaction response time consists of these components:

```
Tr = Ts + Tw
```

Ts is the service time, Tw is the waiting time (also called queue time), and Tr is the total response time.

Figure 4-2 shows the transaction response time components. The explanation of the values Ts and Tw follow the figure.

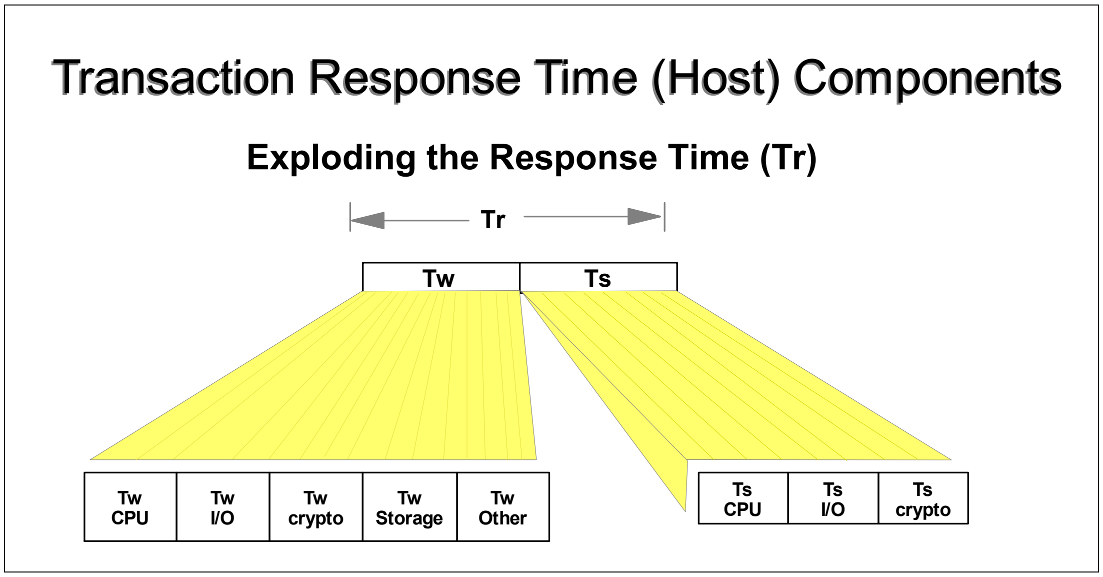

Transaction Response Time (Host) Components
Exploding the Response Time (Tr)
Tr
Tw
Ts
Tw CPU
Tw I/O
Tw crypto
Tw Storage
Tw Other
Ts CPU
Ts I/O
Ts crypto

Figure 4-2 Response time components

Tw CPU):	Time with ready dispatchable units, but not enough dispatching priority. Included are Tw(CPU), Tw(zAAP), and Tw(zIIP)
Tw (I/O):	Time with I/Os delayed in UCB (IOSQ time) or channel subsystem- SAP (Pending time)
Tw (crypto):	Time in the asynch (Crypto Express3) crypto outbound queues
Tw (Storage):	Time that is lost from a page fault or being swapped-out by a WLM decision
Tw (Other):	Time that is delayed by operator, ENQ, data set recall, and server AS service.
Ts CPU):	Time with dispatchable units that run programs in the PU. Included are Ts(CPU), Ts(zAAP), and Ts(zIIP).
Ts (I/O):	Time with I/Os being run (connected or disconnected times)
Ts (crypto):	Time being served in asynchronously (Crypto Express3) in crypto outbound hardware response time components

4.1.3 External throughput rate

External throughput rate (ETR) measures number of ended transactions per elapsed time, or the throughput of the system, as shown in the following formula:

```
ETR = Number of transactions / Elapsed time
```

Normally, when ETR has a high value (which is good), the average response time tends to have a high value (which is bad). There is also a formula (derived from Little's law) relating Tr with ETR:

```
ETR = N / (Tt + Tr)
```

Where:

- N: Average number of active users (logged on)
- Tt: Average thinking time of these users
- Tr: Average host response time

This formula has the following considerations:

- The variables that more intensively affect the ETR are N and Tt because of their usually high numeric values. Therefore, SLA based on ETR are difficult to recognize because only variable that the IT department can directly manage (through a tuning staff) is Tr. For same reason, there is not an ETR type of goal in WLM.
- You can use the formula for deriving the user average thinking time and use it to challenge the policies of a user department. This formula can be used to prove the effectiveness of thin client (desk tops without data and programs) design to decrease the average user thinking time.
- Any internal or external bottleneck in the system affects the ETR. Examples include I/O constraints, tape mount delay, and paging delay.
- In a top of line IBM z10™, you can reach more than 1.3-G transactions per day (including commercial and infrastructure).

Transaction response time is the best way to characterize the performance of a transaction. The goal is to reduce its value and so increase the ETR figures (when the value is below one second).

4.1.4 I/O performance

As processor speed increases, the I/O response time Tr (I/O) becomes the determining factor in the average transaction response time, as shown in the formula:

```
Tr (I/O) = Ts (I/O) + Tw (I/O)
```

Obviously, you can get excellent transaction response time by reducing I/O wait time and I/O service time.

Generally speaking, you can reduce the average I/O response time (Tr (I/O)) by using software and hardware techniques. The software techniques aim to decrease the number of I/O operations, by implementing these techniques:

- Address space buffers (below the line, between the line and the bar, and above the bar) and data space buffers.
- IBM Hiperspace™ buffers (not recommended since the first z/OS release)
- Data compression (done by hardware but activated by the z/OS component CMF)
- Data striping

You can also use the following hardware techniques:

- Faster channels such as FICON Express8
- Faster device paths (host adapters) into the controllers and switches
- Larger controller cache
- More DASD subsystem concurrency, such as Parallel Access Volumes (PAV) in IBM DS8000®.
- Faster disks through high rotations per minute (RPMs), small size, RAID-10, and solid state

However, experience has shown that when you remove an I/O bottleneck (for example, by implementing data striping), the bottleneck is moved from the I/O subsystem to the processor. However, your transaction throughput is better than before.

4.2 VSAM performance management

The goal of VSAM performance management is to decrease the values of I/O wait time (Tw(I/O)) and I/O service time (Ts(I/O)) for the transactions that access VSAM data. This goal can be accomplished by avoiding I/O, performing it faster, and adjusting I/O priorities. These priorities must be closely linked to your business needs through WLM goals.

You can use two approaches to help you to improve VSAM performance. First, you can adjust VSAM external parameters whose values might affect performance. The guidelines provided about how those parameters should be set are based on experience. This is known as "rule-of-thumb" (ROT) mode. The guidelines are provided at the end of each topic.

In the second approach, a scenario is used to simulate VSAM data sets causing performance problems to key transactions in a real installation. A methodology is then described to fix the situation. Some of the guidelines that are covered in the rules of thumb are presented again, now connected with a specific VSAM behavior.

This section also addresses other factors that are not connected to VSAM parameters. These factors include I/O configuration delays, use of FICON channels, SMS storage class attributes, use of Hiperbatch, and workload processor bound.

The figures that show RMF reports covering VSAM I/O measurements that are related to the theories presented were generated in the ITSO lab.

4.3 VSAM defaults and rule-of-thumbs

The defaults for the VSAM cluster parameters that affect performance were established when virtual storage was all below the 16 M address line. At that time, central storage size was restricted, the DASD was slow, removable, and expensive, and channels were a scarce resource.

VSAM defaults can become outdated as new code and hardware are developed. Defaults are not changed to maintain compatibility with your existing workload. Compatibility is an important feature that protects your investment in your environment. You do not need to throw out your code and hardware when z/OS changes a release. However, keep in mind that these VSAM defaults can be easily overwritten through JCL, the ACB macro, data class constructs, ACS routines, and IDCAMS.

Rules-of-thumbs (ROT) are the technique that you use when you do not master the theory behind the function you are tuning or installing. However, ROT can change with technology enhancements without notice. In other words, a ROT becomes obsolescent. These obsolete ROTS are called "invalid rules-of-thumb" (IROTs). The IROTs applying to I/O become out of date because of these factors, among others:

- The physical 3390 volume concept no longer exists, and you do not know where their logical tracks are on the physical disks. Therefore, you do not need to care about 3390 seek arm movement considerations in such devices.

  Also, several changes were made in the device geometry as new products were introduced. The current logical 3390 geometry will be valid for a long time because it is not affected by enhancements to the disk controllers. For example, the device independence recommendation (records instead of tracks for units of allocation) is not a consideration anymore because of changes in future track or cylinder geometry.
- During a cache miss, the channel always reads (or writes) in cache, asynchronously in relation to the real disk, so there are no extra revolutions. caused by 3390 RPS misses.

- The existence of the Define Extent Format CCW, the Prefix CCW, and parallel access volume (PAV) for the DS8000.
- The introduction of the RAID controllers
- Enhancements to channel programs, such as MIDAW and High Performance FICON for IBM System z®.

The following recommendations are *no longer valid*:

- Place your VTOC around the middle of the volume (or at 1/3 as said by the statistics purists).
- If you must place two active data sets in the same 3390 volume, place them close to each other, to avoid arm stealing along seeks.
- Do not allow much secondary allocation in the same volume because of the embedded long seeks in the same data set.
- Use VSAM KSDS embedded sequence set index records to minimize seeks.
- Use VSAM KSDS replication to avoid unnecessary revolutions.
- When you are allocating a data set on device-dependent geometry, use cylinders, not tracks, for better performance.
- It is better to use a device independent geometry for allocation units (for example, records) to avoid modifications when the 3390 geometry changes.
- Reorganize your KSDS data set after some CA splits to avoid long embedded seeks. For more information about VSAM KSDS data set reorganization, see 2.1, “Reorganization considerations” on page 46.
- Avoid channel utilization above 30% to inhibit RPS misses and another disk revolution
- Use the VSAM keyrange option so that you have control over the allocation of the key ranges of your data set.
- Avoid many active data sets in the same 3390 volume.

4.4 Parameters affecting performance

The following VSAM parameters and options can affect performance.

4.4.1 Allocation units

When you define your new data set, either you or the SMS administrator must specify the amount of space to be allocated for it by defining a data class (DC). One of the allocation functions for a new data sets is the creation of the DSCB in the VTOC, by the DFSMS DADSM routine. For more information about allocation, see 1.7.2, “Allocating a VSAM data set” on page 38. If SMS is active, you can specify a data class and take advantage of the space allocation number that was set by your storage administrator. If you choose to specify space explicitly, you can specify it for VSAM data sets in several types of units. You can ask for number of logical records, kilobytes, megabytes, tracks, or cylinders. When cylinder and track are referred to, it refers to the logical 3390 cylinder and track, not the cylinder and track of the disks in the RAID DASD controllers.

Another comment applies to extended address volumes (EAV) support. For cylinder-managed space, the space is obtained in 21 cylinder (or 315 tracks) chunks, also called multicylinder unit (MCU). In addition, remember that the allocated space must be a multiple of the control area size.

DADSM only accepts allocation requests in tracks or cylinders. The units that are specified (records, kilobytes, or megabytes) are then converted by IDCAMS (a VSAM utility program) before the DADSM creation request.

If you specify records as the allocation unit, the number of records you declare is multiplied by the value of the keyword RECORDSIZE(AVERAGE) value (defined at ACB control block). This process derives the space in bytes. This keyword can also indicate the maximum value that is allowed for the record length. If the maximum record length is exceeded, VSAM rejects the new record.

When the primary amount on the first volume is used up, a secondary amount is allocated on that volume by the end-of-volume (EOV) routine. VSAM acquires space in increments of control areas (CAs). Each time a new logical record does not fit in the allocated space, EOV allocates more space in the secondary space amount. This process repeats until the volume is out of space. Depending on a DC parameter, there might not be a limit for the number of extents for a VSAM data set. Depending on the type and amount of data set allocation requests, a new volume can be used.

Do not define a small amount of secondary space allocation value, especially for a KSDS or a VRRDS data set. Many I/O operations are involved when the secondary allocation takes place.

There are no performance recommendations on allocation because these depend on the use of the Guaranteed Space option.

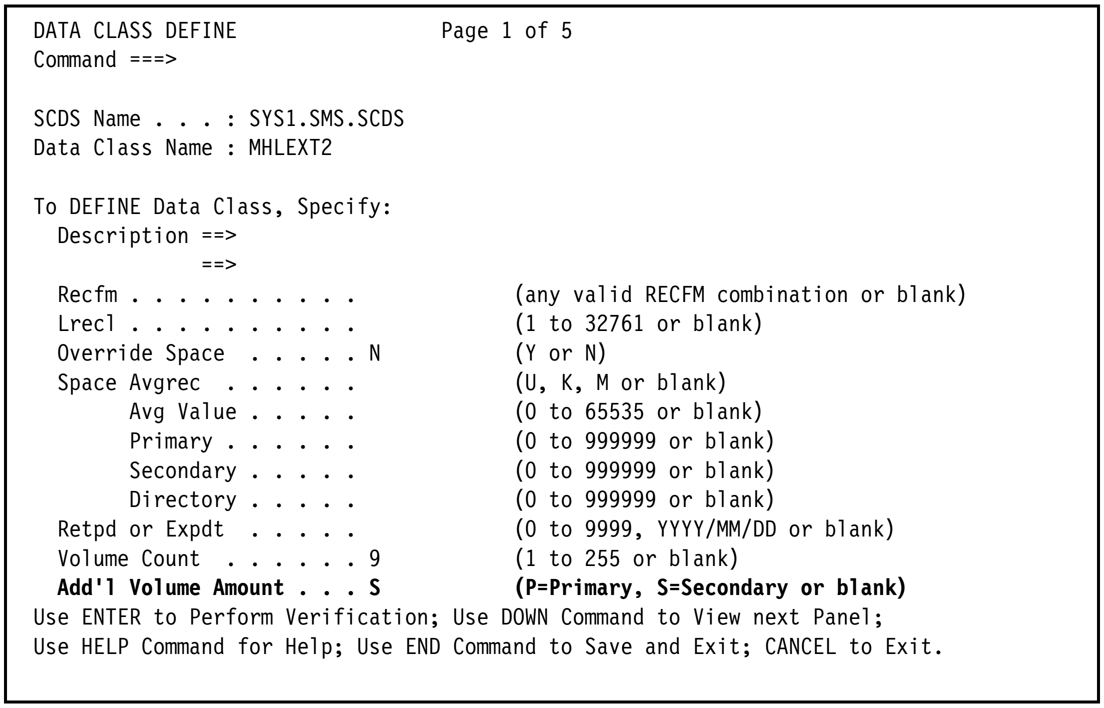
Guaranteed Space

The allocation function depends on whether the data set has the Guaranteed Space attribute.

The Guaranteed Space is a storage class SMS attribute. It is used to reserve space on specific volumes (VOLSER) for clusters that require special placement to meet performance or availability requirements. For example, IBM IMS™ logs that are duplexed by IMS in other data set to improve availability should be on separate volumes. The specified candidate volumes must be described in the storage group that is associated with the cluster.

Use storage class performance keywords, availability attributes at management class, and storage group assignments to determine where to place your data set. All these constructs are selected by your installation ACS routines. SMS then selects volumes by evaluating each volume candidate's ability to satisfy the performance, availability, and space requirements for the data set. To place data sets on specific volumes, use the VOLSER parameter at allocation (through a DD card, for example). Assign a storage class to the data set that supports Guaranteed Space and maps to the correct storage group. If the resulting storage group does not contain the specified volume, or all the volumes are not in the same storage group, the allocation fails.

For non-guaranteed space data set allocation, when you allocate space, you can specify whether to use primary or instead the secondary allocation amounts when extending to a new volume. Do this process with an SMS DATACLASS parameter for VSAM attributes of the ISMF application as shown in Figure 4-3.



```
DATA CLASS DEFINE                Page 1 of 5
Command ===>

SCDS Name . . . : SYS1.SMS.SCDS
Data Class Name : MHLEXT2

To DEFINE Data Class, Specify:
  Description ==>
              ==>
  Recfm . . . . . . . . . .            (any valid RECFM combination or blank)
  Lrecl . . . . . . . . . .            (1 to 32761 or blank)
  Override Space  . . . . . N          (Y or N)
  Space Avgrec  . . . . . .            (U, K, M or blank)
        Avg Value . . . . .            (0 to 65535 or blank)
        Primary . . . . . .            (0 to 999999 or blank)
        Secondary . . . . .            (0 to 999999 or blank)
        Directory . . . . .            (0 to 999999 or blank)
  Retpd or Expdt  . . . . .            (0 to 9999, YYYY/MM/DD or blank)
  Volume Count  . . . . . . 9          (1 to 255 or blank)
  Add'l Volume Amount . . . S          (P=Primary, S=Secondary or blank)
Use ENTER to Perform Verification; Use DOWN Command to View next Panel;
Use HELP Command for Help; Use END Command to Save and Exit; CANCEL to Exit.
```

Figure 4-3 Data Class panel additional volume space allocation

For a Guaranteed Space data set allocation, you must meet the following conditions:

- All volumes that are specified in VOLSER must belong to the same storage group.
- The storage group to which these volumes belong must be in the list of storage groups that are selected by the ACS routines for this allocation.

Generally, allocate space at the cluster data component level because you can size this component and leave the sizing of the index component to VSAM. VSAM allocates space for the cluster by using these guidelines:

- If space is specified at the cluster or alternate index level, the amount that is needed for the index is subtracted from the specified amount. The remainder of the specified amount is assigned to data. For more information, see 1.4.8, “Alternate indexes” on page 15.
- If space is specified at the data component level only, the specified amount is assigned to data. The amount that is needed for the index is in addition to the specified amount.
- If allocation is specified at both the data and index levels, the specified data amount is assigned to data. The specified index amount is then assigned to the index.
- If secondary allocation is specified at the data component level, secondary allocation must be specified at the index level or the cluster level.

Guidelines:

- For KSDS and VRRDS, formulate space for data component only.
- Use the most space information that you can from data class definition.
- Avoid Guaranteed Space if possible.
- Avoid using tracks as a unit to get rid of the contiguous space requirement. Use records because this unit is closer to the application needs.
- Do not use a small secondary allocation figure.
- Consider using "Allocation constraint relief" on page 138.

4.4.2 Optimizing control area (CA) size

In most data components, the size of the data control area is the size of the 3390 cylinder, which is 15 tracks. There is not an installation keyword, where the size of a data or index component CA is defined. The primary and secondary space allocation amounts determine the data CA sizes:

- If either the primary or secondary allocation is smaller than one cylinder, the smaller of the two size values is used as the CA size:
  - TRACKS(100,3): This results in a 3-track CA size.
  - TRACKS(3,100): This results in a 3-track CA size.
  - KILOBYTES(100,50): The system determines the control area based on 50 KB, resulting in a 1-track CA size.
  - RECORDS(2000,5): Assuming that 10 records would fit on a track, this results in a 1-track CA size.
- If both the primary and secondary allocations are equal to or larger than one cylinder, the CA size is one cylinder. For data striping, however, the CA size can be 16 tracks (one more than the cylinder), which is the maximum size for a CA.
  - CYLINDERS(5,10): This results in a 1-cylinder CA size.

For KSDS and VRRDS organizations, the index CI size and buffer space can also affect the CA size. The examples assume that the index CI is large enough to handle all the data CIs in the CA. They also assume that the buffer space is large enough to not to affect the CI size.

A spanned record cannot be larger than a CA minus the control information size (10 bytes per CI for fixed logical records length). Therefore, do not specify large spanned records and a small primary or secondary allocation that results in a CA not large enough to contain the largest spanned record.

CA size has significant performance implications. One-cylinder CAs have the following advantages:

- There is a smaller probability of CA splits.
- The index is more consolidated. One index CI addresses all the data CIs in a CA. If the CA is large, fewer index records and index levels are required. For sequential access, a large CA decreases the number of reads of index records.
- There are fewer sequence set index CIs because there are fewer CAs. In sequential access, all those records must be read. Fewer sequence set index CIs means more effective channel programs.

- If you allocate enough buffers, a large CA allows you to read more buffers into storage at one time. A large CA is useful if you are accessing records sequentially.
- The I/O operations are always scheduled within CA boundaries. The overlap between I/O and processor for sequential processing is then done in a CA boundary. When this boundary is reached, the application must wait until the last input/output to the CA is done before it proceeds to the next CA.

The following disadvantages of a one-cylinder CA must also be considered:

- If there is a CA split, more data is moved.
- During sequential I/O, a large CA (causing many CIs in the same channel program) ties up more real storage and more buffers.

Guideline: Always use CAs of one 3390 cylinder size.

4.4.3 Partial release

Partial release is used to release data (not index) unused space from the end of an *extended format* data set. This process is done when the data set is closed. Partial release is specified through the SMS management class or by the DD RLSE sub parameter.

All space after the high used RBA (HURBA) is released on a CA boundary up to the high allocated RBA (HARBA). If the high used RBA is not on a CA boundary, the high used amount is rounded to the next CA boundary. Partial release has these restrictions:

- Alternate indexes (AIXs) opened for path or upgrade processing are not eligible for partial release. However, the data component of an AIX, when opened as cluster, can be eligible for partial release.
- Partial release processing is not supported for temporary close. Temporary close (TYPE=T) means that the ACB is still opened, so there is no need for an Open to restart processing. However, the data set looks closed with EOF marks at the end.
- Partial release processing is not supported for data sets defined with guaranteed space.
- Extended format is a requirement.

The positive aspect of Partial Release is clearly to save DASD space. The drawback is whether the data set is extended, it must go through secondary space allocation again.

Since DFSMS 1.12, you can perform partial release of a data set, when the HURBA points to a 3390 volume and the HARBA points to another. In this case, full extents can be scratched from VTOC.

Using Extended Address Volume (EAV) can affect partial release function if the data set is at the cylinder allocation area (above 64 K cylinders). The reason is that this data set must have a size multiple of 21 cylinders (MCU).

Guideline: Users trend to exaggerate the amount of required DASD space. Use the Partial Release function.

4.4.4 Allocation constraint relief

Users occasionally encounter data set primary allocation failures or extension failures (X37 abends) because there is not enough space available on a 3390 volume to satisfy the request. SMS alleviates this situation to by running the volume selection again and checking all candidate volumes before failing an allocation.

You can also use the Space Constraint Relief and Reduce Space Up To (%) attributes to request that an allocation be tried again if it fails because of space constraints. For more information, see 2.6.4, “SMS constructs” on page 75.

Guidelines:

- Do not define a small amount of secondary space allocation for a KSDS or VRRDS data set.
- Allocate space at the cluster or data levels.
- Do not use tracks as an allocation unit. Doing so forces the CONTIG attribute for secondary allocations.
- Avoid either primary or secondary allocation smaller than one cylinder because it makes the CA size less than one cylinder.
- Consider partial release, if applicable.
- Consider constraint relief, if applicable.

4.4.5 Control interval size

There are two types of control intervals (CIs) in a KSDS and VRRDS: The index and the data. The other VSAM organizations have only data CIs. The data CI and index CI sizes are declared in the IDCAMS DEFINE command, and kept in the catalog. If you define CI size for the cluster, this value is used for both the data and index CIs. This section addresses how to size both.

Data control interval size

There are arguments that favor both large and small CI sizes:

- For sequential processing, larger data CI sizes are desirable. For example, given a 16-KB data buffer space to be fulfilled, it is faster to read two 8-KB CIs with one I/O operation than four 4-KB CIs with one operation.
- A large CI is a more centralized management of the free space CI, and so causes fewer CI splits. One component with one 16 KB CI with 20% of free space has fewer splits than a two 8 KB CIs with the same free space.
- For direct processing, smaller data CIs are desirable because the application needs to retrieve only one logical record at a time. This process decreases the amount of useless data transfer.
- For data set shared access within an address space, the size of the data CI affects the amount of data that are locked by VSAM. In this case, smaller, is better. For more information, see 2.2.5, “Sharing data in a single VSAM control block structure” on page 58. In this type of sharing, the granularity of the locked resource is the CI. Several performance problems were reported where a large CI is locked because of one logical record in a CI with many logical records. One solution is to convert the buffering mode to LSR, where the serialization granularity is at the logical record level.

Guideline: For data sets accessed both randomly and sequentially, a small data CI with multiple buffers to help for sequential processing can improve performance.

Index control interval size

Usually you do not specify index CI size for KSDS and VRRDS. Allow VSAM to calculate it. Recall that VSAM uses one index record per data CI and one index CI per data CA. The size of the index CI affects these aspects:

- The number of levels in the index set. However, the number of indexes affects the performance only for large clusters that are greater than four levels. Up to four levels, you can buffer all the index CIs.
- The usable capacity of a data CA. Sometimes a small index CI size can cause some loss in the usable capacity of the associated data CA. For the calculation of the CI index size, VSAM assumes a compression ratio of 0.3. However, for some keys this is an underestimate. To see this modification and determine whether your CI size is too small and therefore causing loss of data CA space, see "Index CI size calculation algorithm" on page 50.

Guidelines:

- For sequential access, define data CIs with 16 KB or larger, up to 32 KB (the CI maximum size).
- For random access, define data CIs with 4 KB.
- For mixed random and sequential access, define data CIs with 4 KB to favor random access. In addition, define plenty of buffer space in the buffer pool, allowing CCW chaining for sequential access.
- Allow VSAM to derive the size of the index CI. Define it yourself only when you are losing data CA capacity because of small CI index size.

4.4.6 FREESPACE definition for KSDS and VRRDS

The FREESPACE option specifies, through IDCAMS DEFINE, the percentage of each data CI and each data CA is to be set aside as free space. This process occurs when the cluster KSDS or VRRDS is initially loaded. This percentage is also set aside when a mass insertion (writes skip sequential) is done after the initial load. It is stored in the catalog. You can change the amount of free space by using the IDCAMS ALTER command. Free space is specified as a percentage. If the FREESPACE amount is altered after the data set is initially loaded and sequential insert processing is used, the allocation of free space is not maintained.

The syntax of the FREESPACE parameter is:

```
FREESPACE(CI-percent CA-percent)
```

There is no free space external specification for index CIs or index CAs. FREESPACE is used to reduce the number of CI and CA splits along insertions, and updating in-place records with a length increase.

When you specify free space in CI (the first value in FREESPACE CI percentage), ensure that the CI free space percentage does not result in lots of unused free space in the data CI. You can ensure this by taking into account the logical record length, the size of the CI, and the length of CIDF and RDFs. Following is a numeric example:

- Assume a fixed-length record data set where each CI is 4096 bytes, and 10 bytes are reserved for control information (2 RDFs and 1 CIDF). Each logical record has 1000 bytes and you want to reserve room for 10% inclusion.
- If you specify 10% of CI free space, VSAM reserves 410 bytes (4096 * 0.10) for free space. The logical record space is 3676 (4096 - 420) bytes.

- Because the records loaded in the data set are 1000-byte records, there is only space for three records, leaving 1086 (410 + 676) for insertions. In this free space you can fit another logical 1000-byte record, so your free space setting is correct. You are losing only 86 bytes of unused space. All this calculation is done for a non-compressed data set. For compression information, see 4.4.16, “Data compression” on page 170.

For CA free space, VSAM ensures that at least one CI per CA remains empty during loading when you declare a FREESPACE CA percentage amount other than zero.

Too much free space in a CI or CA can result in these problems:

- Increased number of index levels because of less data per data CI/CA, which affects run times for direct processing slightly.
- More DASD storage is required to contain the same data set
- More I/O operations are required to sequentially process the same number of records. These extra I/O operations are only affected by an excess of CI free space. CA free space does not increase the number of I/O operations in a sequential read because totally free CIs are not moved to storage.

Too little free space can result in an excessive number of CI and CA splits, depending on the key pattern of the records to be inserted. Having too little space can have these consequences:

- The CA splits use many resource because approximately half of the CIs from the CA are moved to another CA at the end of the occupied part of data set (beyond HURBA). Also, to ensure the integrity of the data set, an ENQ SYSVSAM exclusive is issued that might block the access to the data set.
- CI and CA splits can also affect the sequential processing because the DASD controller, to better use the cache, detects only 3390 tracks physical sequence. Splits make the logical sequence different from the physical sequence.

If you know the pattern of the keys that are being inserted, you can select the technique that is best suited for the application.

Guidelines: Determine the amount of CI free space based on the percentage of record additions that you expect, and their distribution:

- *No additions*. If no records will be added and if record sizes will not be changed, there is no need for free space.
- *Few additions*. If few records will be added to the data set, consider a free space specification of (0 0). When records are added, new CAs are created to provide room for more insertions. This guideline is the same for both even and uneven key distributions.
- *Evenly distributed additions*. If new many records will be evenly distributed throughout the data set, set CA free space to the percentage of records to be added to the data set after the data set is loaded. Use the command FSPC (0 nn), where nn equals the percentage of records to be added.
- *Unevenly distributed additions***.** If new records will be key unevenly distributed throughout the data set, do not reserve free space by specifying (0,0). Additional splits, after the first, in that part of the data set with the most growth will produce CIs with only a small amount of unneeded free space.
- *Mass insertion***.** If you are inserting a group of sequential records, take advantage of mass insertion by using the ALTER command to change free space to (0 0) after the data set is loaded.

To summarize these guidelines, do not use FREESPACE. In the case of large evenly distributes inclusions, define (0 nn), with nn not being larger than 10%.

4.4.7 Data Control Area Reclaim

Empty data control areas are control areas that are totally free and located below the HURBA. They are created when at load time a data CA is loaded with records that belong to a keyrange. Later these records are deleted, the key value is not reutilized, the CA is empty, and the free space is not reclaimed for records with keys from the original keyrange. This set of keys is called creeping keys, which continue to grow in its alphanumeric value. VSAM was not designed to effectively process creeping keys. Empty data control areas cause the following storage and performance problems:

- Unused and fragmented space on volumes as the data set grows
- Multiple volumes and multiple extents and lots of activation of the EOV routine
- Slow sequential processing because sequence set must be read to determine the CA is empty or not

Before DFSMS V1.12, the only solution was to reorganize the data set. This solution can create problems, such as resource consumption and lack of availability.

In DFSMS V1.12, the Control Area Reclaim function was introduced. This function adds the empty CAs to a free CA list, so VSAM realizes the last original record is deleted. This function applies to any KSDSs, including ICF catalog data sets, AIX and base clusters, and SMS and non-SMS managed and temporary data sets. CA Reclaim is not possible for data set types other than KSDS or a KSDS defined with IMBED. Remember that for an existing VSAM data set, enabling CA reclaim does not reclaim internal CAs that were previously made totally free. Only the ones that become free after CA reclaim is enabled are reclaimed.

However, the Control Area Reclaim function does not reclaim the following data component CAs for a KSDS:

- Partially empty CAs
- CAs already empty before CA reclaim is enabled
- CA with RBA 0 and the CA with the highest key of the KSDS
- CAs in KSDSs with the IMBED option active
- Application opening a data set with global shared resources (GSR)

Control Area Reclaim is activated at two levels:

- System Level:
  - SYS1.PARMLIB(IGDSMSxx) specifies CA_RECLAIM(DATACLAS|DATACLASS).

    Default: NONE.
  - SETSMS command is issued that explicitly specifies CA_RECLAIM(DATACLAS|DATACLASS).
    - Overrides the SYS1.PARMLIB(IGDSMSxx).
    - CA_RECLAIM(NONE) to stop CA Reclaim immediately as an emergency measure.
    - To activate at Sysplex scope, you must route to other MVS systems through the RO *ALL, SETSMS command.
- Data Set Level:
  - The DATACLAS for a specific KSDS data set specifies or defaults to CA Reclaim=Y during IDCAMS DEFINE or DD define.

    CA Reclaim=N set to not do CA Reclaim.
  - Or a subsequent IDCAMS ALTER command is issued that explicitly specifies RECLAIMCA.

4.4.8 Index options

There are two index options that are associated with a cluster defined in the IDCAMS DEFINE command and stored in the catalog: REPLICATE and IMBED. They use real 3390 characteristics that do not exist anymore with RAID controllers, such as the DS8000.

These options, which started with DFSMS 1.5, are no longer valid. No warning message is issued. Because of that they are not addressed here. However, data sets already defined (before DFSMS 1.5) with the IMBED or REPLICATE options are still supported.

With DFSMS 1.11, there is a new Health Check warning message that detects these attributes for catalogs.

With DFSMS 1.12, when these data sets are migrated or dumped by DFDSS or DFHSM, at restore or recall these attributes are deleted.

Guideline: Do not use the IMBED or REPLICATE options.

4.4.9 Keyrange and ordered options

No supported release of z/OS accepts the IMBED, REPLICATE, and KEYRANGE attributes for new VSAM data sets. In fact, using these attributes can waste storage space and often degrades performance. Servicing these VSAM data sets has become increasingly difficult. In some cases, unplanned outages have occurred.

Guideline: Minimize or eliminate your use of KEYRANGE.

4.4.10 Share options

There are several mechanisms to implement VSAM data set read and write integrity when accessed by several tasks:

- Intra address space locks for serializing VSAM data sets among tasks of the same address space. The granularity of serialization is at data CI level.
- VSAM SHAREOPTIONS play an important role in VSAM integrity. They specify whether and to what extent VSAM data sets are to be shared among tasks in one or multiple z/OS address spaces. This feature uses ENQ on SYSVSAM.data-set-name to achieve the required serialization. The granularity of serialization is at cluster level.
- ENQ/Reserve serialization functions that are issued by the application.
- JCL disposition (OLD or SHR), which also implies in an ENQ. The granularity of serialization is at cluster level.
- Record level sharing (RLS) locking mechanism, which is covered in “Implementing VSAM RLS” on page 220. It uses a structure at the coupling facility, and the granularity of serialization is at logical record level.

These mechanisms are covered in the performance chapter because usually integrity and performance vary inversely. Total integrity might result in poor performance.

When you define VSAM data sets, specify how the data is to be serialized (if shared) within a single system or among multiple systems that have access to your data. Before you define the level of sharing and the serialization mechanism for a data set, evaluate the consequences of these processes:

- Reading incorrect data (a loss of read integrity)
- Writing incorrect data (a loss of write integrity)

Also, analyze the situations that can result when one or more task programs do not adhere to guidelines for accessing such shared data sets. It is also important to avoid the unnecessary use of certain serialization functions, which can cause a performance degradation.

For more information, see 2.2, “Sharing VSAM data sets” on page 50.

Guidelines:

- If you are sure that no application updates or deletes records in the VSAM data set, do not use SHAREOPTIONS 4. It causes buffer pool refresh for direct reads or writes. The buffer pool is useless and no I/Os are saved.
- If you are doing your own serialization, use ENQ SHARE instead of ENQ EXCLUSIVE for reads.
- In a cross-systems environment, avoid the use of the RESERVE macro, which locks the full 3390 volume. Instead, use the ENQ macro for a global resource with GRS. However, for applications that already use the Reserve macro, it can be transformed (without changing the code) in a Systems ENQ through the GRS Conversion RNL.
- Review 2.2, “Sharing VSAM data sets” on page 50.

4.4.11 Initial load option

Initial load mode occurs when you load *(write)* your data *sequentially* in a VSAM data set that has a HURBA equal to zero. The data set to be loaded is already IDCAMS defined (cataloged and described through the DSCB in VTOC), but empty. Initial load can be done by using IDCAMS REPRO, IDCAMS IMPORT, or an application program. There are two cases:

- After the DEFINE that indicates that this is the first time you are writing into the data sets its initial contents
- When a data set that is defined with the REUSE attribute is opened with MACRF=RST (reset) in the ACB. This command means that you reuse the data set with new data.

You must load a data set first because in VSAM non-RLS mode, your application program cannot open for input to an empty data set.

Initial load always uses an NSR buffering. For more information, see “Nonshared resource (NSR)” on page 153.

RECOVERY and SPEED options

There are two options that might affect performance of an initial load process: RECOVERY and SPEED. Those options have to do with the techniques used by an access method for creating physical records in the empty data set:

- RECOVERY: This option has two steps:
  - VSAM formats a a set of 3390 track (erases the previous contents) by running the Write Count Data CCW (also called Write Format). This process creates physical records full of zeros in the Data part (of a CKD physical record). These zeros also serve as a VSAM logical end-of-file indicator.
  - The data portion of tracks is filled with real data by running the Write Data CCW (Write Modified).

  The total number of I/Os increases significantly when compared with SPEED. For more information about Write Format and Write Modified CCWs, see “DASD cache highlights” on page 199. Many of the tracks beyond HURBA are not formatted.

  The Recovery option allows the restart of the load operation from the last written record just before the first end-of-file physical record. However, other difficulties such as tape repositioning (in the input file) have discouraged customers from using it.
- SPEED: The data CIs/CAs are not pre-formatted with zeros. The Write Count Data CCW is run in just one pass, creating the physical record with real data in the data part. An end-of-file indicator is written only after the last record is loaded. As in the Write Modified type of channel program (in RECOVERY), many writes are assembled in the same channel program to improve processor and I/O usage. This configuration depends on the number of NSR data buffers. Free CIs (if you declare CAs with free CIs, in the second parameter of FREESPACE) are not written together with occupied CIs in the same channel program.

  Because SPEED runs a single pass, you get better performance with it for initial load mode.

  Remember that during load mode processing, you cannot share the data set. Share options are overridden during load mode processing to (1 3). For more information, see 2.2, “Sharing VSAM data sets” on page 50. When a shared data set is opened for create or reset processing, your program has exclusive control of the data set within your operating system. This control is accomplished by use of the ENQ Systems exclusive in the SYSVSAM resource. If data sets are shared between systems, never place this resource name (SYSVSAM) in the exclusion list.

Other options that affect performance

The following other VSAM options affect the initial load performance:

- System managed buffering (SMB) because of better buffer management.
- Extended format. For more information, see 2.3, “Extended format” on page 64.

Guidelines:

- Do not use the RECOVERY option
- Use SMB with CO or CR options. For more information, see “System managed buffering (SMB)” on page 164.
- Use extended format
- Remember that any specific SHAREOPTIONS assignment in an initial load is forced to (1 3).

4.4.12 Region size

The key to reduce the number of I/O operations is to keep more data in virtual storage. The guidelines that are provided here use this technique. However, keeping more data in virtual storage might exhaust this resource. Therefore, before addressing buffering, the region size parameter must be covered to avoid an S878 type of abend, as shown in the VSAM message in Figure 4-4.

```
IDC3351I ** VSAM CLOSE RETURN CODE IS 136
Not enough virtual storage was available in the program's address space for a work area
for Close
```

Figure 4-4 VSAM message that indicates too small a region size

The portion of the user's private area within each virtual address space that is available to the user's programs is called the *user region*. It is located from the bottom to top of the private area. There are four user regions per address space:

- Below the 16 M line
- Above the 16 M line and below the 2 G bar (extended user region)
- Above the 2 G bar (low user region)
- Above the 2 G bar (high user region)

The region size imposes a limitation to the two user regions below the 2 G bar. Figure 4-5 shows the virtual storage layout in z/Architecture. For a description of each area, see the *z/OS MVS Initialization and Tuning Guide*, SA22-7591.

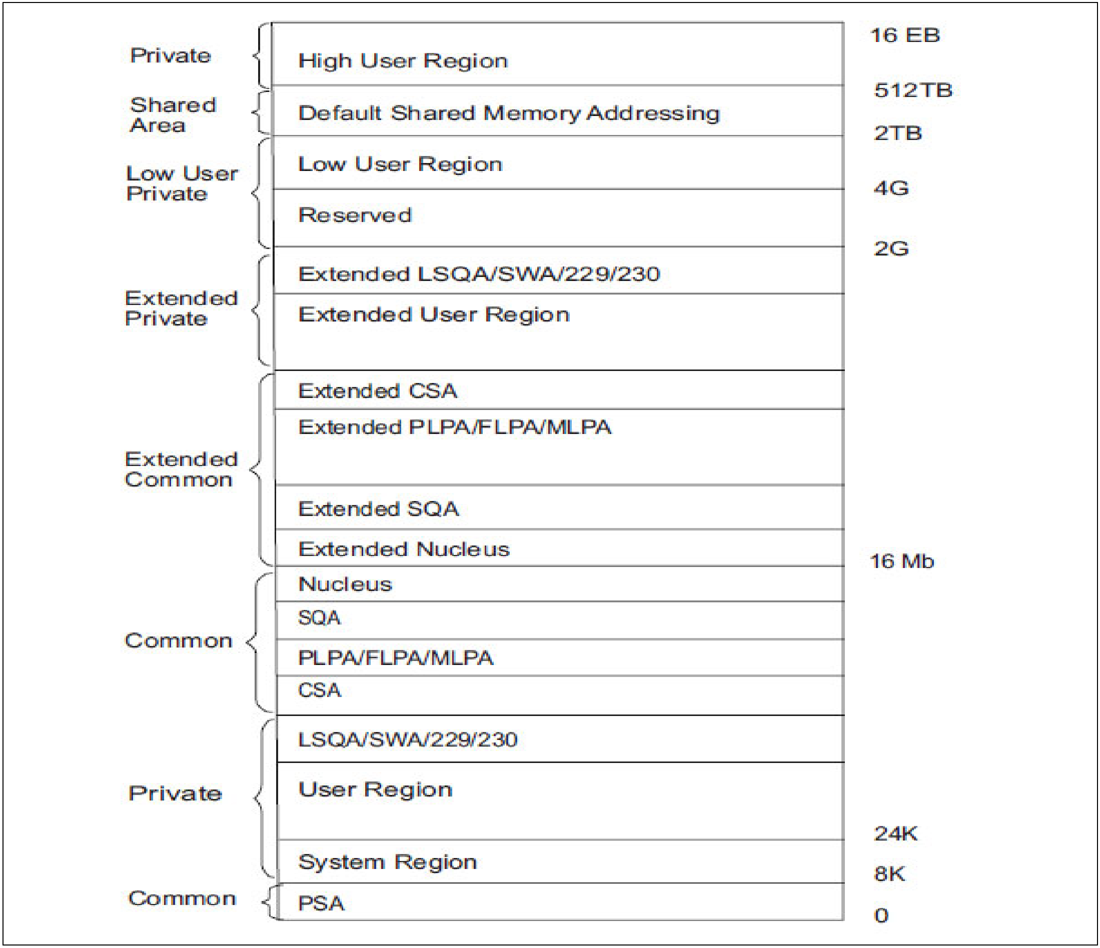


Figure 4-5 Address space layout

Specify a job's region size by coding the REGION parameter in the JOB or EXEC statement of a JCL procedure. However, there is an exit routine (IEFUSI) that delivers the final word about such value. z/OS rounds all region sizes to a 4K multiple. Some jobs run out of virtual space and abend in these circumstances:

- The REGION specified is greater that the available private area.
- A private area GETMAIN reaches the limit that is determined by the IEFUSI exit. The default is the specified REGION plus 64 K. This process occurs even if there is virtual free space available in the user region above the region limit. This behavior is intended to prevent a task from using lots of virtual storage, and so lots of real storage. The REGION value is also a limitation for the high value of a variable GETMAIN.
- A private area GETMAIN cannot be served because no contiguous virtual free space exists with the required size. This means a collision between the top-to-bottom and bottom-to-top sub pools.

Table 4-1 shows how the JCL REGION parameter is interpreted and how much virtual storage is available, below and above 16 M, for each step of a job.

Table 4-1 Region JCL parameter

REGION value	Region available below 16 MB	Region available above 16 MB
0k > REGION < 16 MB	Establishes the private area size below	32 MB
16 MB > REGION =< 32 M	All private area below is available	32 MB
32 MB > REGION =< 2 G	All private area below is available	Establishes the private area size available above 16 MB
0 K or 0 M	All private area below is available	All private area above is available

If your job is experiencing constraints in virtual storage below 16 MB when VSAM data sets are opened, you can relieve storage usage below 16 MB. Specify that VSAM allocate the buffers and control blocks above 16 MB. For more information, see “Locating VSAM buffers above 16 MB” on page 163.

Guidelines: Do not hesitate to increase your region size, or even avoid it by specifying REGION=0 (which indicates no limit). Good buffering can reduce the number of I/Os, job elapsed time, processor time and device connect, and device disconnect time. If your job is I/O bound, giving it enough resources allows it to run more quickly. That configuration is a benefit for the user and for total system performance.

4.4.13 Buffering options

Buffering in virtual storage is an important technique in VSAM to improve performance. You can use this technique through the buffering options.

A VSAM resource pool is a set of VSAM control blocks plus a buffer pool (several buffers with the same size, which is the CI size). These control blocks are not enough to allow the data set to be processed by VSAM. At open time, more control blocks are created, which together with the ones in the buffer pool form structure control blocks. Consider the following key aspects for managing a buffer pool:

- The algorithm to keep CIs in the buffers. For random, use the LRU algorithm, which keeps the most accessed CIs in buffer. For sequential access, use the replacement sequential algorithm, which has two pieces: Look ahead for not yet requested reads, and disposing of already processed CIs (reads or writes) to make room for the new ones.
- What to do with the CIs updated in the buffers. Determine whether they are stored through in DASD immediately (store through algorithm) or later (store in algorithm)? For random, use the Defer Write option, where the installation decides what to do. The default is store through. For sequential, VSAM defers the write until half of the buffers are ready for the write. The VSAM Share options can change this behavior. For more information, see “Sharing VSAM data sets” on page 50. Store in algorithm favors performance, but might cause integrity issues.
- An unnecessary large number of buffers can degrade the performance. For random, have all the index CIs and many data CIs in the buffer pool. For sequential, have one index CI buffer (sequence set) and many data CIs buffers for the look ahead.

- When the same cluster is shared between two applications tasks (for read/write) in different MVS systems, the coherency of the buffer pool must be ensured. That is, if a CI is altered in a local buffer pool in one MVS, the other MVS local buffer pool must reflect the change.

When a buffer's content is written along a random access, the space is not released. The control interval remains in storage until overwritten with a new control interval. If your program refers to that control interval, VSAM does not have to reread it. VSAM checks to see whether the wanted control interval is in storage. This process is not valid for share option 4, where buffers used for random processing are refreshed for each request.

Buffer space is released when all data sets that are using the buffer pool are closed. For the LSR/GSR mode of buffering, it is also released when the DLVRP macro is issued. If an abend occurs before closing VSAM data sets, the buffers are not flushed by the VSAM or MVS Recovery Termination Manager routines. The application must decide during an (E)SPIE/(E)STAE routine to close the data sets and flush the buffers. For more information about ESTAE and ESPIE macros, see *OS/390 MVS Programming: Authorized Assembler Services Reference, Volume 2 (ENFREQ-IXGWRITE)*, GC28-1765. Ensure that such routines do not gain the control when the abend is caused by an operator CANCEL command. Also, there are options (TRAP) in IBM Language Environment® to control the flush of those buffers.

Choosing the correct VSAM buffer pool size and buffering technique is the key to reducing the number of I/Os and reducing the I/O response time (Tr). More buffers (either data or index) than necessary can cause excessive paging and excessive internal VSAM processing. There is an optimum point at which more buffers do not decrease the transaction I/O request response time and device connect time. Attempt to have data available just before it is to be used. If data is read into buffers too far ahead, it might be paged out.

For more efficient use of virtual storage, buffer pools can be shared among data sets by using locally or globally shared buffer pools. There are four types of resource pools management, called modes, according to the technique used to manage them:

- Nonshared resource (NSR)
- Local shared resource (LSR)
- Global shared resource (GSR)
- Record-level sharing (RLS)

These modes are not an attribute of a cluster, but an access option as interpreted by the Open routine. The same cluster can be opened in NSR by a task and in LSR by another task. Also, the same task can open a cluster initially as LSR and later as NSR. For more information about RLS, see Chapter 5, “VSAM Record Level Sharing” on page 209.

Buffers are acquired dynamically in NSR mode when the data set is opened. However, with LSR/GSR they are acquired explicitly by the application program before the Open through the BLDVRP macro. The amount of space for buffers is based on the parameters in effect when the program opens the data set.

One of the major sources for determining how much space is allocated in a VSAM buffer pool is the access method control block (ACB). The system obtains buffer pool information for VSAM data sets from the following sources:

- The program’s ACB
- DD statements that override SMS data class information
- SMS data class
- The catalog entry

For LSR buffering, the BUFND, BUFNI, BUFFERSPACE, and STRNO parameters do not apply and cannot be overridden by JCL.

Table 4-2 lists the parameters that influence buffer pool allocation.

Table 4-2 Parameters affecting buffer allocation

Source	**Parameter**
DD Statement	BUFSP, BUFND, BUFNI, and ACCBIAS
SMS data class	RECORD ACCESS BIAS (ACCBIAS)
The program's ACB	MACRF=(IN|OUT, SEQ|SKP, DIR, NSR, LSR, GSR, RLS, NUB, KEY, ADR, CNV, DFR, NDF, LEW, NLW, SIS, NIS)
	STRNO=n, BUFSP=n, BUFND=n, BUFNI=n[a]
	SHRPOOL=n[b]
The catalog entry for the data set	BUFFERSPACE

a. Applies only to NSR
b. Only for LSR/GSR, connect the ACB to an existent resource pool

The following are the specifications in the ACB MACRF parameter that affect buffering and data set processing:

- Buffering management: NUB for management of I/O buffers to be done by VSAM. Or UBF when management of I/O buffers is left up to the user, or a VSAM exploiter as DB2. NUB is the default value.
- VSAM buffering modes to be used: NSR (default), LSR, GSR, or RLS. These modes assign buffers to strings, so the number of strings also affects the buffering.
- The manner in which the records are *intended* to be accessed: Direct (DIR), sequential (SEQ), or skip sequential (SKP).
- Type of argument that is used to identify records along direct access: By key (KEY option), by RBA (ADR option), by RRN (relative record number), or access is to the entire contents of a control interval rather than to an individual data record (CNV). KEY is the default.
- What type of processing is done in the data set: Input (IN, default) or output (OUT).
- How writes are to be managed: defer writes (DFR) or not (NDF). For more information about deferring writes, see "Deferring write requests" on page 163.
- Using LSR, how VSAM deals with conflict for an exclusive control in buffers by several tasks: VSAM defers the request until the resource becomes available (LEW, the default) or VSAM returns the exclusive control return code X'14' to the application program (NLW). In this case, the application program is then able to determine the next action.
- Insert record strategy: Split CIs and CAs at the insert point (SIS) or at the midpoint (NIS, the default) when running direct PUTs.

Use the ACB MACRF parameter to code the types of access you intend to use during the processing. They are used mainly for buffering management and at open time. To process the data set, use the request parameter lists (RPLs) to specify the processing options for that particular request.

If you open a data set whose ACB includes MACRF=(SEQ,DIR), buffers are allocated according to the rules for sequential processing, as done by NSR buffering management.

4.4.14 How BUFFERSPACE, BUFSP, BUFND, and BUFNI affect buffering

The buffering keywords that are addressed in this section apply only to NSR buffering mode.

BUFFERSPACE keyword

The BUFFERSPACE value in the catalog entry for the data set is the *minimum* amount of buffer space. The value that is assigned to BUFSP, in JCL or ACB, is the *maximum* amount of buffer space. The BUFFERSPACE value applies to the whole cluster. Additional buffer space can be assigned to any data set by these processes:

- Modifying the data set's BUFFERSPACE value
- Specifying a larger BUFSP value with the AMP parameter in the data set's DD statement. If you do not specify BUFSP, the amount of virtual storage that is used for buffers is the *largest* of these values:
  - The amount that is specified in the catalog (BUFFERSPACE)
  - The amount that is determined from BUFND (number of data buffers) and BUFNI (number of index buffers)
  - The minimum storage that is required to process the data set with its specified accessing options, as sequential, direct

BUFSP keyword

The BUFSP value takes precedence over BUFNI and BUFND as follows:

- If the number of buffers that are specified in the BUFND and BUFNI sub parameters exceed the virtual storage specified in the BUFSP space, the number of buffers is decreased to fit in the BUFSP space:
  - If the ACB indicates direct access only, the number of data buffers is decreased until it reaches the BUFSP value. However, it never becomes less than the minimum required. If the BUFSP value is not reached, the number of index buffers is decreased until BUFSP is reached.
  - For sequential access, BUFNI is decreased to reach BUFSP up to the minimum plus one. If the limit is not reached, BUFND is decreased. When the minimum is reached, but BUFSP is not reached, one buffer is subtracted from the number of index buffers.
- If BUFSP specifies more space than is required by BUFND and BUFNI, the number of buffers is increased to fill the BUFSP space as follows:
  - For direct access only, more index buffers are allocated.
  - For sequential access, one more index and as many data buffers as possible are allocated.

If BUFSP is specified and the amount is smaller than the minimum amount of storage required by VSAM to process the data set, VSAM fails when opening the data set.

When processing a data set by using a path (for AIX), the number of needed buffers increases. This increase is because buffers are needed for the alternate index, the base cluster, and any alternate indexes in the upgrade set.

When a base cluster is opened for processing with its alternate index, the BUFSP, BUFND, BUFNI, and STRNO parameters apply only to the path's alternate index. The minimum number of buffers are allocated to the base cluster unless the cluster's BUFFERSPACE value (specified in the DEFINE command) or BSTRNO value (specified in the ACB macro) allows for more buffers. VSAM assumes direct processing, and extra buffers are allocated between data and index components accordingly.

The default for the number of VSAM allocation buffers depends on the type of buffering. In NSR, the numbers are as follows:

- For the index component: BUFNI=STRNO + 2
- For the data buffers: BUFND=STRNO+1

BUFND keyword

One of the data buffers (BUFND) is used only for formatting CAs and splitting CIs and CAs. Only data buffers are needed for ESDS, RRDS, or LDS. No index buffers are needed.

This default at NSR for buffering is also the *minimum* number of buffers that are required by VSAM.

BUFNI keyword

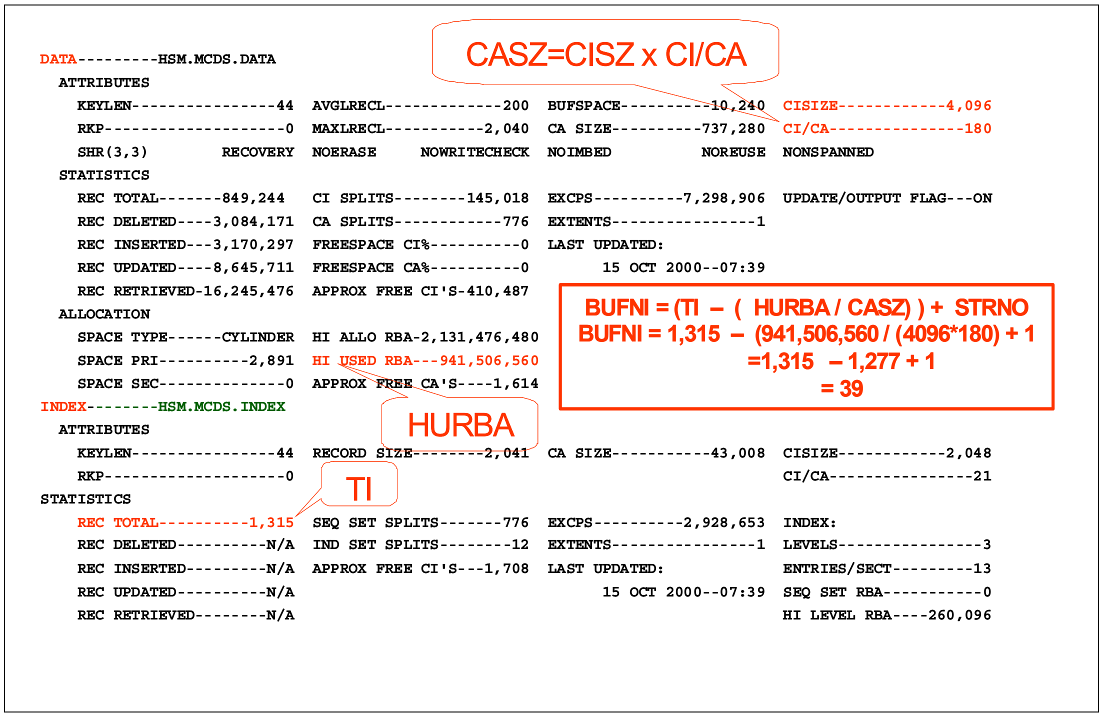
The formula for recalculating BUFNI for NSR key direct buffering by using data from LISTCAT output is:

BUFNI = TI - (HURBA / CASIZE) + STRNO

Where TI is the total number of CI index records.

HURBA / CASIZE gives you the number of occupied data CAs. There is one sequence set index CI per every data CA. The subtraction calculates the number of index set CIs only. In a direct access is important to have all the index sets buffered in memory.

In Figure 4-6, the string number used was 1.



```
DATA---------HSM.MCDS.DATA
  ATTRIBUTES
    KEYLEN----------------44   AVGLRECL-------------200   BUFSPACE----------10,240   CISIZE------------4,096
    RKP--------------------0   MAXLRECL-----------2,040   CA SIZE---------737,280   CI/CA---------------180
    SHR(3,3)        RECOVERY   NOERASE     NOWRITECHECK   NOIMBED          NOREUSE   NONSPANNED
  STATISTICS
    REC TOTAL-------849,244    CI SPLITS--------145,018   EXCPS----------7,298,906   UPDATE/OUTPUT FLAG---ON
    REC DELETED----3,084,171   CA SPLITS------------776   EXTENTS----------------1
    REC INSERTED---3,170,297   FREESPACE CI%----------0   LAST UPDATED:
    REC UPDATED----8,645,711   FREESPACE CA%----------0         15 OCT 2000--07:39
    REC RETRIEVED-16,245,476   APPROX FREE CI'S-410,487
  ALLOCATION
    SPACE TYPE------CYLINDER   HI ALLO RBA-2,131,476,480
    SPACE PRI----------2,891   HI USED RBA---941,506,560
    SPACE SEC--------------0   APPROX FREE CA'S----1,614
INDEX--------HSM.MCDS.INDEX
  ATTRIBUTES
    KEYLEN----------------44   RECORD SIZE--------2,041   CA SIZE-----------43,008   CISIZE------------2,048
    RKP--------------------0                                                          CI/CA----------------21
STATISTICS
    REC TOTAL----------1,315   SEQ SET SPLITS-------776   EXCPS----------2,928,653   INDEX:
    REC DELETED----------N/A   IND SET SPLITS--------12   EXTENTS----------------1   LEVELS----------------3
    REC INSERTED---------N/A   APPROX FREE CI'S---1,708   LAST UPDATED:              ENTRIES/SECT---------13
    REC UPDATED----------N/A                                    15 OCT 2000--07:39   SEQ SET RBA-----------0
    REC RETRIEVED--------N/A                                                         HI LEVEL RBA----260,096
```

CASZ=CISZ x CI/CA
BUFNI = (TI – (HURBA / CASZ)) + STRNO
BUFNI = 1,315 – (941,506,560 / (4096*180) + 1
=1,315 – 1,277 + 1
= 39
HURBA
TI

Figure 4-6 LISTCAT for the NSR BUFNI calculation for STRNO=1

Guidelines: Do not specify BUFSP or BUFFERSPACE, take the default value. Doing so avoids mistakes in calculation that can lead to different results from those expected. SMB frequently uses NSR, and remember that these parameters do not apply to LSR, GSR and LSR.

4.4.15 Buffering techniques

This section addresses the various buffering techniques (modes).

Nonshared resource (NSR)

NSR is the default VSAM buffering technique. It has the following characteristics:

- The buffers in the resource pool are *not* shared among VSAM data sets. A resource pool contains only CIs from the same VSAM data set.
- The buffers are in the application address space private area (user region).
- It is suited for sequential processing because the buffers are managed by using a sequential replacement algorithm:
  - For sequential processing, there is read look-ahead (CIs that are not yet requested are brought to buffers). CIs in the buffer pool are not managed by LRU. After use by the application program, they are strong candidates to be covered in its buffer by another CI.
  - For direct processing, there is no read look-ahead (which is good). However, CIs are still managed by the sequential replacement algorithm, so the LRU is not active (which is bad). Generally, use SMB buffering when in NSR mode for processing records randomly.
- High-level languages such as Cobol only use NSR buffering mode.
- The resource pool (buffer pool plus control blocks) is built automatically by OPEN according to the BUFND, BUFNI, BUFFERSPACE, BUFSPC. parameters.
- NSR has specific properties when it is assigning buffers to strings (or place holders):
  - Each string has its own buffer for the index sequence set component. The additional index buffers provided are used to cache index set records, which are shared among strings. The additional buffers are allowed by the buffer parameters.
  - Each string has its own buffer for the data component. If more data buffers are provided, they are used for these processes:
    - Sequential reads and writes
    - CA splits
    - Spanned records
  - There is also a buffer for insert records is used only along CI splits.
- It dynamically extends the number of strings as they are needed by concurrent requests for the ACB.
- For subtask sharing (tasks in the same address space), when a CI is not available (serialize) for the type of task processing that is requested, VSAM under NSR buffering can manage the contention. For more information, see 4.4.10, “Share options” on page 144.

Figure 4-7 shows how NSR buffers are constructed.

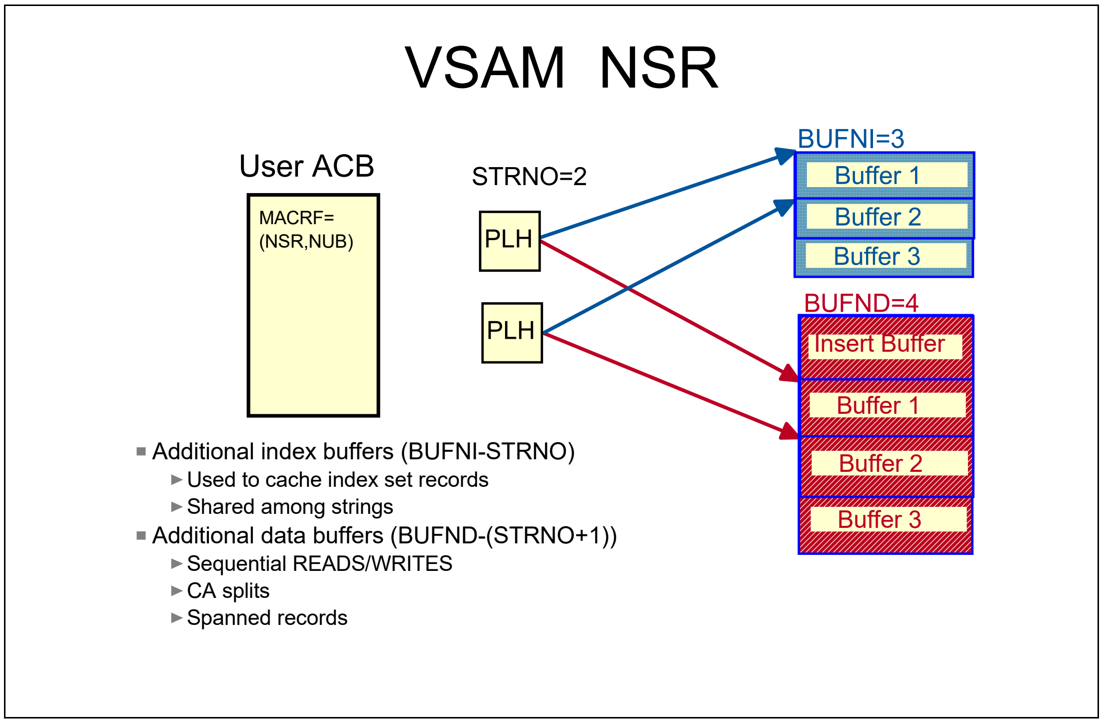


Figure 4-7 NSR buffering

NSR with sequential access

NSR is best used for applications that use sequential or skip sequential as their primary access mode.

At initial buffer pool loading for reads (done by the Open routine) on the behalf of a string, VSAM NSR loads the buffers. It uses just one channel program with chained CCWs (one Read CCW per each physical block). The number of CIs in the channel program is equal to the number string's assigned buffer plus *all* additional buffers to a *maximum* of CIs/CA. The I/Os are scheduled on CA boundaries.

When another string, at the same time, needs buffers, VSAM uses its assigned buffer and the remaining buffers to a maximum of CIs/CA. Having more buffers than CI/CA plus one is useful only when you have more than one string. When a string finishes using its buffers (processed by the application program that is associated with the string), more buffers are available to other strings.

The overlap between processor and I/O is implemented by VSAM NSR after initial buffer pool load:

- For sequential reads, after one-half of the buffers is being processed (read) by the application program task, an I/O operation is scheduled for this half. This process continues until a CA boundary is encountered. The application must then wait until the last I/O to the CA is done before it proceeds to the next CA. The I/O operations are always scheduled within CA boundaries.
- For sequential writes, after half of the buffers are filled by the application program task with logical records, an I/O operation is scheduled for this half. Then, for sequential PUT processing, VSAM NSR does not immediately write the updated CI from the buffer unless

a CI split is required. VSAM saves I/O operations by deferring sequential writes. For more information about deferred write (sequential and direct), see "Deferring write requests" on page 163.

When you are accessing data sequentially, you can increase performance by increasing the number of data buffers, up to a certain limit. When there are multiple data buffers, VSAM NSR in sequential access uses a read-ahead function to read the next data control intervals into buffers as they become available. Table 4-3 shows results of lab tests with various numbers of buffers. Remember that, in these lab tests, the data set record processing is minimal (I/O-bound). Also, these tests were not run in a controlled environment, so the elapsed time value can vary according to priorities and the system workload. The time values are total and not an average per I/O operation.

Table 4-3 NSR: Read sequential varying the number of buffers

Data buffers	**Index buffers**	**EXCPs**	**Device connect time**	**SRB time**	**TCB time**	**Elapsed time**
Default=2	Default=1	167828	23.50	2.12	7.52	120
10	1	33568	14.48	0.53	3.99	36
30	1	11577	11.82	0.26	3.44	22
50	1	6948	11.40	0.21	3.35	19
50	2	6948	11.40	0.21	3.37	19
90	1	4633	11.26	0.18	3.44	18
181	1	3476	11.15	0.19	3.86	14
181	3	3476	11.19	0.19	3.91	19
10000	1	3476	11.16	0.19	3.89	19
SMB 01 Stripe		4633	15.75	0.23	3.45	22
SMB - 02 Stripes		7580	17.11	0.17	3.61	15
SMB - 04 Stripes		14073	19.3	0.17	3.65	11
SMB 08 Stripes		27061	24.30	0.17	3.77	10
Tip: All times are in seconds						

You can draw the following conclusions based on the numbers in Table 4-3.

- As more buffers are used, the number of Execute Channel Programs (EXCPs) is reduced. For more information about a VSAM EXCP in numerical values, see Appendix B, "Miscellaneous performance items" on page 407. With VSAM, the number of EXCPs is equal to the number of SSCH instructions. That is the number of I/O operations, and not the number of transferred CIs or even the transferred physical blocks like in QSAM.
- The decrease in the number of EXCPs means that there are fewer I/O operations and so fewer I/O interrupts. That is why SRB time consumption drops compared with TCB time. Remember that the posting of an I/O waiting task runs in SRB mode.
- The TCB time drops because of the decrease in I/O preparation. After an optimum point, the TCB time increases because of excessive buffering management processing. Remember that the program in the example is *read and forget,* so all the TCB time is used in the I/O process.

- Having only one index I/O buffer per string does not hinder performance in a read sequential access. To access a logical record, VSAM gets the next index sequence set and uses it for the complete data CA. At end of the CA, the next index sequence set CI is retrieved by using the horizontal pointers, rather than the vertical pointers. Extra index buffers have little effect during sequential processing as shown in Table 4-3 on page 155.

For more information about SMB mode of NSR buffering, see "System managed buffering (SMB)" on page 164. Briefly, SMB is a modification of NSR buffering that boosts the performance for random NSR accesses.

With SHAREOPTIONS 4, buffers are refreshed at each request. Also, the read-ahead (a look-ahead synonym) function has no effect and defer write is not used. Therefore, for SHAREOPTIONS 4, keeping data buffers at a minimum can actually improve performance.

The use of the POINT macro randomly positions the data set for subsequent sequential retrieval. It does not cause read-ahead processing unless RPL OPTCD=SEQ is specified.

When reading sequentially, buffers are used to balance the ability of the application to process data with the capacity of the storage device to deliver the data to the application. Factors that affect the amount of data CI buffers needed are the number of logical records per data CI, and how heavy its processing is. A data CI with many records saturates the read data buffers before a data CI with few records. Saturated means that you do not get better performance by increasing the number of buffers. In this case, the CPU processing becomes the limiting factor.

By specifying enough data buffers, you can sequentially access the same amount of data per I/O operation with small data CIs as with large data CIs.

If you experience a sequential performance problem while you are waiting for input from the device, specify more data buffers to improve your job's run time. More data buffers allow more read-ahead processing and less I/O operation for the same amount of data. This advantage drastically decreases the total connect time and some of the total disconnect time.

If your data set is SMS-managed, NSR uses extended format. Therefore, you do not need to worry about how many buffers to specify for the index or data component. This configuration allows you to take advantage of system managed buffering. For more information, see "System managed buffering (SMB)" on page 164.

Buffering in NSR initial load mode

Initial load mode occurs when you load your data *sequentially* in a VSAM data set that has a HURBA equal to zero (empty). IDCAMS Repro Initial Load always uses an NSR buffering option (can be modified by SMB).

Pay attention to performance aspects of using RECOVERY or SPEED attributes. For more information, see 4.4.11, "Initial load option" on page 145.

There is a difference with index buffers when you compare sequentially reading and initial loading (or extending) a VSAM KSDS cluster. In the first case (reading), VSAM never refers to the index set, so there is no need to set aside buffers for such index CIs. With initial loads the index set CIs are built, so there is a little performance gain if you provide buffering for them.

With extended format data sets and SPEED (writing each CA with just one I/O pass), you can get much better performance by using SMB. For more information, see "System managed buffering (SMB)" on page 164.

Table 4-4 shows the results of loading an empty data set with 2,000,000 records by using IDCAMS REPRO. The best non-SMB result is obtained when:

Data buffers = 181 = CIs/CA + 1

Index Buffers = 3 (the number of index levels after data set loading)

Table 4-4 NSR: Initial Load mode, varying the number of buffers

Data buffers	**Index buffers**	**Extended Format / stripes**	**EXCPs**	**Device connect time (sec)**	**CPU Time (sec)**	**Elapsed time (sec)**
Default	Default	No / 1	27856	41.2	8.65	68
90	Default	Yes / 1	13970	24.9	8.08	41
181	Default	Yes / 1	13969	24.5	8.08	42
181	3	Yes / 1	12810	24.6	8.08	44
360	Default	Yes / 1	13969	25.5	8.07	44
10000	3	Yes / 1	13971	24.3	8.06	42
SMB	SMB	Yes / 1	12813	28.4	8.05	48
SMB	SMB	Yes / 2	14153	28.6	8.18	30
SMB	SMB	Yes / 4	11991	20.2	8.2	28
SMB	SMB	Yes / 8	22815	34	8.24	22

NSR with direct access

NSR is not intended for direct (random) access. However, many of your applications can use NSR for direct processing because it is simple. This section explores how NSR works for direct access. Remember that today you have solutions available to avoid NSR direct processing without changing your code: System-managed buffering (SMB) and batch local shared resources (BLSR). Their functions are addressed later in this chapter.

In direct access, records are randomly accessed by Key, RBA, or RRN depending on the data set organization. Increasing data buffers do not boost performance because VSAM NSR does not implement an LRU algorithm to manage its buffer pool. In other words, no buffer hits are caused by revisiting the same CI.

NSR read direct processing has these advantages:

- Does not use look-ahead buffers
- Reads only one CI per GET request

Within an NSR buffer pool, data and index sequence set buffers are not shared among strings. For more information, see “Nonshared resource (NSR)” on page 153. Each string is associated with one RPL. When an application uses an RPL to issue a direct GET for a record with a key of 1234 using NSR, VSAM manages buffers as follows:

1. VSAM locates the correct CI by using a top-down search through the index set.
2. VSAM reads the data CI into storage and gives the user the requested record.
3. If the user issues another direct GET for a record with a key of 6789 in a different CI from record 1234, the same data buffer is used. The CI containing 6789 overlays the CI containing 1234.

4. If the user issues another direct GET for a record with key 1235 in the same CI as 1234, that CI must be read in again. This process occurs because the intervening GET for 6789 causes the previous buffer contents to be lost.

This problem cannot be solved by using multiple strings. For example, if the requests for 1234, 6789, and 1235 use three different RPLs, both CIs are in storage when the request for 1235 occurs. However, VSAM looks *only* in the buffer that is assigned to the string related to RPL requesting 1235. It does not look in another string's buffers to satisfy the request. The CI must read in again anyway.

This NSR characteristic leads to performance complaints in cases where the processing is random, but the same CI is requested multiple times (revisiting) with intervening requests to other CIs.

When the number of I/O buffers provided for index records is greater than the number of strings, the surplus is shared among the strings:

- One buffer is used for the highest-level index record.
- Additional buffers are used, as required, for other index set index records, as shown in Figure 4-7 on page 154.

For a KSDS or VRRDS with NSR direct access, provide enough index buffers. This number must be at least to the value of the STRNO parameter of the ACB plus one, if you want VSAM to always keep the highest-level index record resident. You can increase performance by providing more buffers. Unused index buffers do not degrade performance. Direct processing always requires a top-down search through the index.

For optimum performance, the number of index buffers should at least equal the number of high-level index set CIs plus one per string. This configuration allows the entire high-level index set and one sequence set CI per string be in virtual storage. Table 4-5 shows how adding index buffers improves performance. The elapsed time can vary according to the system workload. Note that additional index buffers are not use for more than one sequence set buffer per string.

VSAM NSR reads index buffers one at a time (per I/O operation). Index buffers are loaded when the index is referred to. When many index buffers are provided, index buffers are not reused for another index CI until a requested index CI is not in storage.

VSAM NSR keeps as many index set records as the buffer space allows in virtual storage. Ideally, the index would be small enough to allow the entire index set to remain in virtual storage. Be aware of how index I/O buffers are used so you know how many to provide.

Additional data buffers do not increase performance because only one data buffer is used for each access, as shown in Table 4-5. The elapsed time can vary according to the system workload.

Table 4-5 NSR buffering with direct access; STRNO=1

Data Buffers	Index Buffers	EXCPs	Device Connect time (sec)	CPU time (sec)	Elapsed time (sec)
Default	Default	794950	103.34	31.76	506
90	1	794950	103.34	32.42	506
Default	3	505171	76.67	21.11	326
Default	5	445160	62.32	18.87	291

Data Buffers	Index Buffers	EXCPs	Device Connect time (sec)	CPU time (sec)	Elapsed time (sec)
Default	50	368823	51.64	16.36	242
Default	1200	368823	51.64	17.13	243

If your job is having performance problems randomly accessing VSAM data sets in NSR mode, you can improve performance with no changes in your applications with these tips:

- If your data set is SMS-managed and extended format, use system managed buffering. For more information, see “System managed buffering (SMB)” on page 164.
- If the above conditions are not met, use BLSR. For more information, see “Batch local shared resources (BLSR)” on page 168.

For more information about finding data sets that are candidates to use SMB or BSLR, see Appendix A, “Sample code” on page 381. Both offer real gains for direct access.

If your data set meets the requirements for both SMB and BLSR, use SMB for better results.

For direct output processing (PUT for update), VSAM defers write *only* when OPTCD=NSP option is specified in the RPL macro. Otherwise, VSAM immediately writes the updated CI.

Guidelines: Follow these guidelines for NSR:

- Even for sequential processing, use SMB whether possible. If not, adjust these settings:

  BUFNI = Number of index levels

  BUFND = Number of CI/CA plus one

  Additional data buffers, when STRNO higher than one.
- For direct access, use SMB or BLSR to convert to LRU algorithm. At worst, keep NSR and use this formula to recalculate BUFNI using data from a LISTCAT output:

  ```
  BUFNI = TI - (HURBA / CASIZE) + STRNO
  ```

  Where TI is the total number of CI index records.

  In batch application, there usually is just one requester, so STRNO=1.

  HURBA / CASIZE gives you the number of occupied data CAs. There is one sequence set index CI per every data CA. Then, by the subtraction, you calculate the number of index set CIs only. In direct access, have all the index sets buffered in memory.
- BUFND = Accept the default value (STRNO plus one)

For NSR with mixed access, use SMB or BLSR. If you must keep NSR for mixed access situations (sequential and direct), improve performance by performing these steps:

- Increase the number of index component buffers to the number of index levels to favor direct access. Table 4-5 on page 158 shows how the number of index component buffers can improve performance for direct access.
- Increase the number of data component buffers to favor sequential access. Table 4-3 on page 155 shows how this configuration can improve performance.

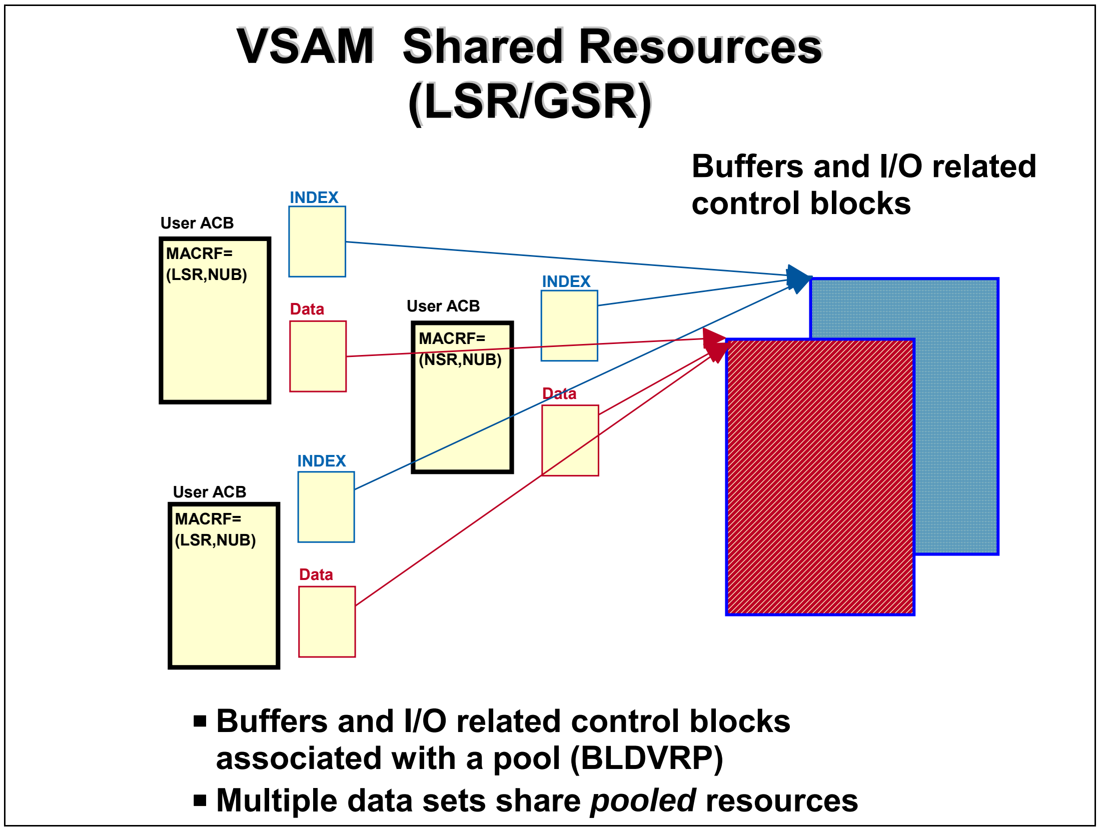
Local Shared Resources (LSR)

LSR is another mode of managing VSAM buffers. The buffers in an LSR resource pool have the following characteristics in this mode:

- They are shared among VSAM data sets accessed by tasks in the same address space. This configuration is a centralized way for buffer management. Instead of having individual buffer pools, you have a large one shared by several VSAM data sets, improving the utilization of the buffer pool resource.
- They are explicitly constructed by the application through the BLDVRP macro before the OPEN for the first data set that uses it.
- They are explicitly deleted by the DLTVRP macro.
- They are in the private area or in a hiperspace (if specified in BLDVRP macro).
- Its CIs are replaced based on the least recently used (LRU) algorithm, which is designed for random processing.
- They have no look-ahead, even for sequential processing.
- They are usually used only by CICS and IMS/DC.
- For subtask sharing, when a CI is not available (locked) to the task for the type of processing requested, VSAM under LSR and GSR buffering can manage the contention. For more information, see 4.4.10, “Share options” on page 144.
- Applications that use LSR can invalidate the buffer pool contents by using the MRKBFR macro. They can force the contents to be written immediately through the WRTBFR macro.

LSR relieves virtual storage constraint and reduces I/O for applications that access the same data multiple times. This technique is best used for truly random access, with multiple references to the same data. Subsequent access to data does not have to go through DASD.

With LSR (and GSR), the number and size of buffers are specified in BLDVRP macro (see Figure 4-8) and are not overridden by ACB or JCL. The buffer pool is identified by a number. A data set is connected to the buffer pool through its related ACB macro, where the buffer pool ID is specified. After all data sets that use a resource pool are closed, the resource pool can be deleted by issuing the DLVRP (delete VSAM resource pool) macro. For more information about macros, see *z/OS DFSMS Macro Instructions for Data Sets*, SC26-7408.


VSAM Shared Resources (LSR/GSR)

- Buffers and I/O related control blocks associated with a pool (BLDVRP)
- Multiple data sets share *pooled* resources

Figure 4-8 VSAM shared resources (LSR/GSR)

Online transaction processing (OLTP) applications like CICS and IMS/DC are the biggest users of LSR shared resources. They typically must have hundreds of data sets open in one address space at a time. Having an individual buffer pool for each data set would be a waste of virtual storage. With CICS and IMS, the BLDVRP is issued by the OLTP program, not directly by the user application. Information for building the shared buffer pool is specified in the product FCT table (for CICS). For more information, see the product manuals.

With LSR and GSR, writes can be deferred until VSAM needs a buffer to satisfy a GET request. Deferring writes saves I/O operations in cases where subsequent requests can be satisfied by the data already in the buffer pool. However, by deferring writes that you can save n-1 I/O operations, if the CI is modified n times. Also, the write requesting task is not placed in the wait state if the defer write option is active. For more information, see “Deferring write requests” on page 163.

The increased processor usage for searching a CI in the buffer pool is not affected by the pool size after the search is by hashing. Also, with LSR buffering, your application can use hiperspace as a second level of buffering.

If you intend to build an application by using LSR buffering techniques, follow these guidelines:

- LSR is suited for direct access. For sequential access, use NSR.
- Build resource pools before any Open to the data sets that use them.
- Build a separate resource pool for indexes to avoid index data being flushed by data that is being read.
- Use defer write if possible.

For more information about how to build an LSR resource pool, see *z/OS DFSMS Using Data Sets*, SC26-7410.

LSR with direct access

LSR buffering mode is designed for direct access. If your applications use NSR buffering and direct access and have performance problems, take advantage of LSR buffering techniques using SMB or BLSR.

For more information, see “System managed buffering (SMB)” on page 164, and “Batch local shared resources (BLSR)” on page 168.

LSR with sequential access

LSR buffering mode is designed for random access, not sequential. If you are relying on sequential read look-ahead (also called read-ahead) for good performance, LSR buffering might degrade your performance. LSR does not do read-ahead. Your sequential request is read in just one data CI per I/O operation. Use NSR buffering, where VSAM can read up to a CA worth of data CIs in a single I/O, when possible. For more information about NSR, see “NSR with sequential access” on page 154.

Guidelines: If you intend for your batch application to use the LSR buffering technique, do not use the LSR buffering technique. Perform these steps instead:

1. Write the application by using the default buffering NSR. It is easier and you can use a high-level language, such as COBOL.
2. Use SMB to convert to LSR (LRU algorithm) when processing or BLSR if there is no possibility of SMB.

Global shared resource (GSR)

GSR is similar to LSR buffering technique. GSR has the following differences from LSR:

- The buffer pool is shared among VSAM data sets accessed by tasks in *multiple* address spaces in the same MVS image.
- The resource pools are in CSA.
- The code that uses the buffering technique must be in supervisor state.
- Buffers cannot use hiperspace.
- The separate index resource pools are not supported for GSR.

GSR has the following *disadvantages*:

- GSR uses the common area, a potentially limited resource in the system.
- The address space that opened the data set for the first time must remain started to allow it can acquire new space for the data set.

- Only one BLDVRP TYPE=GSR may be issued for the system for each of the protection keys 0 through 7. The program that issues BLDVRP TYPE=GSR must be in supervisor state with protection key 0 to 7.
- All VSAM requests related to the global resource pool can be issued only by a program in supervisor state with the same protection key as the resource pool.
- With workload balancing, you *cannot* distribute the workload between system images and keep a single control block structure.
- GSR is not used by any of the main VSAM exploiters.

Guideline: Do not use GSR mode for buffering. Use RLS instead.

Deferring write requests

Specify defer write, if possible, in the ACB. With defer write, VSAM uses the buffer pool in a *store in* mode. That is, the updates are not propagated immediately to DASD. They are deferred until the WRTBFR macro is issued or until VSAM needs a buffer to satisfy an incoming GET request.

Deferring writes has the following performance advantages:

- If the same record is updated *n* times before it is written asynchronously to storage, you save *n-1* I/O operations. If there are no further updates, there are no saves.
- The application task that issued the write does not enter in wait state because an I/O operation.

The disadvantage of deferring writes is that VSAM does not have a log. If the system fails before the buffer destage, you lose some updates in your data set.

The Defer Write option is only valid for LSR and GSR. The defer write option is bypassed in LSR and GSR if SHAREOPTIONS 4 is specified. For more information, see 2.2, “Sharing VSAM data sets” on page 50. Deferring writes are also possible with NSR when modified by the SMB option. For more information, see “System managed buffering (SMB)” on page 164.

Specify that writes are to be deferred by coding MACRF=DFR in the ACB, along with MACRF=LSR or GSR as shown in Example 4-1.

Example 4-1 MACRF example

```
ACB      MACRF=({LSR|GSR},{DFR|NDF},...),...
```

The default is NDF.

When in RLS mode, deferred writes are ignored and direct request modified buffers are immediately written to disk and then optionally to the coupling facility.

Locating VSAM buffers above 16 MB

The *default* VSAM allocation for a resource pool is below the 16 MB line. The exception is SMB, where the default is above the 16 MB line. The location of the buffers and I/O control blocks can be controlled by using these techniques:

- In an assembler application program:
  - For NSR buffering, using the RMODE31 keyword in ACB
  - For LSR/GSR buffering, using the BLDVRP macro
- In the JCL, through RMODE31 (for NSR) parameter on the DD statement

You can specify the following values for RMODE31:

- ALL is used to allocate I/O relate control blocks and buffers above 16 MB. IDCAMS does not accept this value because it only accesses control blocks below the 16 MB line.
- BUFF is used only to allocate buffers above 16 MB.
- CB is used only for I/O relate control blocks above 16 M. IDCAMS does not accept this value because it only accesses control blocks below the 16 M line.
- NONE is the default. VSAM allocates I/O related control blocks and buffers below 16 M.
- With VSAM RLS, it is possible to define buffers as objects above the 2-G address line.

With the flexibility of JCL, you can avoid changing an existing application. But be aware that programs with AMODE=24 can abend with S0C4 when *locate* mode is used and RMODE31=BUFF is specified.

Locate mode is used when OPTCD=LOC is specified in the RPL. It indicates to VSAM to return to the application the address of the record, which is in the buffer. Using locate mode, programs that address 24 bits cannot access data above 16 M (addressing 31 bits).

There is also a *move* mode, where VSAM moves your logical record from the buffer to a user area that your program specifies.

The COBOL Enterprise product requires the following settings:

- Uses move mode. VSAM moves the record to an area whose address is indicated in the RPL, built by COBOL.
- Requires VSAM buffers above 16 M.

You can relieve the storage constraint below 16 M for application programs using VSAM data sets by specifying RMODE31 in the AMP parameter. You can increase the number of buffers with no effect on virtual storage below 16 MB. Do not forget to specify enough private area above the 16 MB line. For more information, see 4.4.12, “Region size” on page 146.

Guideline: Do not you use VSAM buffers above 16 MB.

System managed buffering (SMB)

SMB is available to all VSAM data sets types opened for NSR processing. SMB enables VSAM to determine the optimum number of index and data buffers, as well as the type of buffer management:

- LSR (LRU, lots of index buffers, lots of data buffers containing up to 20% of all data records, no look-ahead)
- NSR (discard the ones that are already processed, just one index buffer, look-ahead)

SMB allocates, at Open time, the optimum number of data and index buffers based on the type of access declared in the ACB’s MACRF parameter. It uses the BLDVRP macro to perform this function. Usually SMB allocates many more buffers than you would see without SMB. The installation then allows a larger virtual storage, making REGION equal to zero. While processing a VSAM data set by using SMB, when SMB switches to LSR mode, you can receive the message IEC161I 001(8,36)-087. This message is issued because of an error while building the VSAM resource pool (BLDVRP macro). It indicates that there is not enough virtual storage to satisfy the request that is done by SMB. SMB gets the available storage and processing goes on.

You can set the amount of virtual storage (as a limit) to be used by SMB buffering by using the SMBVSP parameter at DD ACCBIAS statement. Without this parameter, the first DD statement that indicates SMB to be opened gets lots of virtual storage and the others just a little.

Performance improvements can be dramatic with random access, particularly if you have little NSR buffering. SMB enlarges the number of buffers and switches from NSR (sequential) mode to LSR (LRU) mode as far as buffer pool managing is concerned.

You can relieve the use of storage below 16 MB by specifying that VSAM allocates buffers above 16 MB. For more information, see “Locating VSAM buffers above 16 MB” on page 163. However, for SMB the default is to allocate the buffer pool above the line.

SMB is available under the following conditions:

- The data set must be in extended format. To be in extended format, the data set must be System Managed Storage (SMS) and use a data class that is defined with DSNTYPE=EXT. For more information about extended format, see 2.3, “Extended format” on page 64.
- In the application program, ACB *must* be NSR, and the MACRF keyword cannot contain these items:
  - ICI data list: Improved control interval processing.
  - UBF: Management of I/O buffers left up to the VSAM user, as in DB2.

When these conditions are not satisfied, the job does not abend, the SMB services are not used, and no messages are issued.

Processing the data set through the alternate index of the path that is specified in the DDname is supported.

You can start SMB services by using these methods:

- Using a data class that is defined with RECORD_ACCESS_BIAS
- Specifying ACCBIAS in the AMP parameter of JCL DD statement

The order of precedence for specifying values is shown in the **Source** column in Table 4-2 on page 150.

To improve the VSAM performance, SMB must discover the most frequent type of access that will happen against such data set at open time. The MACRF type of access is just an *intention*. The real type of access is declared per I/O operation in the RPL. If you know in advance that the MACRF is not correct, provide SMB with the real type of access.

Also, in MACRF all the accessing types can be specified, which does not help SMB to discover the best buffer management method. This situation also happens in COBOL when *ACCESS MODE IS DYNAMIC* statement is used. It means all of them (direct, sequential, and skip sequential) are declared in the MACRF. To help SMB, the following options can be defined by the installation at ACCBIAS in the DD statement:

- SYSTEM: Have VSAM determine the buffering option based on these factors:
  - The ACB’s MACRF type of access intention: Sequential, direct, skip sequential, or a combination
  - The Storage Class specifications: Read or write bias for sequential or direct access at millisecond response (MSR) keyword.

  If you specify SYSTEM, you should:

  - Rely on the MACRF accessing options
  - Make sure that MACFR has only one access type
  - Ensure that the storage class bias parameters are correctly set

One of the following options in Table 4-6 is used by SMB, when ACCBIAS=SYSTEM: DO, DW, SO, or SW.

Table 4-6 Effects of ACB MACRF and storage class BIAS parameters

ACB MACRF parameters	***Storage class BIAS***			
	SEQ	**DIR**	**Both**	**None**
MACRF=DIR	DW	DO	DO	DO
MACRF=SEQ (default)	SO	SW	SO	SO
MACRF=(SEQ,SKP)	SO	SW	SW	SW
MACRF=SKP	DW	DW	DW	DW
MACRF=(DIR,SEQ) or (DIR,SKP) or (DIR,SEQ,SKP)	SW	DW	DW	DW
Note: DO=Direct Optimized; DW=Direct Weighted; SO=Sequential Optimized; SW = Sequential Weighted				

Those options can also be declared through the ACCBIAS.

- DO (Direct Optimized): SMB optimizes buffers management to direct access. SMB changes the buffering management to LSR (LRU) and allocates the adequate number of buffers for direct processing.
- DW (Direct Weight): To indicate to SMB that the processing is mixed direct and sequential, but with dominance of direct access. More index buffers are allocated to support direct processing, but some buffers are reserved for data to help with sequential processing. The buffering management technique is changed to LSR (LRU).
- SO (Sequential Optimized): Indicates to SMB to optimize buffer allocations for sequential processing. More data buffers (less index buffers) are allocated to support sequential access.
- SW (Sequential Weight): Specifies mixed processing, but with dominance of sequential access. SMB optimizes buffer handling for sequential processing by allocating more data buffers, but buffers are reserved for index to help direct access. SMB is even faster than NSR for sequential process.
- CO (Create Optimized): This option is not declared at ACCCBIAS. It is the most efficient option, as far as physical I/Os to the data component, for loading a VSAM data set. It applies only when the data set is in initial load status and when defined with the SPEED option. For more information, see “RECOVERY and SPEED options” on page 145. VSAM starts it internally. There is no user control other than the specification of RECORD ACCESS BIAS in the data class or an AMP=(ACCBIAS=) value of SYSTEM. The size of the data set is not a factor in the amount of storage that buffering requires. This option does not change the buffering implementation that the application specified (NSR). This technique requires a maximum of approximately 2 M of processor virtual storage for buffers, with the default above 16 M.
- CR (Create Recovery Optimized): This option is not declared at ACCCBIAS. VSAM uses this technique when a data set that is defined with the RECOVERY option is in initial load status. For more information, see “RECOVERY and SPEED options” on page 145. The Optimizing VSAM Performance system starts CR internally. There is no user control other than the specification of RECORD ACCESS BIAS in the data class or an AMP=(ACCBIAS=) value of SYSTEM. The size of the data set is not a factor in the amount of storage that buffering requires. This option does not change the buffering implementation that the application specified (NSR). This technique requires a maximum

of approximately 1 M of processor virtual storage for buffers, with the default above 16 M. In general, the Recovery option of initial load is not recommended.

- USER: Bypass SMB. This setting is the default if you code no specification for the ACCBIAS subparameter. This default is not used when the data class specifies RECORD_ACCESS_BIAS.

For direct optimization (DO), you can use the following DD AMP parameters (not present in the data class). These parameters tell the SMB by using LSR buffer manager how to handle the processing of the buffers:

- SMBVSP: Specifies the amount of virtual storage to obtain for buffers (calculating BUFND and BUFNI) when opening the data set. This parameter is used to override the default buffer space to be getmained. The buffer space is calculated assuming that 20% of the data records accounts for 80% of the accesses. Usually the default uses a large amount of virtual storage. The buffer space that is acquired is split across two LSR pools: One for the index and one for the data. The specification can be done in one of these formats:
  - SMBVSP=nnK
  - SMBVSP=nnM

  One performance problem for DO processing is too few index buffers being allocated by SMB. This can occur opening a data set that is small at first (at OPEN time) but grows as it is processed. Before DFSMS 1.11, the only solution was to close and open the data set to allow a new calculation. Since DFSMS 1.11, SMB takes 20% for BUFNI, which is more than the previous implementation.

  SMBVSP is the only option of the DD keyword ACCBIAS that can also be defined at data class. This implementation makes the modification of that parameter more flexible.

- SMBHWT: Used to allocate hiperspace buffers based on a multiple of the number of address space virtual buffers that have been allocated. It can be an integer from 0 to 99. However, the value specified is not a direct multiple of the number of virtual buffers that are allocated to the resource pool. Instead, it acts as a weighting factor for the number of hiperspace buffers to be established. The hiperspace size buffer is a multiple of 4K. These buffers can be allocated for the base data component of a sphere. If the CI size of the data component is not a multiple of 4K, both virtual space and hiperspace are wasted. The default is 0, which means that hiperspace is not used. Generally, do not use hiperspaces in z/Architecture.

- SMBDFR: Allows you to specify whether writing the data from a buffer to storage can be deferred until the buffer is required for another request or the data set is closed. A CLOSE macro started with the TYPE=T (temporary, does not need an OPEN to restart processing) option does not write the buffers to storage when LSR processing is used for direct optimization. The format is:

  ```
  SMBDFR=Y or N
  Y is the default for SHAREOPTIONS (1,3) and (2,3)
  N is the default for SHAREOPTIONS (3,3), (4,3) and (x,4)
  ```

To determine the final SMB processing technique and more resource-handling information that is related to SMB, examine the SMF type 64 records. This information is mainly for late performance analysis purposes because SMF type 64 records are gathered during the CLOSE processing of a data set.

Also, the JCL DD keyword MSG=SMBBIAS generates the IEC161I message that specifies the SMB buffering option that is selected by VSAM for each opened SMB buffering data set.

Retry capability for Direct Optimized technique (DO)

The retry capability belongs to the DO option only. It has the logic that is associated when the optimum amount of virtual storage is not available at the user region. The retry capability is then able to perform these tasks:

1. Obtain storage buffers (through a conditional Getmain) to about 20% of data set records plus to buffer the entire index component, if the VSAM organization is KSDS.
2. If the Getmain fails, VSAM reduces the numbers of buffers by 50% from the optimum amount for the data components and retry.
3. Reduce the number of *data buffers* to the minimum and retry. The minimum size is 1 MB.
4. Reduce the number of *index buffers* to the minimum and retry. The minimum is the amount of virtual storage to contain the entire index set plus 20% of the sequence set records.

If none of the retries are successful, change the technique to Direct Weighted (DW).

Messages that are related to SMB

The following messages are related to SMB:

- IEC161I 001(sfi)-032:

  sfi is 101, 102 or BLDVRP return and reason code. For BLDVRP return codes, see “Return Codes from Macros Used to Share Resources Among Data Sets” in *z/OS DFSMS Macro Instructions for Data Sets,* SC26-7408.

- IEC161I 001(sfi)-033: means that SMB did not return ACCBIAS.
- IEC161I 001(sfi)-034: sfi is for the CATALOG LOCATE ERROR.

SMB support for AIX

VSAM alternate index data sets that use SMB might have better performance. SMB supports data sets with alternate indexes for the DO option. SMB does a good job of sizing the resources that are needed for processing when not using DO.

Using SMB, the buffer allocation and buffering technique used depends on these factors:

- Order in which these components are opened:
  - Base Cluster
  - Path Cluster
- ACCBIAS option as specified by the user or detected by SMB

During OPEN time, the decision to share a control block structure is based on DSN and DDN sharing criteria. For more information about sharing criteria, see 2.2.5, “Sharing data in a single VSAM control block structure” on page 58.

Batch local shared resources (BLSR)

BLSR is another solution for the NSR random access performance problem. In general, the SMB implementation is superior to BLSR. BLSR is an MVS subsystem that provides performance advantages for an application that uses VSAM NSR buffering techniques to switch to LSR (LRU) buffering. It provides these advantages without changing the application source code or link-editing the application again. Only a JCL change is required.

Your application performance improves using BLSR when direct access is used and the same CI is referenced more than once during processing. Using the BLSR subsystem with sequential access can degrade performance rather than improve it. For information about how LSR works, see “Local Shared Resources (LSR)” on page 160.

For mixed processing (some direct, some sequential), you can benefit from using BLSR. If the amount of data to be processed sequentially is not large, you can compensate for the lack of read-ahead (at BLSR) by using a large data CI size.

BLSR supports the VSAM organization data set types KSDS, ESDS, RRDS, and VRRDS. Using BSLR, you can force VSAM buffers and control blocks to be located above 16 MB without having to use hiperspace. It also allows the buffer pool (up to 16) to be shared among several VSAM data sets. You can also use hiperspace and restrict its use with RACF or an equivalent security software.

Starting BLSR

To start BLSR subsystem services, add the SUBSYS parameter to the specific DD statement at JCL.

Before you can use BSLR, its subsystem must be installed. Contact your installation system programmer, or see MVS Batch Local Shared Resources, GC28-1469.

When the BLSR subsystem is installed, start its services by adding the SUBSYS parameter to the JCL.

When the application opens the VSAMDD ACB, the BLSR subsystem completes the conversion to LSR processing.

Specify the SUBSYS parameter as shown in Example 4-2.

Example 4-2 SUBSYS parameter specification

```
SUBSYS=(subsys-name,'DDNAME=value','subparm1=value',....,'subparmn=value')
```

Where:

- *subsys-name* is the name that is given to the BLSR subsystem, usually BLSR.
- *DDNAME=value* is the name of the DDNAME to be open by the application program.

The following sub parameters are allowed on the BLSR SUBSYS parameter: BUFND, BUFNI, HBUFND, HBUFNI, RMODE31, STRNO, DEFERW, SHRPOOL, BUFSD, BUFSI, and MSG.

The following system parameters are not allowed with the SUBSYS parameter: *, AMP, BURST, CHARS, COPIES, DATA, DDNAME, DYNAM, FLASH, MODIFY, QNAME, SPACE, and SYSOUT. You must nullify any of these parameters if they are specified on a DD statement that you are overriding.

The following example illustrates how to start BLSR. Example 4-3 on page 169 shows a data set the application opens for NSR processing. It is converted to BLSR processing.

Example 4-3 DD statement for BLSR example

```
//VSAMDD DD DISP=SHR,DSN=VSAMDSN
```

You can convert to the BLSR subsystem by performing these steps:

1. Copy the statement in Example 4-3 and change the DDNAME on the JCL statement from VSAMDD to NEWBUFF as shown in Example 4-4.

   Example 4-4 DDNAME change to reflect use of BLSR

   ```
   //NEWBUFF DD DISP=SHR,DSN=VSAMDSN
   ```

2. Add the DD statement that is shown in Example 4-5. The SUBSYS subsystem-name subparameter is BLSR and the SUBSYS DDNAME subparameter is the DD name that was selected in Step 1.

Example 4-5 BLSR DD statement

```
//VSAMDD DD SUBSYS=(BLSR,'DDNAME=NEWBUFF')
```

When SUBSYS is specified in JCL, the job's INITIATOR issues the IEFSSREQ macro, invoking BLSR subsystem services, passing the information specified in the SUBSYS parameter.

Then, the BSLR subsystem:

1. Includes an EXIT to call BLSR at OPEN.
2. Dynamically allocates the data set to the ACB defined DDNAME (`VSAMDD` in the example).

BLSR restrictions

BLSR can be used with SMS and non-SMS-managed data sets.

If your data set is SMS-managed and is in extended format, you get better performance by using SMB. For more information, see "System managed buffering (SMB)" on page 164.

The BLSR subsystem has the following restrictions:

- The ACB cannot be above 16 MB. Otherwise, the system fails the OPEN request with error message IEC190I.
- If the application closes and then reopens the ACB without refreshing the DDNAME, the request is bypassed. The data set is opened using the same options as the last time it was opened. That is, if the data set was previously opened for LSR processing, it is reopened for LSR. Similarly, if the data set was not eligible for LSR processing (through BLSR) the first time, it is reopened for NSR processing. This process occurs even if LSR is now applicable.

High-level languages refresh the DDNAME for each open. Therefore, the BLSR subsystem is always called for these programs:

- COBOL/VS programs that use the ISAM interface to access VSAM data sets
- PL/I programs that were written for ISAM (an old access method) accessing VSAM data sets
- Other programs that use the RDJFCB macro to try to identify the file type, such as IDCAMS

Guidelines: Use SMB for better results. If possible, locate the buffers above 16 MB.

4.4.16 Data compression

Compressing data means to keep data in a format that requires less space than the original data. To compact data has the same objective of compressing, but is simpler. Compacting just gets rid of blanks and zeros. There are quite a few methods (algorithms) to compress data:

- Character-based methods: Huffman, Run-length encoding, Ziv-Lempel
- Bit-level methods: Image data compression, IDRC (used in tape controllers)

Some of these methods use the concept of a dictionary, which is a mapping from one vocabulary to another. There are two types of dictionaries, where one is the inverted list of the other:

- Compression dictionary that is created along compression
- Expansion dictionary that is used for extracting

The same method can use many dictionaries.

z/OS uses the Ziv-Lempel (ZL) algorithm that is installed in z/Architecture processors for data compression and decompression. ZL-based schemes work by entering phrases into a dictionary. When a repeat occurrence of that particular phrase is found, they output the dictionary index (a small set of characters) instead of the phrase. Several compression algorithms are based on this principle. They differ mainly in the manner in which they manage the dictionary, and all of them tend to perform much better in decoding than in encoding. z/Architecture compression uses the ZL1 version of the ZL implementation. In a z196, a type of inbound co-processor runs the ZL1 algorithm, together with some cryptographic protocols. Each PU chip has two such co-processors. Each one is shared by two PUs. They are activated by the PU when the instruction Compression Call is run by the PU.

There are a few DASD controllers in the market able to compress by using derivations of the ZL implementation.

Double compressing the same data set just enlarges it. Do not do that.

Where to compress data

Compression can be run in the processor (PU) or outbound in an I/O controller (for example, tape). Processor compression saves the following resources:

- I/O buffer pool virtual and real memory
- Channel cycles
- Controller cache space
- Controller internal data path cycles
- Media space (disk and tape)
- Transmission line cycles (in the network)

Compressing in the controller saves these resources:

- Controller cache space
- Controller internal data path cycles
- Media space (disk and tape)
- CPU cycles (important)

Compression service activation

Compression service activation covers checking and setting up the required compression environment, usually during the IPL process of the z/OS. It has the following functions:

- Determines whether the Compression Call instruction is available in the processor. If it is not, the compression is done by software emulation, which is slow and consumes lots of processor cycles. All the modern z processors are able to runs this instruction.
- Verifies whether the SYS1.DBBLIB data set is available. It contains dictionary building blocks (DBB) to be used for compression.

Compression management services (CMS)

CMS covers the actual process a data set goes through when compression is requested by an application. CMS provides and carries out the following services that are used by VSAM:

- VSAM Candidate data set verification
- Dictionary selection
- Forms of candidate data sets

VSAM Candidate data set verification

VSAM Candidate data set verification has the following requirements:

- KSDS organization only. Only data are compressed. The indexes and AIXs are not compressed. For ESDS and RRDS data set types, data compression cannot be tolerated. These organizations do not support the change of a logical record size after it was initially stored. Data compression can produce different logical record size at update in place, depending on its contents.
- Data set must be SMS-managed and in extended format. The data class that is assigned must then specify the DSNTYPE=EXT with a required COMPACTION=Y (blank defaults to N) keyword.
- Have a primary allocation of at least 5 MB (data component only) because of the amount of sampling needed to develop a dictionary token. If no secondary allocation is specified, the primary allocation must be at least 8 MB. The idea is to not compress small data sets.
- Have a minimum record length of 40 bytes (not including the key length).
- CI mode processing is not allowed.
- BCS catalog, system data sets, and temporary data sets cannot be compressed because extended format is not supported.

For VSAM extended format and compression restrictions and incompatibilities, see *DFSMS Using Data Sets*, SC26-4922.

Dictionary selection

There are two general forms of compression, generic and tailored, as shown in Figure 4-9 on page 173.

- Generic compression

  This compression technique involves the sampling and interrogation of the compression eligible data set by CMF. A sample from the data set is taken at load time, and compared to the DBBs for similar content and compression efficiency. Up to 64 KB of the data set can be sampled and written before the data set starts to be compressed. The first time a compressed format data set is written to disk, the first bytes are written in non-compressed form.

  Because the dictionary is assembled by using DBBs during the sampling and interrogation, the dictionary does not exist when the first bytes of the data set are written. After the dictionary is built, the data can use this dictionary to compress the rest of the data set. Information about the mix of DBBs selected during the sampling and interrogation process is kept in the catalog. This information enables the decompression of the data set when required, but does not require a dictionary to be stored with the data set.

- Tailored compression

  Tailored compression introduces a new form of compression for sequential extended format data sets. However, VSAM data sets do not support tailored compression.

With tailored compression, the CMF attempts to derive a compression dictionary that is tailored specifically to the initial data written to a data set. After a tailored dictionary is derived, it is embedded in the compressed data set. The dictionary must be stored with the compressed data. Otherwise, the data cannot be extracted. This technique is expected to provide improved compression ratios, reducing storage usage and channel traffic.

Because the dictionary is tailored to the user data, more data is sampled than is required by generic compression. The process of sampling the data and building the dictionary during creation of a new data set therefore takes more processor cycles. This difference is most noticeable when compressing small data sets. For larger data sets, the cost of sampling is amortized significantly. Reuse of a tailored compressed data set saves the cost of sampling and dictionary creation.

An application has the option of either using the new tailored compression or continuing to use generic DBB-based compression. Remember that VSAM uses only generic compression. Tailored compression allows for more types of data to be compressed, for example non-English languages.

The original generic dictionary was developed from standard American-English scripts. Therefore, files that contain sequences of characters that do not appear in the American scripts do not compress well. The tailored dictionary avoids this problem because it is generated dynamically from the data itself. Dynamically generating a tailored dictionary from the data itself therefore makes tailored compression more useful to the non-English-speaking world.

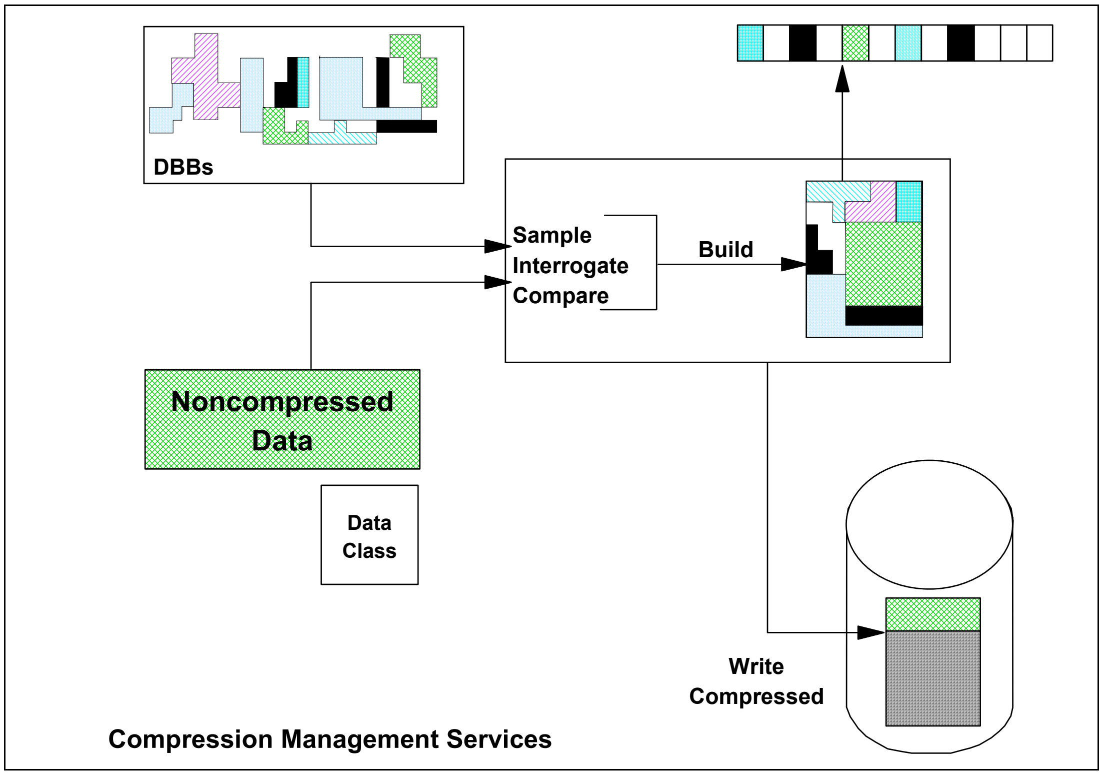


Figure 4-9 CMS dictionary selection

Forms of candidate data sets

A compression-eligible data set can exist in one of these forms:

- Dictionary selection form: The data set is eligible, but its makeup is yet to be determined. The dictionary tokens are being compared with the data set contents during interrogation and sampling. Interrogation maps bytes into alphabetic, numeric, upper-, or lowercase, and sampling evaluates DBB compression efficiency.
- Mated form: The data set is mated with the appropriate dictionary. This form concludes the sampling and interrogation processes. It is the most common form for a compressed data set.
- Rejected form: A suitable dictionary match was not found during sampling and interrogation. Compression is bypassed, or for a sequential data set, the data set was closed before a token could be selected. Note that the data set is in compressed format.

There are some types of data sets that are not suitable for ZL1 data compression, resulting in rejection:

- Small data sets.
- Data sets in which the pattern of the data does not produce a good compression rate, such as image data and seismic data.
- Small records. Compression is performed on a logical record basis. When logical records are short, the cost of more work might become excessive compared to the reduction in size that compression achieves.
- Logical records with the primary key with a high off set. The data compressing is run to the right of the primary key.

Data compression and extraction

Data compression is done at logical record level when the application writes this logical record. Data decompression is done at logical record level, when the application reads this logical record in a mated data set. A mated data set is one that has acquired a suitable dictionary token through successful interrogation and sampling processes. In other words, suitable building blocks have been found and selected from the DBB distribution library. This combination constitutes the dictionary token that is associated with the data set and held in the catalog entry. Compression and extraction of a mated data set use the dictionary built from the dictionary token associated with the data set entry in the extended format cell of the catalog.

The dictionary is customized to the data set through the dictionary selection process. However, the dictionary is not stored with the data. Only the token information is stored. The dictionary does not need to be stored because it is a table that is dynamically built in storage from the token information.

Using this approach, DFSMS retains responsibility for dictionary management and shields the user and application from the physical representation of compressed data on disk.

Compression facts

Compression has the following basic facts:

- IDCAMS Import into an empty data set does not propagate extended format and compression information.
- When data are compressed, the length of a stored record might change after an update without any logical record length change.

- Locate mode (the application that accesses the logical record in the CI buffer) processing is allowed, but requires a larger number of internal work areas to process.
- ISMF adds a *%user data reduction* (compression factor) field in data set application window.
- IDCAMS REPRO copies data sets without extracting:
  - If the target data set is eligible for compression
  - If the target device is a like device, or CI sizes are equal

  Otherwise, REPRO extracts the data while reading and optionally compresses the data while writing.
- LISTCAT lists the compression-related information. The REXX code that is described in Appendix A, “Sample code” on page 381 displays compression information as well.
- IEHLIST indicates that the data set is compressed.
- DFSMSdss functions and compression have these issues:
  - Logical dump: This process does not change format from compressed to non-compressed or vice versa, and includes cataloging information with the dump.
  - Logical restore: This process does not change format from non-compressed to compressed or vice versa.
  - Logical copy: This process never changes the format from non-compressed to compressed or vice versa.
  - DFSMShsm avoids double compression.

VSAM compression

VSAM compression is done transparently to the application, and is required through the data class (DC) parameter in SMS data sets. The DC assigned to the data set must specify the following options: DSNTYPE=EXT with a required COMPACTION=Y. Figure 4-10 shows the ISMF list of the DC.

```
DATACLAS                     EXTENDED                    MEDIA
NAME     DATA SET NAME TYPE  ADDRESSABILITY COMPACTION TYPE
--(2)--- -------(26)-------  -----(27)-----  ---(28)---  -(29)-
DCSMB    EXTENDED REQUIRED   NO              ----        ------
DCSTRIPE EXTENDED REQUIRED   NO              ----        ------
DCXXXX   EXTENDED REQUIRED   NO              ----        ------
DIRECT   ------------------  NO              ----        ------
ECCST    ------------------  NO              YES         MEDIA2
EHPCT    ------------------  NO              YES         MEDIA3
```

Figure 4-10 ISMF Data Class List display

As already mentioned, VSAM compression applies only to KSDS in extended format. All the fields to the left of the key in the logical record are not compressed.

Compression affects the catalog, VTOC, and SMF information about VSAM data sets. For more information about how to get this data, see “Compression information sources” on page 176.

Compression performance

The relative processor compression cost depends on these factors:

- Dictionary size

  Usually, you do not have much control over this factor.

- Ratio of reads to writes of compressed records

  Ziv-Lempel extracts (expands) faster (fewer processor cycles) than compresses. It is therefore better suited for data sets with a high read-to-write ratio.

- The number of I/O operations that are avoided because less data to transfer can compensate for processor usage during compression. Each I/O operation consumes several thousand instructions.

Compression information sources

Compressed data sets have specific information about the compression process in different sources:

- Catalogs

  ICF catalogs are not eligible for compression, but are enhanced to contain more information about compressed mated data sets in the extended format cell of the catalog entry. The extended format cell is part of the VSAM volume data set (VVDS) component of the ICF catalog.

  The extended format cell holds the following information:

  – Number of stripes (STRIPE-COUNT)
  – Compression flags (COMP-FORMAT)
  – Physical block size
  – Non-compressed user data set size in bytes (USER-DATA-SIZE)
  – Compressed user data set size in bytes (COMP-USER-DATA-SIZE)
  – Active dictionary token (ACT-DIC-TOKEN: Active Dictionary Token or NULL)
  – If user data sizes are valid (SIZES-VALID)
  – Compression characteristic record

  Appendix A, “Sample code” on page 381 contains a REXX routine to search in a catalog for all compressed data sets or a specific one. It shows the compression ratio for each data set. This tool allows you to determine whether your processor cycles are efficiently used in terms of saving storage media, channel cycles, and buffer pools. For more information, see “SMFLSR sample program” on page 395.

- SMF

  New fields have also been added to SMF type 64 records for VSAM compression. These new fields record the size of the data before and after compression, flags for extended format and compression, and dictionary tokens that are used for compression. For more information about the SMF record layout, see *z/OS MVS System Management Facilities (SMF),* SA22-7630.

  QSAM compression information in the type 14 and 15 records is updated when CLOSE is run on a sequential compressed format data set. Information about the dictionary token selected and the compressed and non-compressed data set size is added to the SMF type 14 and 15 records.

  This information is also maintained in the extended format cell in the catalog.

- VTOC

There is now a compression indicator, DS1COMPR, and an extended format data set indicator, DS1STRP, in the VTOC. Because a compressed format data set must be an extended format data set, both indicators are on for compressed format data sets:

– VTOC Format 1 DSCB
– DS1STRP: Extended format VSAM data set. Contained within the DS1SMSFG offset x’ 4 E’
– DS1COMPR: Compressed data set. Contained within the DS1FLAG1 offset x’ 3 D’

Guidelines:

- Do not use compression for data sets with low read-to-write ratio (less than 60%).
- Set the key offset equal to zero.

4.4.17 VSAM data striping

Usually, in a multi-extent, multi-volume VSAM data set that is processed in sequential access mode, there is no of parallelism for I/O operations among the volumes.

Even parallel access volume (PAV) is not able to have parallel sequential writes I/Os because of integrity reasons. For reads, the access method does not usually send several I/Os to the same volume, so PAV also is not able to make them parallel.

When a sequential I/O operation is run for an extent in a volume, no other I/O activity from the same task and same data set is scheduled to the other volumes. When I/Os are the major bottleneck, and there are available resources in the channel subsystem and controllers, it is a waste of these resources.

Data striping addresses this performance problem by imposing two modifications to the traditional data organization:

- The records are not placed in *key ranges* (a set of logical records with their keys in sequence) along the volumes. Instead, they are organized in stripes.
- Parallel I/O operations are scheduled by VSAM to sequential stripes in different volumes.

By striping data, the tracks in the case of SAM, and the control intervals (CIs) for VSAM are spread across multiple devices. This format allows a single application request for records in multiple tracks and CIs to be satisfied by sequential concurrent I/O requests to multiple 3390 volumes.

The result is improved performance by achieving data transfer into the application at a rate greater than any single I/O path. The scheduling of I/O to multiple devices to satisfy a single application request is referred to as an *I/O packet*.

Guideline: Use data striping to speed up batch sequential access.

VSAM data striping topics

Data striping support is provided for all VSAM organization, including KSDS, ESDS, RRDS, VRRDS, and LDS.

The idea is to spread sequenced data CIs in different data CAs in different volumes. The following are the characteristics of a VSAM striped data set:

- Striping is done by CI, as opposed to a track for SAM.
- A stripe is associated with a single volume or a set of volumes (if you have multi-layer implementation).
- Only the data component of a base cluster can be striped.
- A data set can have a maximum of 16 stripes.
- A stripe on a single volume has a maximum of 123 extents per stripe.
- A multi-layered stripe (a stripe that is allocated on several volumes) has a maximum of 255 extents per stripe.
- The following features are supported in a striped VSAM data set:
  - Extended addressability (data set size greater than 4 GB)
  - Compressed format
  - Partial release
  - Reused
- The data set must be system-managed.
- The data must be in extended format.
- If the data set has the GUARANTEED SPACE option, there is no support for multi-layered. For GUARANTEED SPACE, the number of stripes is equal to the number of volumes (VOLUME COUNT) you specify in the data class or the VOLUMES parameter in JCL. It has a maximum of 16 stripes. The JCL specification overrides the data class.
- For non-guaranteed space, SMS determines the number of stripes to use based on the value of the SUSTAINED DATA RATE(SDR) keyword that is defined in the storage class.

Figure 4-11 shows an example of a four-stripe VSAM data set.

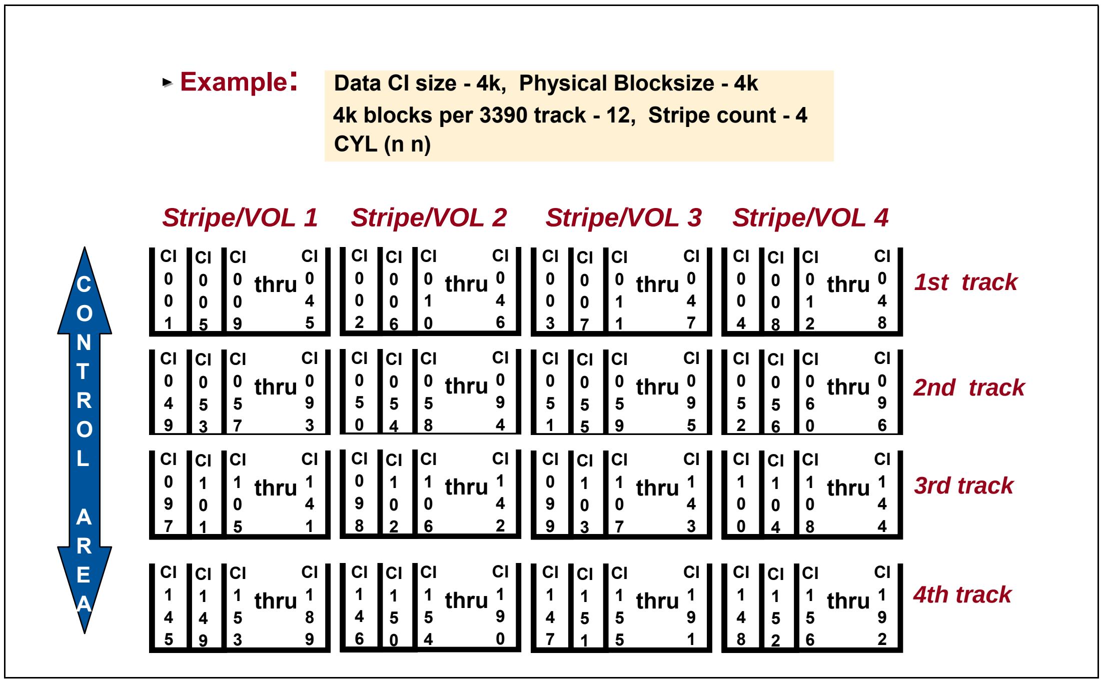

Example:
Data CI size - 4k, Physical Blocksize - 4k
4k blocks per 3390 track - 12, Stripe count - 4
CYL (n n)
Stripe/VOL 1
Stripe/VOL 2
Stripe/VOL 3
Stripe/VOL 4
CONTROL AREA
CI 001, CI 005, CI 009 thru CI 045
CI 002, CI 006, CI 010 thru CI 046
CI 003, CI 007, CI 011 thru CI 047
CI 004, CI 008, CI 012 thru CI 048
1st track
CI 049, CI 053, CI 057 thru CI 093
CI 050, CI 054, CI 058 thru CI 094
CI 051, CI 055, CI 059 thru CI 095
CI 052, CI 056, CI 060 thru CI 096
2nd track
CI 097, CI 101, CI 105 thru CI 141
CI 098, CI 102, CI 106 thru CI 142
CI 099, CI 103, CI 107 thru CI 143
CI 100, CI 104, CI 108 thru CI 144
3rd track
CI 145, CI 149, CI 153 thru CI 189
CI 146, CI 150, CI 154 thru CI 190
CI 147, CI 151, CI 155 thru CI 191
CI 148, CI 152, CI 156 thru CI 192
4th track

Figure 4-11 Striped VSAM data set

As you can see, a data control area is formed by CIs that is not within the sequenced keyrange. Also, for a KSDS, index records in the sequence set that is contained in the same CI index point to different data CAs.

Layering and striped data

VSAM supports multi-layering. A layer in a striped environment is defined as the relationship of the volumes that make up the total number of stripes. That is, those volumes that participate as part of an I/O packet.

After the stripe extends to a new volume, and the I/O packet changes, this constitutes another layer. The sequential access method (SAM) is restricted to extending only to the current stripe volume and does not support the concept of multi-layering.

Figure 4-12 shows an example of layering with a three-stripe data set.

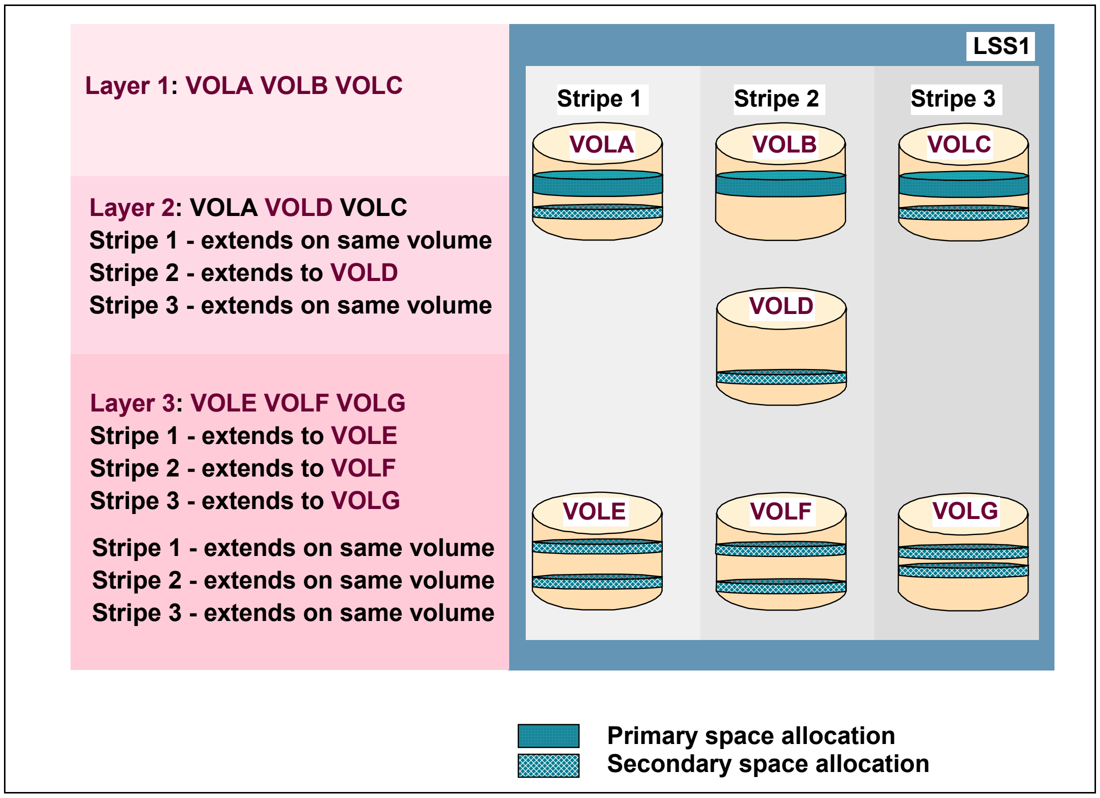

LSS1
Stripe 1
Stripe 2
Stripe 3
VOLA
VOLB
VOLC
VOLD
VOLE
VOLF
VOLG
Layer 1: VOLA VOLB VOLC
Layer 2: VOLA VOLD VOLC
Stripe 1 - extends on same volume
Stripe 2 - extends to VOLD
Stripe 3 - extends on same volume
Layer 3: VOLE VOLF VOLG
Stripe 1 - extends to VOLE
Stripe 2 - extends to VOLF
Stripe 3 - extends to VOLG
Stripe 1 - extends on same volume
Stripe 2 - extends on same volume
Stripe 3 - extends on same volume
Primary space allocation
Secondary space allocation

Figure 4-12 Layering in VSAM data set striping

Implementing VSAM data striping

To create a striped VSAM data set, perform the following steps:

1. Define an SMS-managed VSAM data set in extended format.

   Specify *Data Set Name Type* in the data class. *Data Set Name Type* can be set to EXT Preferred. In this case, SMS tries to allocate in a controller that supports extended format. All modern controllers support extended format. The data set is not defined as striped if it is not allocated in extended format.

2. Specify the Sustained Data rate (SDR) parameter in the storage class for either guaranteed or non-guaranteed space. This setting tells SMS that you want to implement striping.

3. For *guaranteed space*, specify the VOLUME COUNT in the data class or the VOLUMES parameter in JCL. The JCL specification overrides the data class.

   VOLUME COUNT represents the number of volume cells that are associated with the data set in catalog entries. If more volumes are required, the data set access must be closed, and the IDCAMS ALTER ADDVOLUMES command must be issued. MAXIMUM VOLUME COUNT data class parameter represents actual maximum number of volumes an SMS-managed VSAM cluster can span. Volumes can be added dynamically to the cluster without manual intervention and without taking the application down.

   If you do not specify either the volume count or the volume serial numbers, you only get a single stripe.

4. For *non-guaranteed space*, specify the SDR value in the storage class. SMS computes the number of stripes based on an old rule that each 3390 volume delivers a 4-MBps rate. The volume count is ignored. Then, if you specify 16 MBps, VSAM generates a 4-stripe data set.

   As a simple example: if you specify an SDR of 17 MBps, your data set is defined with four stripes.

Examples of defining striped data sets

Figure 4-13 shows a JCL to create a striped VSAM data set that uses guaranteed space, and a specified volume count of 4. The data class name is KEYEDEXG, and the storage class is STRIPE.

```
//STRIPED  JOB 'DEF STRIPED-EF VSAM DS',MSGCLASS=X,NOTIFY=&SYSUID
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
           DELETE DAWN.KSDSEXG
           DEFINE CLUSTER -
              (NAME(DAWN.KSDSEXG)  -
              DATACLASS(KEYEDEXG) -
              STORAGECLASS(STRIPE))
           LISTC ALL ENTRIES(DAWN.KSDSEXG)
/*
```

Figure 4-13 Sample JCL to create a striped data set

The content of the KEYEDEXG data class is shown in Figure 4-14.

```
CDS Name  . . . : SYS1.SMS.SCDS
Data Class Name : KEYEDEXG

Description : USING GUARANTEED SPACE

Recorg . . . . . . . . . : KS
Recfm  . . . . . . . . . .:
Lrecl. . . . . . . . . . : 300
Keylen . . . . . . . . . : 8
Keyoff . . . . . . . . . : 0
Space Avgrec . . . . . . : K
      Avg Value  . . . . : 300
      Primary  . . . . . : 600
      Secondary  . . . . : 100
      Directory  . . . . :
Retpd Or Expdt . . . . . :
Volume Count . . . . . . : 4
  Add'l Volume Amount  . :
Imbed  . . . . . . . . . :
Replicate  . . . . . . . :
CIsize Data  . . . . . . : 4096
% Freespace CI . . . . . : 10
            CA . . . . . : 10
Shareoptions Xregion . . :
             Xsystem . . :
Compaction . . . . . . . :
Media Interchange
  Media Type . . . . . :
  Recording Technology :
Data Set Name Type  . . . : EXTENDED
  If Extended . . . . . . : REQUIRED
  Extended Addressability : NO
  Record Access Bias  . . : USER
Reuse . . . . . . . . . . : NO
Initial Load  . . . . . . : RECOVERY
Spanned / Nonspanned  . . :
BWO . . . . . . . . . . . :
Log . . . . . . . . . . . :
Logstream Id  . . . . . . :
Space Constraint Relief . : NO
  Reduce Space Up To (%)  :
```

Figure 4-14 Sample content of KEYEDEXG data class

The content of the STRIPE storage class is shown in Figure 4-15.

```
CDS Name  . . . . . : SYS1.SMS.SCDS
Storage Class Name  : STRIPE
Description  : TO BE USED FOR STRIPING TEST

Performance Objectives
  Direct Millisecond Response  . . . :
  Direct Bias  . . . . . . . . . . . :
  Sequential Millisecond Response  . :
  Sequential Bias  . . . . . . . . . :
  Initial Access Response Seconds  . :
  Sustained Data Rate (MB/sec) . . . : 17
Availability . . . . . . . . . . . . : NOPREF
Accessibility  . . . . . . . . . . . : NOPREF
  Backup . . . . . . . . . . . . . . :
  Versioning . . . . . . . . . . . . :
Guaranteed Space . . . . . . . . : YES
Guaranteed Synchronous Write . . : NO
Cache Set Name . . . . . . . . . :
CF Direct Weight . . . . . . . . :
CF Sequential Weight . . . . . . :
```

Figure 4-15 Sample content of STRIPE storage class

Figure 4-16 shows the job output. The data set is explicitly defined with four stripes with the specified volume count of 4.

```
DEFINE CLUSTER -
              (NAME(DAWN.KSDSEXG)  -
              DATACLASS(KEYEDEXG) -
              STORAGECLASS(STRIPE))
IGD17070I DATA SET DAWN.KSDSEXG ALLOCATED
SUCCESSFULLY WITH 4 STRIPE(S).
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX30 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME MHLV14 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX31 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX28 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SBOX30 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME MHLV14 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SBOX31 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SBOX28 IS 0
IDC0512I NAME GENERATED-(D) DAWN.KSDSEXG.DATA
IDC0512I NAME GENERATED-(I) DAWN.KSDSEXG.INDEX
IDC0181I STORAGECLASS USED IS STRIPE
IDC0181I MANAGEMENTCLASS USED IS MCDB22
IDC0181I DATACLASS USED IS KEYEDEXG
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure 4-16 Sample IDCAMS control statements and output for four stripes

Figure 4-17 shows how to create a striped VSAM data set that uses non-guaranteed space and SDR 17 MBps.

```
//STRIPED  JOB 'DEF STRIPED-EF VSAM DS',MSGCLASS=X,NOTIFY=&SYSUID
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
           DELETE DAWN.KSDSEXT
           DEFINE CLUSTER -
              (NAME(DAWN.KSDSEXT)  -
              DATACLASS(KEYEDEXT) -
              STORAGECLASS(STRIPE))
           LISTC ALL ENTRIES(DAWN.KSDSEXT)
/*
```

Figure 4-17   Sample striped VSAM data set using non-guaranteed space

The content of the STRIPE storage class using non-guaranteed space is shown in Figure 4-18.

```
CDS Name  . . . . . : SYS1.SMS.SCDS
Storage Class Name  : STRIPE
Description  : TO BE USED FOR STRIPING TEST
Performance Objectives
  Direct Millisecond Response  . . . :
  Direct Bias  . . . . . . . . . . . :
  Sequential Millisecond Response  . :
  Sequential Bias  . . . . . . . . . :
  Initial Access Response Seconds  . :
  Sustained Data Rate (MB/sec) . . . : 17
Availability . . . . . . . . . . . . : NOPREF
Accessibility  . . . . . . . . . . . : NOPREF
  Backup . . . . . . . . . . . . . . :
  Versioning . . . . . . . . . . . . :
Guaranteed Space . . . . . . . . . : NO
Guaranteed Synchronous Write . . : NO
Cache Set Name . . . . . . . . . :
CF Direct Weight . . . . . . . . :
CF Sequential Weight . . . . . . :
```

Figure 4-18   Example of content of STRIPE storage class

Figure 4-19 shows the job output of the second example. The data set is defined with five stripes. SMS is allocated on five volumes out of the eight volumes that are defined in the storage group. Only the data component is striped.

```
DEFINE CLUSTER -
              (NAME(DAWN.KSDSEXT)  -
              DATACLASS(KEYEDEXT) -
              STORAGECLASS(STRIPE))
IGD17070I DATA SET DAWN.KSDSEXT ALLOCATED
SUCCESSFULLY WITH 5 STRIPE(S).
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX31 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX28 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX29 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME SBOX30 IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME MHLV13 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SBOX31 IS 0
IDCAMS  SYSTEM SERVICES
IDC0512I NAME GENERATED-(D) DAWN.KSDSEXT.DATA
IDC0512I NAME GENERATED-(I) DAWN.KSDSEXT.INDEX
IDC0181I STORAGECLASS USED IS STRIPE
IDC0181I MANAGEMENTCLASS USED IS MCDB22
IDC0181I DATACLASS USED IS KEYEDEXT
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure 4-19   Second example with five stripes

After you load the data set, the LISTCAT output shows the HURBA only on the first volume. For the other volumes, HURBA=0.

CA size considerations in data striping

CA size calculations are affected by striping. Normally, the CA size is the lesser of the primary or secondary value. It must not exceed 15 tracks if allocation is in tracks, or a cylinder if allocation is in cylinders.

In striped data sets, the CA size amount must be a multiple of the number of stripes. That is, a CA cannot end in the middle of a stripe.

To meet this restriction, the CA size might have to be rounded to the next integral of the stripe count. Also, because the maximum stripe count is 16, a CA size of 16 tracks must be allowed to accommodate 16 stripes. The CA size maximum is increased to 16 tracks from the previous one cylinder (15 tracks) allocation.

For striped data sets, all computations for CA size are performed by using the equivalent number of tracks.

Performance considerations

Consider the following points when you use VSAM striped data sets:

- The effective throughput gains on sequential access increases almost in proportion to the number of stripes, unless there is contention at the device, DASD controller, path, or channel level.
- The throughput gains decrease if you work with many stripes, or the Law of Diminishing Returns starts to work because of an increase in management. For KSDS, the processing improvement curve flattens after the fourth stripe. You might not get performance improvements if you define more than four stripes.

- The I/O response time for a striped data set is equal to the longest I/O in the stripe. The effective throughput is limited to the slowest stripe.
- The I/O post to the application is done when all the I/O to the stripes ends. If an I/O error occurs in one of the stripes/volumes, the I/O operation ends with an error.
- For striped data sets, use SMB to determine the number of buffers, or allocate a larger value for the BUFND, depending on your application.

  If you are not using SMB, specify a BUFND value that is at least equal to the number of stripes. SMS needs at least one buffer for each stripe/volume accessed by the I/O request. For more information, see 4.4.13, "Buffering options" on page 148 for more information. Using the default BUFND eliminates some of the benefits of striping.
- Using LSR for a VSAM striped data set is allowed. However, you might not see a performance improvement in the way that you have with NSR.
- SMS computes the number of stripes based on your SDR parameter divided by a 4-MBps rate. Because your storage volume performs much better than this figure, you might get an actual better rate than you specified in the SDR in the storage class.
- Striping does not give a benefit for directly accessed data sets. However, these data sets are also accessed sequentially during backup, report generating, and so on. Therefore, you might want to consider striping many of your VSAM data sets.
- Data stripping run intensively by batch jobs at online shift might stress the I/O configuration, which can cause overuse and bottlenecks to the online I/Os.
- Since DFSMS 1.12, VSAM data striping is also available for RLS buffering mode.

Guidelines:

- For striped data sets, use SMB to determine the number of buffers or allocate a larger value for the BUFND, depending on your application. Using the default BUFND eliminates some of the benefits of striping.
- All the volumes that contain the stripes should have the same speed. With the RAID controllers, it means that these 3390s should be mapped to disk of the same speed.
- For VSAM, do not go beyond four stripes. If you do, you might overload the processor.
- Define striping only for sequential processing
- Do not exaggerate in the use of data striping by batch along the online shift. It use many I/O resources.

4.5 VSAM performance by scenarios

This section contains an example scenario, where one or more VSAM data sets are causing performance problems to key transactions in a real installation. It covers all the aspects that can affect these performance indicators:

- VSAM I/O waiting time, (Tw(IO))
- VSAM I/O service time, (Ts(IO))
- VSAM CPU service time, (Ts(CPU)).

All of them affect the total response time. This section offers suggestions on how to decrease it. In the example, Resource Measurement Facility (RMF) reports are used.

4.5.1 I/O response time components

Before starting the scenario, you must understand the four time components of the I/O response time. Figure 4-20 shows the DASD Activity report where you can see a practical example of those times.

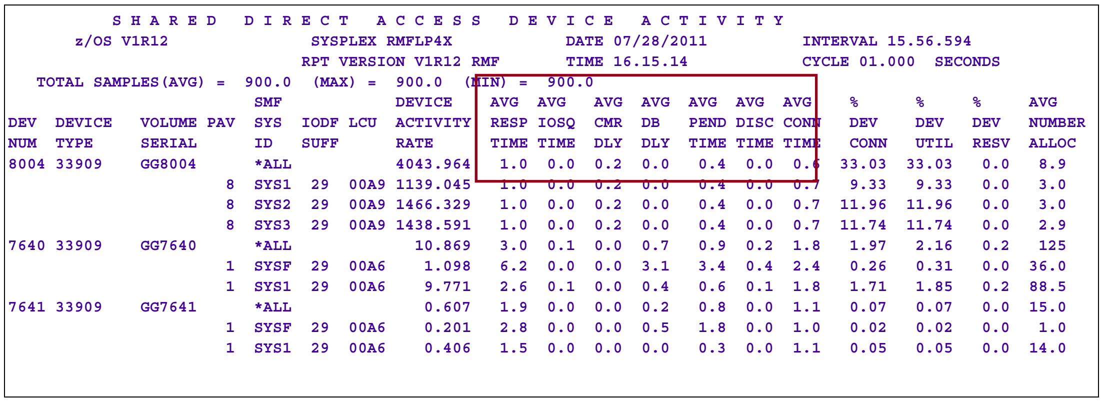
The DASD response time (Tr) is a metric for a device performance mainly for devices that contain major installation data sets, such as DB2 table spaces. The formulas that are related with I/O response time (Tr I/O) are:

```
Tr I/O = IOSQ Time + Pending Time + Connect Time + Disconnect Time
```

Where:

```
Tw I/O = IOSQ Time + Pending Time
Ts I/O = Connect Time + Disconnect Time
```

Some workloads are prone to higher average response times because they move many bytes per I/O operation. This configuration might still be acceptable because that movement implies high connect time. For example, DB2 sequential prefetch, DB2 castout, page/swap I/O, and batch sequential I/Os (as DFDSS file transfer, or DFSORT) are I/O operations that have a high amount of data transfer per I/O operation. Therefore, they have a high connect time. Figure 4-20 shows an RMF Shared DASD Activity Report showing real lifetime components of the I/O response time. This report shows all the activity per 3390 generated by all z/OS systems in the sysplex. The information is captured in SMF record 74.1.



```
           S H A R E D   D I R E C T   A C C E S S   D E V I C E   A C T I V I T Y
        z/OS V1R12              SYSPLEX RMFLP4X          DATE 07/28/2011         INTERVAL 15.56.594
                               RPT VERSION V1R12 RMF     TIME 16.15.14           CYCLE 01.000  SECONDS
    TOTAL SAMPLES(AVG) =  900.0  (MAX) =  900.0  (MIN) =  900.0
                           SMF            DEVICE    AVG  AVG   AVG  AVG   AVG  AVG  AVG    %      %      %     AVG
DEV  DEVICE    VOLUME PAV  SYS  IODF LCU  ACTIVITY  RESP IOSQ  CMR  DB    PEND DISC CONN   DEV    DEV    DEV   NUMBER
NUM  TYPE      SERIAL      ID   SUFF      RATE      TIME TIME  DLY  DLY   TIME TIME TIME   CONN   UTIL   RESV  ALLOC
8004 33909     GG8004      *ALL           4043.964   1.0  0.0  0.2  0.0    0.4  0.0  0.6  33.03  33.03   0.0    8.9
                        8  SYS1  29   00A9 1139.045   1.0  0.0  0.2  0.0    0.4  0.0  0.7   9.33   9.33   0.0    3.0
                        8  SYS2  29   00A9 1466.329   1.0  0.0  0.2  0.0    0.4  0.0  0.7  11.96  11.96   0.0    3.0
                        8  SYS3  29   00A9 1438.591   1.0  0.0  0.2  0.0    0.4  0.0  0.7  11.74  11.74   0.0    2.9
7640 33909     GG7640      *ALL             10.869   3.0  0.1  0.0  0.7    0.9  0.2  1.8   1.97   2.16   0.2    125
                        1  SYSF  29   00A6    1.098   6.2  0.0  0.0  3.1    3.4  0.4  2.4   0.26   0.31   0.0   36.0
                        1  SYS1  29   00A6    9.771   2.6  0.1  0.0  0.4    0.6  0.1  1.8   1.71   1.85   0.2   88.5
7641 33909     GG7641      *ALL              0.607   1.9  0.0  0.0  0.2    0.8  0.0  1.1   0.07   0.07   0.0   15.0
                        1  SYSF  29   00A6    0.201   2.8  0.0  0.0  0.5    1.8  0.0  1.0   0.02   0.02   0.0    1.0
                        1  SYS1  29   00A6    0.406   1.5  0.0  0.0  0.0    0.3  0.0  1.1   0.05   0.05   0.0   14.0
```

Figure 4-20 RMF Shared DASD Activity report

As shown in the formula, I/O response time is formed by four components. For more information about I/O response time components, see 4.5.4, "Decreasing the VSAM I/O response time" on page 196. Following is a brief description of each of these four components as shown in Figure 4-21 on page 187.

IOSQ time

IOS queue time starts when the I/O operation is passed to the Input/Output Supervisor (IOS, a z/OS component) and finishes when the SSCH instruction is run. It is caused by the I/O request being queued in the UCB (a control block that represents the 3390 device) by IOS.

Pending time

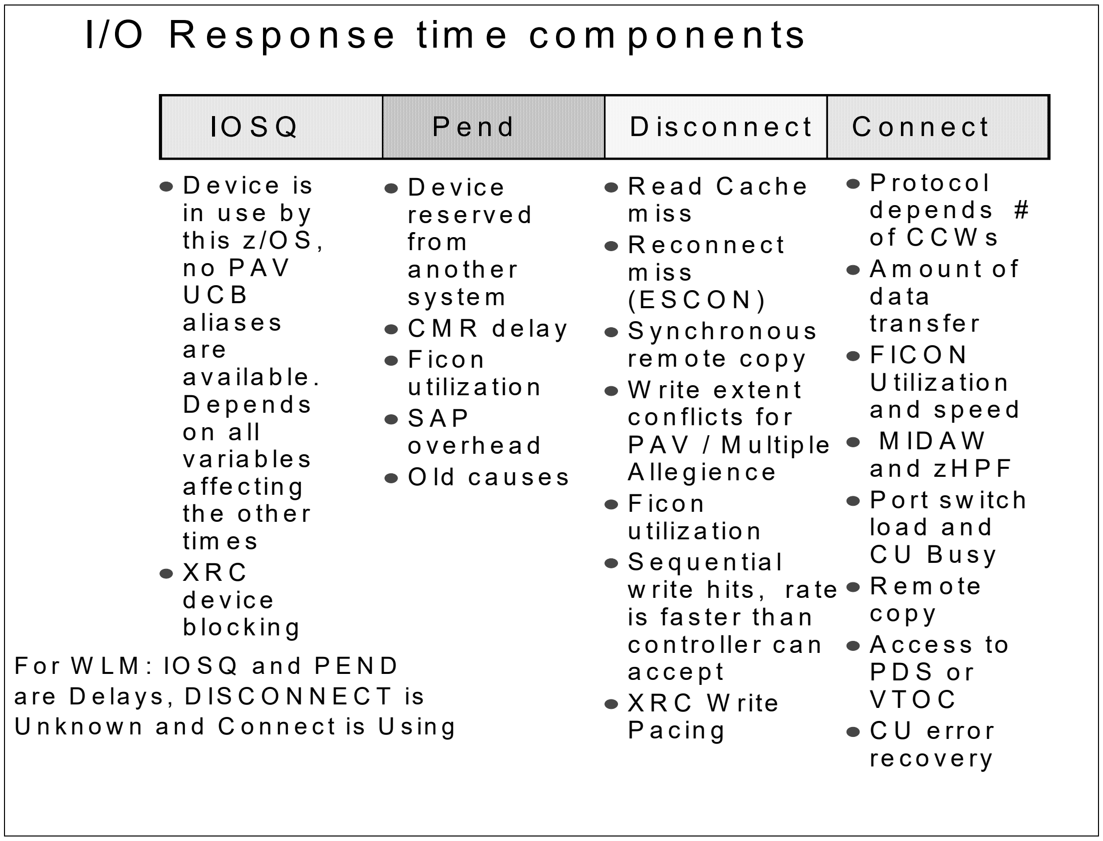
Starts with the execution of the SSCH instruction by IOS. It ends with the beginning of the dialog between the channel and the controller ("command reply" (CMR) frame from the controller). It is a measurement of the queuing time in the initiative queue.

Disconnect time

Disconnect time is when the I/O operation is already started (after CMR frame is received by the channel), but FICON and I/O controller are not in a dialog because of events happening at the controller level. The reasons for the disconnect time are not individually presented in the DASD Activity RMF report. If your disconnect time is above 0.5 millisecond, take care of the reasons for disconnect time (all of them at controller level).

Connect time

Connect time is when the channel is transferring data from or to the controller cache or exchanging protocol control information with controller about an I/O operation (Figure 4-21).



I/O Response time components

IOSQ	Pend	Disconnect	Connect
● Device is in use by this z/OS, no PAV UCB aliases are available. Depends on all variables affecting the other times ● XRC device blocking	● Device reserved from another system ● CMR delay ● Ficon utilization ● SAP overhead ● Old causes	● Read Cache miss ● Reconnect miss (ESCON) ● Synchronous remote copy ● Write extent conflicts for PAV / Multiple Allegience ● Ficon utilization ● Sequential write hits, rate is faster than controller can accept ● XRC Write Pacing	● Protocol depends # of CCWs ● Amount of data transfer ● FICON Utilization and speed ● MIDAW and zHPF ● Port switch load and CU Busy ● Remote copy ● Access to PDS or VTOC ● CU error recovery

For WLM: IOSQ and PEND are Delays, DISCONNECT is Unknown and Connect is Using

Figure 4-21 I/O response time components

4.5.2 Performance scenario that uses RMF reports

Resource Measurement Facility (RMF) is an IBM licensed program that measures z/OS system performance data. RMF has three Monitors, a Postprocessor function, and several programs (RMF Spreadsheet is one of them) to analyze z/OS performance data on a desktop system.

Each Monitor has a Data Gatherer and a Data Reporter component. For Monitor I and Monitor II, both functions are clustered in the same address space. For Monitor III, they are in distinct address spaces.

- Monitor I is an ongoing non-interactive monitor that mainly measures system (z/OS and hardware) variables.
- Monitor II is interactive, and shows snapshots performance data about address spaces and the system itself.

- Monitor III is interactive, and shows the performance of transactions and the system through a technique named Contention Analysis that uses the concept of Using and Delay.

This example uses steps that are based on RMF Monitor III.

RMF contention analysis

The formula that is presented in Figure 4-21 on page 187 is convenient to determine the component that is contributing the most to a long transaction response time. However, measuring all the elements of such equation requires too many resources. The solution is to sample the state of the transactions (in address spaces and enclaves basis) to derive an approximation of those values. Sampling is a much less invasive technique than measuring data in real time. Instead of measuring the real value, RMF monitor III samples the state of the dispatchable units (TCB and SRB) running in address spaces and enclaves. It keeps the result in four state counters:

- Using state is the sampled view of service time (Ts) broken by processor and I/O
- Delay state is the sampled view of wait time (Tw) broken by six major reasons
- Unknown state is the sample view of wait time (Tw) caused by any other reason not included in the six reasons above.
- Idle state is the sample view of the idle time, that is, no transaction (no service) is being run in the address space or enclave.

Workflow% is another Contention Analysis metric that is based in the Using and Delay sampled numbers.

The Workflow% formula is: USING / (USING + DELAY). The higher the value, the fewer delays the dispatchable units (TCB and SRB) in address space and in the enclave are facing.

These measurements provide a detailed view of what happens in the system, and can help you understand and fix system performance. You can see which address spaces, enclaves, and workloads are performing well and which ones are not. Use these measurements to learn the reasons for performance problems that are occurring in workloads or resources.

WLM is a z/OS component in charge of enforcing transactions business goals. It uses Contention Analysis to discover what components are causing the slowdown of transactions. This discovery is done by inspecting the delay figures. WLM derives a metric (used as a type of goal) called Execution Velocity% that is similar to Workflow%.

VSAM performance scenario

This section walks you through an example VSAM performance scenario.

1. The RMF Monitor III SYSSUM report (Figure 4-22) shows that the important business service class period (HTTPS1) is not reaching the goal defined in the WLM policy. This policy is sourced from SLA. To get the SYSSUM report, select the SYSPLEX option from RMF III Primary menu.

```
                ------- Goals versus Actuals --------  Trans --Avg. Resp. Time-
                Exec Vel  --- Response Time ---  Perf  Ended  WAIT EXECUT ACTUAL
 Name     T  I  Goal Act  ---Goal--- --Actual--  Indx  Rate   Time   Time   Time
BATCH    W          100                                0.000 0.000  0.000  0.000
BATCHLOW S  5    25 100                          0.25  0.000 0.000  0.000  0.000
HTTPW1   W          100                                0.000 0.000  0.000  0.000
HTTPS1   S  1        25   0.80 AVG  1.60   AVG   2.0  44.4  0.000  1.600  1.600
OMVS     W           97                                0.030 0.001  0.711  0.712
OMVS     S           98                                0.030 0.001  0.711  0.712
         1  2       0.0  0.500 AVG  0.001  AVG  0.00  0.010 0.000  0.001  0.001
```

Figure 4-22 RMF SYSSUM report

HTTPS1 is a service class that is associated with the transactions of an HTTP scalable server. These transactions run in tasks under enclaves. Its importance (I) is one (the maximum). The goal is an average response time of 0.80 seconds. Its actual response time is 1.6 seconds and the performance index is 2.0, meaning it is 100% out of the target.

2. Look at the Enclave report in Figure 4-23 to see where the problem is.

```
ENCLAVE Attribute  CLS/GRP  P Goal    %   D    EAppl%   TCPU    USG  DLY  IDL

ENC0003 CCT        HTTPS1   1 0.80    25         18.75  26.78    30   88  0.0
        HTML
        COLLOR
ENC0001 CTT        HTTPS1   1 0.80    24         16.27   23.12   29   89  0.0
          HTML
        CARDOSO
```

Figure 4-23 RMF Enclave report

3. In this report, you see two enclaves that belong to the service class HTTPS1 that are suffering 88% and 89% delay. These percentages mean that transactions do not advance for that percentage of time. There are two types of resource delays, as far as WLM is concerned:

   – Delays for resources that are controlled by WLM. A resource that is controlled by WLM is the one where WLM can change the priority of the request in the queue. This adjustment then changes the delay’s value. Therefore, those delays are measured by WLM in order for the delays to be minimized (by playing with the queue priority), depending on the performance index of the service class. The following items can be adjusted:

     - Processor
     - I/O (if you select the WLM I/O Management option in WLM policy)
     - Central Storage (page faults and non-wait swaps)
     - Delay for any Queue Server for certain workloads, such as HTTP, WAS, and DB2 Stored Procedures
     - WLM capping

The 88% and 89% values that are pictured in the RMF report refer to the WLM managed resources.

– Delays for a resource that is not controlled by WLM:

  - ENQ delays (RMF Monitor III)
  - Operator delays for mount delays and WTOR message delays (RMF Monitor III)
  - Subsystem delays, such as JES, HSM, and XCF (RMF Monitor III)
  - DB2 lock delays
  - z/OS lock and latch delays

4. Examine these delays in the report Enclave Classification Data that is shown in Figure 4-24.

```
The following details are available for enclave ENC00003 :
 Press Enter to return to the Report panel.

  Detailed Performance Statistics:

  -- CPU Time --   ------------- Execution States -------------
  Total    26.78   #STS  -Using-  ------ Delay ------  IDL  UNK
  Delta    22.50         CPU I/O  CPU I/O  STO CAP QUE
                   592   6  27    12  77.0 0.0 0.0 0.0  0.0  0.3
```

Figure 4-24 Enclave Classification Data report

5. In enclave ENC00003, 77% of the WLM managed delays are caused by I/O delays. I/O requests from the tasks enclaves are being delayed in the UCB (IOS queue time) or in the channel subsystem (pending time). The Using I/O value is 27%. The Using I/O concept's meaning is ambiguous, and depends on who is creating such figure:

   – In RMF terms (as in the case of the report), the task enclave application is running a channel program (connected plus disconnect) for 27% of the observed time.

   – In WLM terms, the I/O disconnect time is not included in the WLM Using state. WLM uses this Delay% figure to derive the Execution Velocity type of goal values.

6. Examine the Monitor III Device Delay report (DEV) in Figure 4-25.

```
           Service  DLY USG CON  ------------ Main Delay Volume(s) -
Jobname  C Class    %   %   %    %  VOLSER   %  VOLSER   %  VOLSER

HTTPS000 S SYSSTC    77  27  23   70 VSMS15  11 VSMS19
MICHAELL B NRPRIME   39  15  14   39 BPXLK1
MCPDUMP  S SYSSTC    36  18  20   36 D24PK2
CHARLESR B NRPRIME   33  13  13   28 BPXLK1    3 HSML02    2 BPXSSK
DFHSM    S SYSSTC    30  83  35   10 HSML17    5 SMS026    4 HSMOCD
SHUMA3   T TSOPRIME  18  52  53   13 D83ID0    5 HSML02
```

Figure 4-25 Monitor III Device Delay report

To this date, there is no RMF report that shows details (at a volume level) about the I/O use and I/O delay of an enclave. You must know the name of the address space where the HTTP transactions are doing I/O. It must be one of the HTTP server address spaces. In this example, it is the HTTPS000. You can see that the volume VSMS15 is responsible for 70% of the delays that are experienced by the enclave HTTP transactions.

7. Examine the Monitor III DEVN report in Figure 4-26 to see more information about this volume.

```
Device Identification --    -- Activity --     ACT CON DSC - Pending - - Jobs -
VolSer Num  Type    CU      S  Rate RspT IosQ    %   %   %   %  Rsn. %  USG  DEL

VSMS15 006C  33903  3990-3   S  42.1 .022 .006  68   8  60   2 DB    1  0.0  0.8
VSMS10 0051  33903  3990-3   S  80.7 .011 .005  47  24   1  22 DB   11  0.2  0.7
JOBL17 0703  33903  3990-3   S  52.2 .015 .000  76  22  54   0          0.2  0.6
TSO015 006E  33903  3990-3   S  11.1 .024 .001  26   3  20   3          0.0  0.3
VSMS06 0056  33903  3990-3   S   8.9 .034 .001  30   9  18   3 DB    2  0.1  0.2
```

Figure 4-26 Monitor III DEVN report

The DEVN report for volume VSMS15 shows the I/O response time (.022 seconds), the I/O rate (42.1 I/Os per second), the IOSQ time (0.006 second), and the percent distribution of connect (8%), disconnect (60%), and pending (2%). If you want to know the connect per I/O operation (AVG CONN TIME), use this formula:

```
If the RMF range is 100 seconds, the total connect time was 8 seconds. In 100
seconds, it was running (42.1 * 100) I/O instructions. Divide 8 by 4210, to
find you have 2 milliseconds of AVG CONN TIME.
```

To get the DEVN report, use the U.DA option from the RMFIII Primary menu.

8. Find the data set that is involved in the performance problem in the Monitor III DSNV report that is shown in Figure 4-27.

```
 RMF V1R12  Data Set Delays - Volume         Line 1 of 2
 Command ===>                                                 Scroll ===> CSR

 Samples: 60      System: SC70  Date: 06/01/11  Time: 19.24.00  Range: 60    Sec

 -------------------------- Volume VSMS15 Device Data --------------------------
 Number:    D83A       Active:      23%       Pending:    9%     Average Users
 Device:    33909      Connect:     12%       Delay DB:   0%        Delayed
 Shared:    Yes        Disconnect:   2%       Delay CM:   0%          0.0
 PAV:       1.0H
 -------------- Data Set Name ---------------   Jobname   ASID  DUSG% DDLY%
PROD.KSDS.MARCH.U12.DATA                         SMSVSAM   0010     20     2
PROD.KSDS.MARCH.U12.INDEX                        SMSVSAM   0010      2     0
```

Figure 4-27 Monitor III DSNV report

You now have all the information that you need: The volume and the data set name (which is VSAM data set PROD.KSDS.MARCH.U12). Determine the largest figure among IOSQ, pending, connected, and disconnected in your installation (in this example it is disconnect time). IOSQ is better to pick it up in average milliseconds per I/O operation from the DEVN report. Depending on the one you pick, go to the corresponding topic in this chapter, where you find recommendations to improve it. Remember that all of these suggestions refer to one of the three ways of solving performance problems, that is *buy*, *tune*, or *steal*.

However, before you try to find a way to improve your I/O performance, see 4.5.3, “Reducing the number of I/Os” on page 192. This section offers suggestions to eliminate performance problems that are caused by excessive I/O operations.

4.5.3 Reducing the number of I/Os

The general rule that the best I/O is the one that is not run still applies to VSAM. Therefore, before you try to improve the I/O operation, trying to avoid as many I/Os as possible. You can reduce the number of I/Os by using these techniques:

- Buffering
- Saving I/Os associated with CA splits
- Secondary space allocations
- Write checks
- RECOVERY option

Buffering

Buffering is one of the most important VSAM features that can be used to avoid I/Os, and so improve performance. There are two types of buffering: VSAM controlled buffer pools and Hiperbatch.

VSAM controlled buffer pools

VSAM controlled buffer pools are managed in LSR, NSR (with or without SMB), RLS, and GSR buffering modes. Some of them are allocated in the application address space and some in the hiperspace. For more information, see 4.4.13, “Buffering options” on page 148.

The value of buffering from the perspective of I/O performance is shown in the following scenario*:*

- KSDS cluster that requires 0.5 GB in a 3390 volumes.
- Data CI size is 4 KB.
- FREESPACE (15 9), 15% of the data CI is free and 9% of the data CA is free.
- The control area size is not restricted, so it is 15 tracks. There are 180 data CIs in each control area.
- There are 729 Control Areas, and therefore 729 index sequence set CIs.
- The index control interval size is 1536 bytes.
- There are five index set records.
- There are three index levels.

Supposing no hits in the data and in the index buffers, the system has these characteristics:

- The application program generates 45 random reads per second, with an M/M/1queue behavior. M/M/1 stands for normal distribution for the inter-arrival time (M) and for the service time (M) plus a single server
- 100% of DASD caching hits for index that cause an average service time of 0.5 millisecond
- 50% of DASD caching hits for data, each causing an average service time of 6 milliseconds.

Perform the calculation by using these formulas:

- I/O Service_Time per average Read caused by three index I/Os and one data I/O: (3 x 0.5 + 6) = 7.5 ms
- Device Utilization (U)% = IO_Rate x Service_Time x 100: (45 x 0.75) = 33.75%
- I/O Queue_Time for M/M/1 = I/O_Service_Time x (U / (1-U))= 3.8 ms
- I/O_Response_Time: (7.5 + 3.8) = 11.13 ms

Supposing all index set in the buffer (66% buffer hit) and 50% buffer hits for data, the system has these characteristics:

- I/O Service_Time per average Read caused by one index sequence set I/O and one data I/O: (1 x 0.5 + 3) = 3.50 ms
- Device Utilization (U)% = IO_Rate x Service_Time x 100: (45 x 0.3) = 15.75%
- I/O Queue_Time for M/M/1= I/O_Service_Time x (U / (1-U)) = 0.65 ms
- I/O_Response_Time: (3.00 + 0.65) = 3.65 ms
- I/O_Response_Time gain proportion: (11.13 / 3.65) = 3.04 times faster.

To improve 304%, your I/O needs more buffers, and so a trade-off between I/O and memory. Calculate the number of extra buffers by using these formulas:

- For five index set buffers: 5 x 1536 = 7680 KB
- For data buffers calculation, use a derivation of the 80/20 rule. To have 80% hits, you need 20% of buffers. To have 50% hits (as in the scenario), you need 10 percent. Have 180 x 0.91 = 163 CIs not free per CA.
- 10% of the total number of CIs is: 729 x 163 x 0.1 x 4 _KB = 47.5 MB

In an address space of 2 GB and with central storage much larger than 2-GB, the storage price is minimal for the I/O performance gain.

Hiperbatch

Hiperbatch is designed to eliminate the performance problems that are caused by these factors:

- Multiple jobs in one z/OS image that request data from the same QSAM/VSAM NSR data set simultaneously. Each job causes I/O operations to the device that holds the data set. The more jobs that access the data set concurrently, the more I/O operations and contention for the device, and the longer the wait time for each I/O request.
- Jobs or job steps that pass temporary or short-lived QSAM or VSAM NSR data sets to subsequent jobs. As a job completes, it puts the final produced data set used back onto DASD.

One important aspect of Hiperbatch is that installations can take advantage of these performance benefits without having to change existing application programs or the JCL required to run them.

Hiperbatch depends on the data lookaside facility (DLF) address space to control access to an Hiperspace Expanded Storage Only (HS ESO).

RACF DLFCLASS profiles provide the data set name list that DLF needs. The existence of a DLFCLASS profile for a VSAM data set identifies that data set as one that is eligible to be processed as a DLF object.

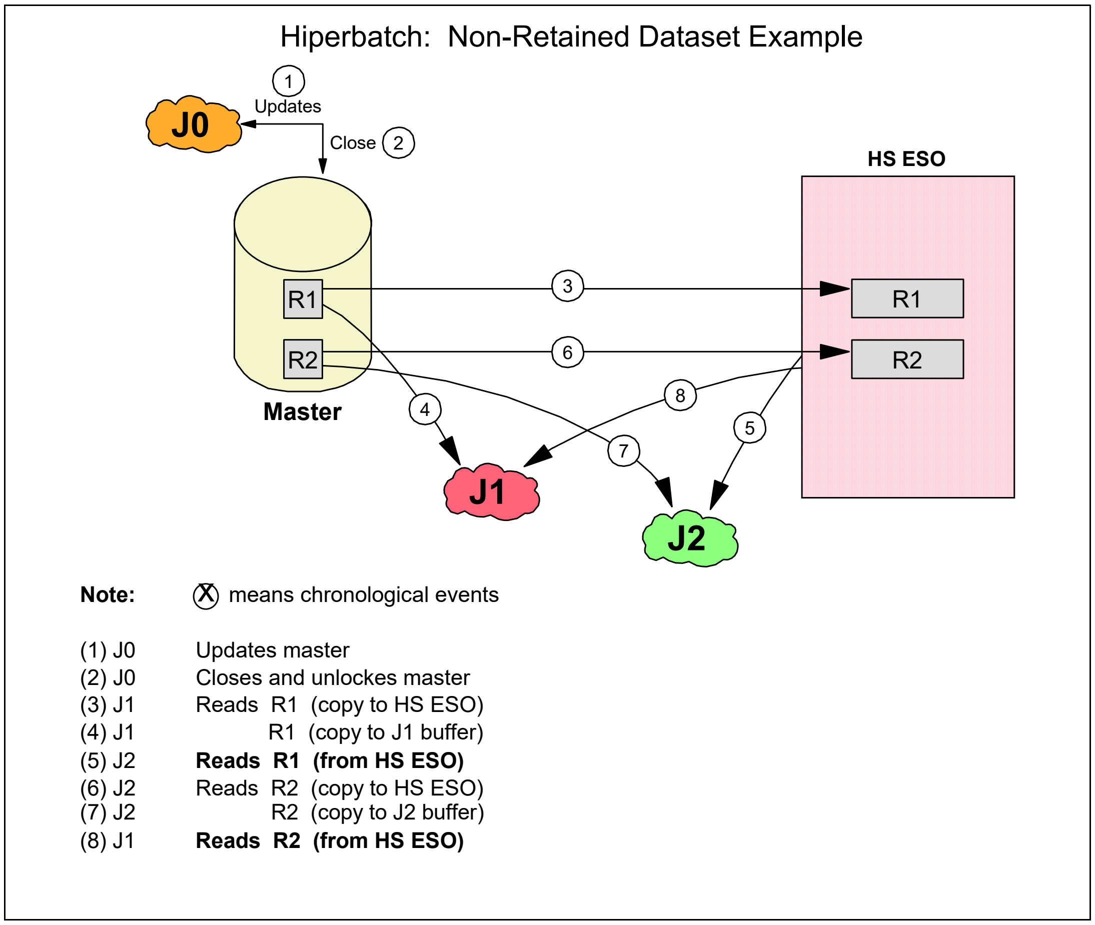
When DLF is active, the first attempt to access a QSAM or VSAM data set defined to DLF causes it to create a DLF object. This object can be a data set in the HS ESO. A DLF object contains data from a single data set that is managed by Hiperbatch. The user (an application program) is connected to the DLF object. The connected user can then access the data in the object through normal QSAM or VSAM macro instructions.

When subsequent users access the data set, they are also connected to the object. The system manages shared access to the DLF object in the same way it would manage shared access to the data set. When a user relinquishes access to the data set, DLF disconnects that user from the object.

A DLF object exists until there are no users of the data set, at which time DLF deletes the object. While there is at least one user of a data set, the access pattern means that the DLF object exists.

However, if a batch job or job step creates a data set and passes it as input to another job or job step, there is not always one user of the data set. In this case, the system deletes the DLF object. To prevent this automatic deletion, define the data set to DLF as a *retained DLF object*. A retained DLF object is one that the system does not automatically delete when there are no users of the data set.

Figure 4-28 shows an example of a non-retained data set (called Master) that is a DLF object. Start all the reading jobs in parallel as usual without Hiperbatch. The first job (J1) reads sequentially the first record (R1) and suffers the I/O delay. After I/O completion, one copy of R1 is delivered to J1, and another copy is kept in the HS ESO by Hiperbatch. When J2 requests an I/O for R1 reading, it is intercepted and fulfilled with the R1 copy in the HS ESO. Then, for N Jobs reading M records each, from the Master, you have only M accesses to DASD, instead of M * N.



Note:	Ⓧ means chronological events	
(1) J0	Updates master	
(2) J0	Closes and unlockes master	
(3) J1	Reads R1 (copy to HS ESO)	
(4) J1	R1 (copy to J1 buffer)	
(5) J2	**Reads R1 (from HS ESO)**	
(6) J2	Reads R2 (copy to HS ESO)	
(7) J2	R2 (copy to J2 buffer)	
(8) J1	**Reads R2 (from HS ESO)**	

Figure 4-28 Hiperbatch example

Using Hiperbatch with VSAM has these considerations:

- VSAM non-shared resource (NSR) access is a requirement.
- Hiperbatch does not support extended format data sets because they are not processed by media manager.

- VSAM organizations KSDS, ESDS, and RRDS with a control interval of 4096 bytes, or a multiple of 4096 bytes, are eligible.
- The EXCP counts in SMF records used to record I/O use do not change. The counts reflect I/O operations requested regardless of whether a request is satisfied by a physical I/O operation or from a DLF object in expanded storage. However, the total I/O connect time of these jobs should decrease considerably.
- If the data set is a VSAM key-sequenced data set (KSDS), the DLF object contains only the data component, and not the index component.
- To avoid data integrity exposures, Hiperbatch does not process a VSAM data set with share options 3 or 4. This restriction is true even if that data set is defined as eligible for Hiperbatch.
- VSAM data sets with the number of strings (ACBSTRNO) value greater than 1 cannot be Hiperbatch objects.
- VSAM data sets must be opened by the base cluster name.
- If a random (or sequential) program changes data in the data set, that change is also made to the DLF object.
- This Hiperbatch method is best suited for sequential processing.

IBM provides an online monitor that you can use to track Hiperbatch activity on your system. Using the monitor, you can evaluate these data:

- How Hiperbatch is using expanded storage (or central storage in z/Architecture)
- The use patterns for individual data sets
- The I/O operation savings for individual jobs

For more information about how to install the Hiperbatch monitor, see *MVS Hiperbatch Guide*, GC28-1470. After it is installed, you can start the monitor under TSO/E by issuing the COFDMON command.

Saving I/Os associated with CA splits

These I/Os should be avoided. CA splits generate many I/O operations. CA splits occur in a KSDS or VRRDS along inclusions or by increasing the logical record size during an update. CA splits can be minimized by the use of CA free space. For more information, see 1.4.10, “Splits” on page 17.

However, a CI split is not a real performance issue because it needs around four I/Os. Also, remember that if a KSDS had lots of splits, then reorganizing it might increase the number of future splits.

Secondary space allocations

Every secondary allocation implies going through End-of-Volume processing with several I/Os in the catalog and VTOC. For more information, see 4.4.1, “Allocation units” on page 134. You should require a consistent amount of secondary allocation.

Write checks

Write checks needlessly increases the number of I/O operations. Avoid this option.

RECOVERY option

Use the SPEED option instead when you are loading a VSAM data set. For more information, see 4.4.11, “Initial load option” on page 145.

4.5.4 Decreasing the VSAM I/O response time

After trying to avoid I/Os, this section addresses how to make the non-avoidable I/Os faster.

I/O wait time for VSAM data sets

As you know, the VSAM I/O response time is made of waiting time plus service time:

```
Tr (I/O) = Tw (I/O) + Ts (I/O)
```

This section repeats some of the concepts that are described in 4.5.1, “I/O response time components” on page 186.

Tw (I/O) has two components: IOS Queue Time and Pending Time.

IOS queue time for VSAM data sets

IOS queue time starts when the I/O operation is passed to the IOS, a z/OS component. It finishes when the SSCH instruction is run. It is caused by I/O requests being queued in the UCB (a control block that represents the 3390 device) by IOS. When PAV is active, all the PAV device base UCB and all aliases UCBs are busy within this z/OS with previous I/O requests. PAV helps to decrease IOS queue time by creating the possibility of parallelism.

Recall that queue time is a consequence (not a cause) of high service time and high arrival rate (demand).

The DS8000 XRC device blocking function might also cause IOSQ Time. When in contention (the write arrival rate is larger than the take out rate done by the System Data Mover), the primary DS8000 tells IOS to inject an artificial “XRC device blocking” delay. This value is between zero and one second for ALL (reads and writes) I/Os to the device.

IOSQ Time is then caused by two major factors:

- High used 3390 device. This utilization can be caused by activity (high rate and high service time) coming from this MVS image and possibly from another image (in a shared DASD case).
- Lack of UCB aliases in your PAV implementation.

To decrease the IOSQ Time, you can use these techniques:

- Buy a faster device/controller/channel to decrease the I/O Service Time and consequently the utilization (for the same I/O load). Reducing the utilization then decreases the queue time. Remember that Tw (I/O) decreases when Ts(I/O) decreases.
- Decrease the Ts(I/O) by tuning it. For more information, see “I/O connect time for VSAM data sets” on page 201.
- Change the service class goal that contains the transactions that are generating the I/Os that are causing the IOSQ time delay. Increase the importance of the goal or change its numerical value to make it more difficult to be obtained. The I/O priority is then raised by WLM. This process occurs when the transaction is not reaching its goal and the major delay is the I/O delay. In this case, you are not improving the I/O in general, but just improving the response time of favored business transactions.
- Decrease the I/O rate against the device by avoiding placement of several active data sets on the same volume, mainly index and data from the same KSDS. If this happens, verify your ACS routines, perhaps by using the SMS Guaranteed Space option to force the index in a specific volume. Use different storage groups or the function DFSMS Data Set Separation, where you can separate data sets from each other in different 3390 volumes and even in different physical DASD controllers.

- Increasing the parallelism is a way to decrease queue time:
  - You can reshape PAV to increase the number of available alias UCBs in this device. With PAV, you can have parallel I/Os against the same logical 3390 device, decreasing the IOSQ time. However, two writes in the same extent are not allowed. Also, PAV has a price. To have *N* parallel access, you need *N* alias UCBs (and UCWs). Generally, use HiperPav instead of other PAV implementations.
  - Use the great level of parallelism that is implemented through FICON channels.

Pending time for VSAM data sets

Pending time starts with the execution of the SSCH instruction by IOS, and ends when the controller replies with a Command Reply (CMR) type of FICON frame. It means that the I/O dialog between the channel and the controller is starting, indicating the end of the I/O queue time. It is a measurement of the queuing time in the SAP initiative queue. This queue is managed by SAP (a PU specialized in handling the start of an I/O operation). New technologies such as FICON, PAV, Multiple Allegiance, and modern controllers have eliminated some of the reasons for pending time and moved others to disconnect time:

- The all channel path busy delay (a former time component of pending time) situation for FICON is almost zero because FICON channels are always available. FICON channels only respond busy to SAP when the 64 concurrent I/Os (open exchange) limit is exceeded. FICON channel utilization can be seen in RMF Channel Path Activity report. FICON is not a real figure, but rather is derived by comparing current average data rate with the maximum data rate (aggregate data rate).
- FICON switches such as IBM ESCON® switches do not present director port switch (DPS) busy (a former time component of pending time).
- Modern controllers hide the CU busy situation (a former time component of pending time), accepting the I/O request even when the host adapter is busy. The former control unit busy time is moved initially to disconnect time and then to connect time. For more information, see YYYY. A new pending time component is measured that is the time that the controller takes to respond to the channel when the I/O operation is starting. It is called CMR delay.
- The DS8000 multiple allegiance function allows multiple concurrent I/Os to the same shared device (no device busy time, a former time component of the pending time). There are two exceptions:
  - CCW reserved device by one of the systems. This delay is still shown as device busy time within pending time.
  - Write extent conflicts for PAV. In this case, the controller queues the second I/O operation, then includes it at disconnect time

Some RMF reports, such as Shared DASD Activity report with a device granularity (Figure 4-20 on page 186) are modified to report on new hardware and software features. Depending on how current your installation is, some of the older report fields are still valid. You can look at the I/O Queuing Activity report (in an LCU granularity) that still shows information, such as ESCON DPS busy and old CU busy.

Pending time is formed by the following delays:

- SAP delay: Consists of SAP processing time plus SAP busy queuing time (usually around 0.1 msecs for a non-overloaded z196 SAP) per I/O request. Presented in RMF I/O Queuing report (LCU basis) column AVG CSS DLY. SAP utilization should not go beyond 40% to avoid such delays. RMF I/O Queuing report also presents detailed SAP performance data that provides their average utilization.
- CMR (command reply) delay: How long FICON channel "waits" for the controller (host adapter) at the start of the I/O dialog. It is shown in RMF Device Activity report and RMF

I/O Queuing report. It is an indirect indication about how busy the controller is. It measures the controller load and in a sense is a replacement for the former CU busy delay figure. For a FICON protocol, after receiving CMR, all conversation along this I/O becomes asynchronous. This figure is presented in the DASD Activity RMF report.

- Device busy delay: Measures how long the device is busy because of a Reserve CCW issued by another z/OS. It is an explanation of these factors:
  - Missing Interrupt Handling messages
  - Bad DASD performance because the full 3390 is serialized to other MVS systems
  - High probability of dead locks.

  Nowadays, it is rare to see devices reserved because of the use of GRS and RNL conversion list. The reserve is used only for a device that is shared by z/OS systems that run in different sysplexes. This information is presented in the DASD Activity RMF report.

To decrease the IOSQ Time, you can use these techniques:

- Convert Reserve CCW to Global ENQ through the GRS RNL Conversion list.
- Decrease the number of busy channels that are connected to the same host adapter in the DS8000. This configuration might cause a high CMR delay. The host adapter is the DS8000 processor that interfaces with the channels though an optical fiber.
- Verify the average utilization of the SAP PUs in the RMF IOQUEUE report. If it is more than 40%, you might need to acquire more SAPs.

I/O disconnect time for VSAM data sets

Disconnect time is the time that the I/O operation already started (after CMR frame is received by the channel). However, FICON and I/O controller are not in a dialog because of events that are happening at the controller level. In a FICON protocol, there is not a real disconnect concept because of the asynchronous way of multiplexing data. There is not a disconnect sequence started from controller. However, a numerically equivalent time figure is accounted through a timing channel hardware named Channel Measurement Facility. This hardware shows the amount of time FICON channel "waits" (but is running other I/Os) for the controller along this I/O operation. The I/O operations delayed along the disconnect time are ordered internally by the controller, which follows the WLM defined I/O priority (it is not FIFO). The reasons for the disconnect time are not individually presented in DASD Activity RMF report. If your disconnect time is above 0.5 millisecond, investigate the reasons for the disconnect time (all of them at controller level). Observe that the RMF DASD Activity report does not portray the time components of the disconnect time:

- Read cache miss: I/O physical block that is requested by a read is not in controller cache and must be retrieved from the physical disk. You can confirm this time component through RMF Cache reports at the “read hit ratio” figure. Generally, the value must be higher than 80%. For more information about DASD cache, see “DASD cache highlights” on page 199.
- Synchronous remote copy (PPRC): Mirroring for writes is done within the disconnect time. See the latency of such copy at RMF DS8000 Link report. Generally, use Fibre Channel links and a distance of less than 20 KM between controllers (latency of 50 microsecond per KM) for heavily write data rates. Remember that only writes can cause PPRC disconnect time.
- Multiple Allegiance or PAV write extent conflicts time delay: These were previously reported as device busy time (pending time). Modern controllers include them in the disconnect time. Write extent conflict has to do with the avoidance of concurrent writes or write/read in the same data set in the same 3390. The serialization is done by the

controller and the time in the queue is accounted as disconnect. As expected these requests are ordered by the WLM defined I/O priority.

- Sequential write rate is faster than controller can accept in cache. The rate of data that is coming into cache is greater than the rate of destaging such data to disks. To decrease this rate, you can use these techniques:
  - Buy more NVS cache
  - Decrease the dispatching priority of the job flooding the controller with writes
  - Decrease the number of buffers
  - In the case of DB2, decrease the buffer pool thresholds for DB2 cast outs
- Control unit busy: This time was previously reported in pending time, but modern controllers include it in disconnect time. The controller does not respond CU busy to the channel, but queues the IO request internally waiting for the host adapter availability. In even more modern controllers, the host adapter accepts the second channel moving data of both channels in multiplexor mode, increasing the connect time of each.
- As the FICON channel processor becomes busy (above 60%), it takes more time for the channel to realize that the disconnect time has already ended. As a consequence, the disconnect time increases. It is uncommon to see FICON channels with such utilization.
- XRC Write Pacing: This is a new DS8000 function that provides a lower affect alternative to XRC device blocking. For more information, see “IOSQ time” on page 186. When XRC contention is detected, the controller injects artificial disconnect delays per set of written records (not per SSCH). Read I/O is not affected.

In general, you can take the following actions to decrease the disconnect time:

- Reducing I/O rate (demand) against the controller. You can do so by using these techniques:
  - Either compression or use of smaller CIs for random processing. If you are accessing data randomly, use a smaller CI for read data or less free space in the CIs. These configurations avoid polluting the cache with other non-referenced logical records or free space

    For more information about compression, see 4.4.16, “Data compression” on page 170. For more information about use of smaller CIs, see 4.4.5, “Control interval size” on page 139.
  - Reorganize of the data set, if there is plenty of free space in the CIs. For more information, see “I/O connect time for VSAM data sets” on page 201. However, this reorganization can cause splits.
  - Reducing the number of active data sets in the controller.
- In RAID 5 arrays, decrease the number of writes by moving data sets with high update rate to RAID 10 arrays. This configuration avoids the effect of write penalty in RAID-5.

DASD cache highlights

DASD cache is a fast memory that is on the DASD controller. It plays an important role in decreasing the total I/O response time. It minimizes I/O disconnect time through hits (accessing data in cache instead of a disk). In a sense, DASD cache uses the concept of buffer pools in memory.

In this explanation, the term ”disk” does not have the same meaning of DASD. Disk implies the RAID media, where data is physically stored in fixed-block architecture (FBA) blocks through an SSA or a SCSI protocol. DASD is the logical 3390 device as perceived by you, your application, and your MVS operating system.

A DASD cache has two functions:

- Minimize access to disks (by having cache hits) and so decreasing the average disconnect time.
- Serve as a *speed matching buffer* to synchronize hardware elements with different speeds (like FICON channels and disks) along a cache miss during sequential access.

Modern DASD I/O definition is a data movement between DASD cache and z196 L4 cache. The relationship between Cache and DASD capacity is currently about 0.2%. DS8700 has a maximum of 384 GB volatile and 12 GB of NVS caches.

In the past, the performance staff had a key role in customizing DASD cache to produce a high hit ratio. However, the following facts change the picture:

- Larger caches
- It is no longer possible to define a data set as these settings:
  - Never cache
  - Maybe cache
  - Always cache
- Almost all writes are cache hits

To have random cache hits (saving disk access) for reads and writes, the I/O workload must frequently access the same data CI or index CI in VSAM terms. Typically there are two types of hits, when the application revisits data:

- The same logical record in a CI that is already in cache.
- A different logical record in the same CI already in cache because another logical record was previously accessed.

For sequential access, cache does not save data CI disk I/O operations. The cache only tries to match the speed of the disks and channels. Consequently, the faster resource is less used. The controller has a sequential algorithm for reads that implements a look ahead mechanism.

Currently the only metric indicating the effectiveness of the DASD cache is the following formula:

```
READ_CACHE_HIT_RATIO = #_READ_HITS / #_TOTAL_CACHEABLE _READS
```

The general rule for this metric is at least 80%.

Figure 4-29 shows an RMF Cache report, where you can report the metric. There are no cache reports per data set in RMF. You can use the volume report that contains your VSAM data set to reach your conclusions. The 3390 volume VSM015 has a READ_CACHE_HIT_RATIO of 98.9%, which is almost perfect.

```
The following details are available for Volume VSM015 on SSID 0043
Press Enter to return to the Report panel.

Cache: Active       DFW: Active       Pinned: None

       ------ Read ------   --------- Write ---------   Read   Tracks
        Rate    Hit  Hit%    Rate   Fast    Hit  Hit%     %
Norm     3.7    3.6  79.8     0.8    0.8    0.8   100   82.2      0.1
Seq      0.0    0.0   100     0.1    0.1    0.1  93.3   21.1      0.0
CFW      0.0    0.0   0.0     0.0    0.0    0.0   0.0    0.0
Total    3.7    3.7  98.9     0.9    0.9    0.9  99.1   80.0

------ Misc ------  - Non-Cache -  --- CKD ----  - Record Caching -
DFW Bypass :   0.0  ICL   :   0.0  Write:   0.0  Read Miss :   0.0
```

Figure 4-29 Volume Cache report

If this metric is less than 80%, there are two possible explanations:

- The cache is overloaded, at least for this volume
- Your data set is cache-unfriendly.

To improve this metric, you can buy more cache or reduce the I/O demand. If the I/O workload that is accessing the cache is unfriendly because large buffer pools in storage, there is not much to be done. This is the case in DB2. Verify DS8000 rank array data in Cache Activity report, which displays real disk performance. Improve performance of DB2 (or any key cache unfriendly software) by spreading heavy loaded DB2 3390s into different arrays, preferably the fastest ones. The best devices are the least capacity disks (146 GB), spinning at 15,000 RPM, RAID 10 and even solid-state devices.

The writes do not participate in the metrics because they are almost 100% hits. When writing a physical record not stored previously in the cache, the controller provides an available cache element in the volatile cache and another in the non-volatile (NVS). It then runs the two copies. There is no need to go to disk along the write operation (it is a type of store-in algorithm).

I/O connect time for VSAM data sets

The I/O connect time is when channel is transferring data from or to the controller cache, or exchanging protocol control information with controller about an I/O operation. It includes these components:

- Real productive time of an I/O operation, meaning the data transfer itself. In the FICON architecture, data and protocol are transferred through 2-KB frames. The connect time is influenced by these factors:
  - The amount of data (number of frames) being transferred
  - Distance
  - Speed
  - The load of host adapter in the controller
  - Channel speed/utilization (that affects the number of available idle frames)
  - Port switch load
  - Remote copy activities

Some workloads are prone to high connect time per I/O because of a large amount of data that is transferred in just one I/O operation. This is an advantage.

- Overhead of the dialog protocol. This time is influenced by these factors:
  - Protocol type, such as FICON Express 8
  - Distance, speed, and utilization of the channel
  - Speed and the load of host adapter in the controller
  - Number CCWs in the channel program
  - Port switch load

  In sequential processing, if you decrease the number of SSCHs (by defining more buffers) even though transferring in total the same amount of data, your total connect time decreases consistently. This decrease occurs because there are fewer I/O operations and therefore less resource usage. With more buffers in sequential processing, you transfer more bytes per I/O (increasing the connect time per I/O) in fewer I/O operations (decreasing the total connect time). MIDAW and zHPF are channel functions that decrease this dialog protocol time.
- Number of concurrent I/Os that a controller subsystem is processing, hence the percentage of its bandwidth in use affects each I/O data transfer rate.
- PDS directory search and VTOC search also produce non-productive connect time because of Search Key High or Equal CCW.
- Special activities in controller, as such recoveries or micro code updates can cause high connect time.

To analyze your VSAM I/O connect time, you need to know the type of access your VSAM data set is facing. The possibilities are sequential and direct:

- Sequential access. If you did not change your channel and controller, and observe a relatively high average connect time for your sequential access VSAM data set, things are working well. A lot of data is being transferred in just one I/O operation, through one CCW (one buffer) with a large blocksize or many CCWs (many buffers). Increase the average connect time by using plenty of buffer space in the buffer pool for sequential processing.

  This process decreases the number of SSCHs instructions (I/O operations). For every I/O operation that starts, there is a standard conversation between the channel and the controller. If you decrease the number of SSCH instructions, but are transferring the same amount of data, you save in total connect time. In other words your average connect time (per I/O operation) increases, but your total connect time decreases. If the cluster is going to be accessed sequentially and randomly, make the CI size smaller (4 KB). To solve the performance problem in sequential access, you can define many data buffers, causing VSAM to chain CIs in just one channel program. For more information, see 4.4.5, “Control interval size” on page 139.
- Direct access. In direct access, the smaller the average connect time, the better. Have small CIs to avoid bringing unneeded logical records into memory.

Use the following guidelines to decrease the average connect time for both direct and sequential accesses:

- Use FICON Express 8 channels mainly for sequential access. For best performance, your controllers must have a host adapter able to transfer data at 800 MBps.
- Use data compression (and compaction) in CPU: There is a difference between compaction and compression. Compaction extracts blanks and zeros from a text. Compression is a much more elaborate task, where dictionaries can be used to obtain the best result (like the Ziv-Lempel algorithm). These dictionaries contain the most repeated set of characters that are found in the text and their respective compressed substitution. For more information, see 4.4.16, “Data compression” on page 170.

- Use the IBM ECKD™ extended format: ECKD extended format is a technique that affects the way count key data is stored in a 3390 track. It improves performance for certain types of I/O operation, by decreasing Ts(I/O) with a better channel program. Extended format is the basis for using VSAM features such as SMB, data striping, compression. Generally, convert your data sets to extended format to get this better performance.

  The large performance appeal of extended format is that it allows the use of strong performance capabilities such as striping, compressing, and SMB.
- Less free space in the CIs: The existence of a significant amount of free space in a data CI might increase the I/O connect time. Remember that any CI with a logical record is moved to storage in a sequential read and the free space is included in such movement. This is not true with totally free CIs, where no I/O operations are run towards them.

  The solution here might be a reorganization. For more information, see 2.1, “Reorganization considerations” on page 46, watching for an increase in the number of splits.
- Parallel VSAM I/O operations: If you cannot decrease the service time (Ts), you can decrease the Tw, and so increase the ETR (number of I/Os per second) by introducing parallelism. There are two types of I/O parallel processing:
  - Within a transaction: The VSAM option that allows I/O parallelism within a transaction or task (when accessing a VSAM cluster) is data striping. For more information, see 4.4.17, “VSAM data striping” on page 177.
  - Between transactions: The following VSAM options allow parallelism between transactions or tasks when accessing a VSAM cluster:
    - STRNO: In multiple string processing, there can be multiple independent request parameter lists (RPLs) within an address space for the same data set. The data set can have multiple tasks that share a common control block structure. Several ACB and RPL arrangements indicate that multiple string processing will occur.
    - In the first Access Control Block (ACB) opened, STRNO or BSTRNO is greater than 1.
    - Multiple ACBs are opened for the same data set within the same address space, and are connected to the same control block structure.
    - Multiple concurrent RPLs are active against the same ACB by using asynchronous requests.
    - Multiple RPLs are active against the same ACB by using synchronous processing with each requiring positioning to be held.

  For more information about multiple string processing, see *z/OS DFSMS Using Data Sets,* SC26-7410.

In DS8000, the PAV and Multiple Allegiance features implement I/O parallelism in the same volume, within or among transactions.

4.5.5 Decreasing VSAM CPU time

This section addresses decreasing the Tr(I/O) for VSAM I/O operations to decrease the response time of key transactions that use VSAM data sets. However, chances are that the enhancements introduced by your changes will shift the bottleneck to the processor side.

As an example, you might find that your changes have these effects:

- If you compress your VSAM data set, your transaction response time decreases, which is good.
- If you stripe your VSAM data set, your transaction response time decreases, which is also good.
- If you compress and stripe your VSAM data set, your response time gets bigger (which is bad). The reason is that the processor is extremely busy, causing large processor delays that negate the I/O gains.

VSAM buffering

The adequate use of buffering (mainly for direct access) can result in savings in the processor usage.

Increasing the number of buffers decreases the number of EXCPs, that is, the number of executed I/O operations. In VSAM, one EXCP corresponds to one SSCH and to one I/O operation. For the same amount of logic in your code, the processor time that you spend is a direct function of the number of EXCPs. Each I/O operation uses around 10,000 processor instructions. In the example that was run in the lab, the read data is not processed by any logic (read and forget), so it has these characteristics:

- SRB time is caused by I/O interrupts (back end) processing.
- TCB time includes the I/O operation preparation and buffer pool management.

Increasing the number of buffers means that the management cycles can obscure the savings in the I/O operations.

Also, the use of better techniques to manage your buffer pool, such as BLSR or SMB, can save processor cycles.

Use of extended format

The use of extended format data sets can save a consistent amount of processor time (TCB plus SRB). For more information, see 2.3, “Extended format” on page 64.

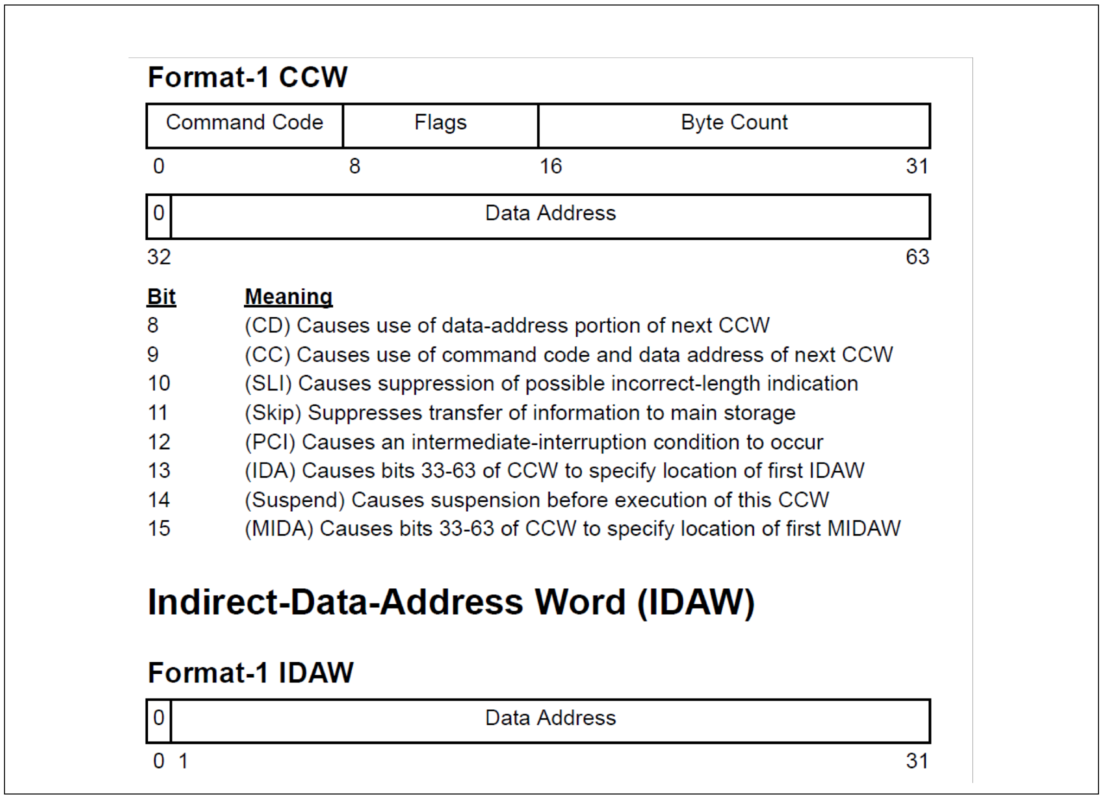
Compression

Compression usually increases processor usage in your program. However, in certain situations, the I/O savings can produce a processor usage reduction.

4.5.6 MIDAW facility

The MIDAW facility is a zArchitecture implementation that improves FICON performance. It does so mainly for extended format data sets (the ones with a 32-bytes suffix per block) including DB2 and VSAM. This 32-byte suffix is transparent to the application and is not stored (after read) at the end of the I/O buffer pool. On top of that, MIDAW can also decrease the FICON channel utilization for the same I/O workload. MIDAW is an enhancement of the previous IDAW facility.

IDAW was designed to solve an S/370 virtual storage problem. An I/O buffer (larger than 4 KB) is contiguous in the pages of the virtual storage and not contiguous in the frames of the central storage. Also, recall that the channel only deals with real addresses. The IDAW solution was to have a flag bit 13 (IDA bit)) at the CCW as shown in Figure 4-30. This bit tells the channel that the data address field at the CCW is not the real address of the I/O buffer. Rather, it is the real address of an indirect list (IDAL). Each IDAL entry (named IDAW) contains the real address of each piece of the I/O buffer. There is not a byte count in the IDAW, so the channel switches (during the data transfer) to another IDAW in the IDAL, every time that a 4 KB bound is reached.



Bit	Meaning
8	(CD) Causes use of data-address portion of next CCW
9	(CC) Causes use of command code and data address of next CCW
10	(SLI) Causes suppression of possible incorrect-length indication
11	(Skip) Suppresses transfer of information to main storage
12	(PCI) Causes an intermediate-interruption condition to occur
13	(IDA) Causes bits 33-63 of CCW to specify location of first IDAW
14	(Suspend) Causes suspension before execution of this CCW
15	(MIDA) Causes bits 33-63 of CCW to specify location of first MIDAW

Figure 4-30 One CCW and one IDAW

The performance issue with IDAW and extended format is caused by this 4-KB boundary. It is impossible to move the suffix to another central storage real address. To circumvent this problem, more “chain data CCWs” are added to the channel program, which increases the I/O connect time. Chain data CCW complexity is beyond the scope of this book.

The MIDAW solution is simple. Add a byte count to each entry in the indirect list as shown in Figure 4-31. With this implementation, you save one CCW per each block that is transferred in extended format. To activate MIDAW use the SETIOS, a z/OS console command. Point to an IECIOSxx Parmlib member with MIDAW=YES specified. YES is the default.

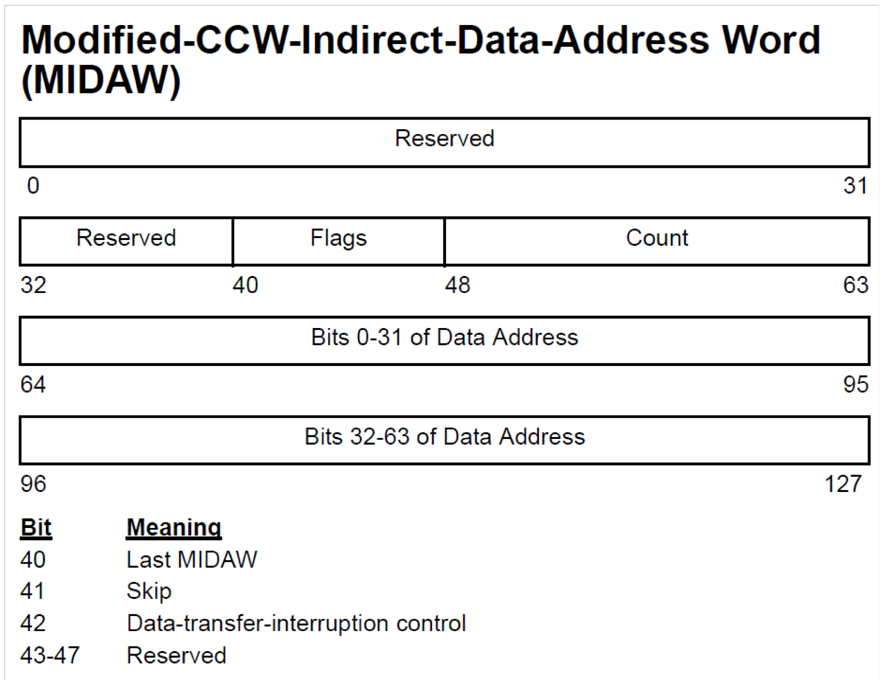

Modified-CCW-Indirect-Data-Address Word (MIDAW)
Reserved
0
31
Reserved
Flags
Count
32
40
48
63
Bits 0-31 of Data Address
64
95
Bits 32-63 of Data Address
96
127
Bit
Meaning
40
Last MIDAW
41
Skip
42
Data-transfer-interruption control
43-47
Reserved

Figure 4-31 MIDAW layout

4.6 Performance monitors

There are a number of performance monitors, whose output can help you in solving SMSVSAM performance problems.

4.6.1 Resource measurement facility

The RMF product issues reports about performance problems as they occur. This information allows your installation to take action before the problems become critical. Your installation can use RMF to run these functions:

- Determine that your system is running smoothly
- Detect system bottlenecks that are caused by contention for resources
- Evaluate the service that your installation provides to different groups of users
- Identify workloads that are delayed and the reasons for those delays
- Monitor system failures, system stalls, and failures of selected applications

RMF comes with three monitors, Monitor I, II and III:

- Monitor III can determine the cause of delay. It provides short-term data collection and online reports for continuous monitoring of system status and solving performance problems. It allows the system tuner to distinguish between delays for important jobs and delays for jobs that are not as important to overall system performance

- Monitor I provides long-term data collection for system workload and resource utilization. The Monitor I session is continuous, and measures various areas of system activity over a long time period.

  You can get Monitor I reports directly as real-time reports for each completed interval (single-system reports only). You can also run the Postprocessor to create the reports, either as single-system or as sysplex reports. Many installations produce daily reports of RMF data for ongoing performance management. Some reports are called Monitor I reports (for example, the Workload Activity report) although they can be created only by the Postprocessor.
- Monitor II provides online measurements on demand for use in solving immediate problems. A Monitor II session can be regarded as a snapshot session. Unlike the continuous Monitor I session, a Monitor II session generates a requested report from a single data sample. Because Monitor II is an ISPF application, you can use Monitor II and Monitor III simultaneously in split-screen mode to get different views of your system performance.

In addition, you can use the Spreadsheet Reporter for further processing the measurement data on a workstation with spreadsheet applications. The following chapters provide sample RMF reports.

4.6.2 Tivoli Decision Support

Tivoli Decision Support is a product that is contained in the Tivoli Information Management for z/OS suite of products.

Tivoli Information Management for z/OS is a process automation tool that delivers Systems Management Services, including Incident, Problem, Change, and Asset management applications, for your end-to-end managed environment.

With Tivoli Decision Support, you can also have immediate access to key enterprise information about performance and capacity, which is stored in a DB2 database. Tivoli Decision Support provides ready-to-use views and reports so that you can identify key problems. You can design your own reports and views by using SQL language code.

Information that is viewed with Tivoli Decision Support can be published on a website. It can also be published as an information ticker that streams across the user's desktop if your company uses broadcast tools. Users can then check the information before they call their internal help desk to report a problem.

4.6.3 Generalized trace facility

Generalized trace facility (GTF) is an MVS component that can capture detailed information about events that are occurring in the system. These events include SSCHs and I/O interruptions. If you have a complex VSAM DASD I/O performance situation, you can run GTF with CCW trace option as shown in the "REXX code to list compression ratio" on page 399. You can use IPCS to format the data that are produced by GTF, or you can use a custom product.

5

VSAM Record Level Sharing

This chapter introduces VSAM Record Level Sharing (RLS) concepts and describes how to implement RLS.

This chapter includes the following sections:

- Introducing VSAM RLS
- RLS terminology
- Planning for RLS
- Implementing VSAM RLS
- RLS problem determination and recovery
- Operational procedures for RLS
- RLS enhancements
- RLS performance
- RMF and VSAM RLS
- VSAM and high availability

5.1 Introducing VSAM RLS

This section introduces the basic concepts of VSAM RLS mode.

5.1.1 What is VSAM RLS?

VSAM RLS is a method of accessing your existing VSAM data sets. It allows any number of users within your Parallel Sysplex to share your existing VSAM data sets. It can provide full data integrity (read and write). The serialization is at record level. However, to implement recoverable VSAM data sets the users must have their own backout log, as CICS has.

VSAM RLS does not introduce new types of VSAM data sets. Rather, it introduces a new way of accessing existing data sets. Apart from the need to open data sets in RLS mode, the same VSAM record management interfaces (get, put, point, erase) are used.

You can specify the RLS mode in the MACRF parameter of the ACB macro that you use to open the data set in your program. You can also specify the RLS mode by using the new keyword RLS in your DD card that points to your VSAM data set in your JCL.

RLS mode can be used with KSDS, RRDS, VRRDS, and ESDS VSAM data sets. PATH access for KSDS and ESDS is allowed with RLS. Extended format, extended addressability, spanned, and compression are also supported by RLS. Beginning with z/OS 1.12, VSAM stripping is also supported for VSAM data sets being accessed in RLS mode.

RLS and non-RLS VSAM data sets can coexist.

5.1.2 Why RLS?

RLS allows concurrent access in a sysplex to your VSAM data sets at record level, while also maintaining data integrity.

VSAM RLS also addresses other VSAM issues, mainly in the CICS environment. For more information, see 5.1.5, “CICS and VSAM RLS” on page 213.

5.1.3 How does RLS work?

RLS logic is implemented in programs that run in a VSAM address space server named SMSVSAM. These programs gain the control from the RLS requesters through cross memory requests (PC instruction).

Figure 5-1 shows the SMSVSAM address space and two SMSVSAM-related data spaces in a z/OS image. In a common z/OS environment that uses VSAM RLS, you see several CICS *Application Owning Regions* (AOR) in several z/OS systems that access the same VSAM clusters for read/write. The worries about read and write integrity are moved from user responsibility to the SMSVSAM address space. Users can get rid of the CICS *file-owning region* (FOR). CICS FOR is a bottleneck and a single point of failure in a Parallel Sysplex. You can improve the availability of your sysplex by eliminating the need for the CICS FOR.

Batch jobs that use VSAM RLS can also open the same VSAM clusters.

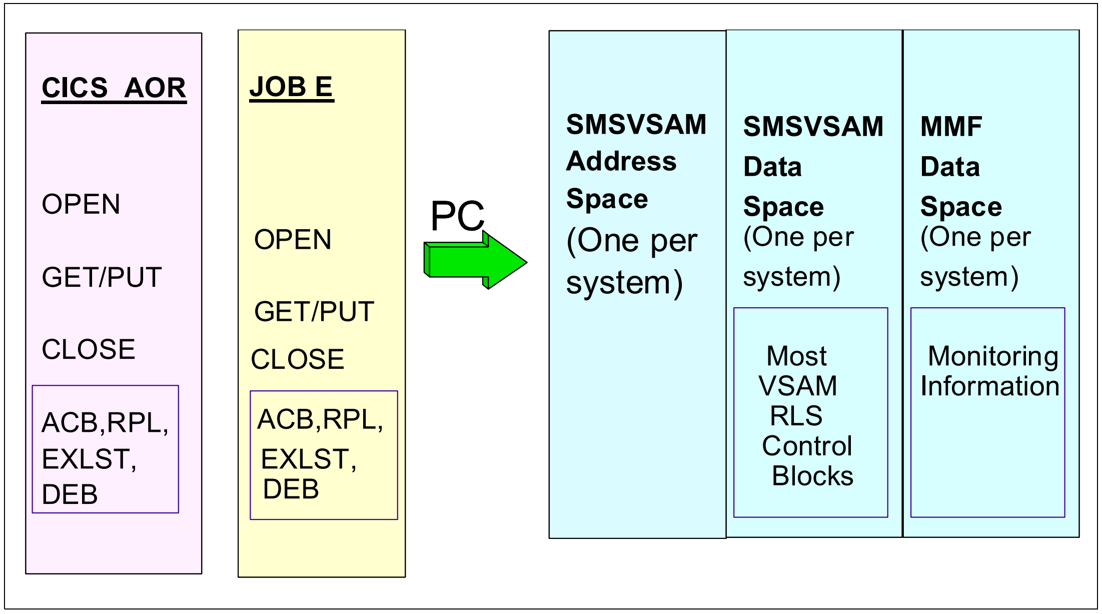

CICS AOR
OPEN
GET/PUT
CLOSE
ACB,RPL, EXLST, DEB
JOB E
OPEN
GET/PUT
CLOSE
ACB,RPL, EXLST, DEB
PC
SMSVSAM Address Space (One per system)
SMSVSAM Data Space (One per system)
Most VSAM RLS Control Blocks
MMF Data Space (One per system)
Monitoring Information

Figure 5-1 SMSVSAM Address Space and SMSVSAM Data Space

In RLS mode, all VSAM buffers are moved from the CICS or batch jobs address spaces to one of these locations:

- A 31-bit buffer pool in the SMSVSAM Data Space
- A 64-bit buffer pool in the SMSVSAM Address Space

These buffers are shared between all the address spaces in the same z/OS image.

Also, in RLS mode most of the VSAM control blocks are in the SMSVSAM Data Space, except for the ACB, RPL, and EXLST.

Figure 5-2 shows in a simplified form how a VSAM data set is shared between various address spaces (CICS and Batch jobs) across a Parallel Sysplex. Each z/OS system in the sysplex has an SMSVSAM address space to coordinate the sharing. Sharing Control Data Set (SHCDS) contains critical information that is used by various SMSVSAM address spaces for RLS. The coupling facility contains these features:

- Lock structure IGWLOCK00, which contains global locks to serialize at record level and to serialize functions as splits
- Secondary lock structures, if multiple lock structures support is implemented
- Cache structures that are used to hold shared data and control block structures

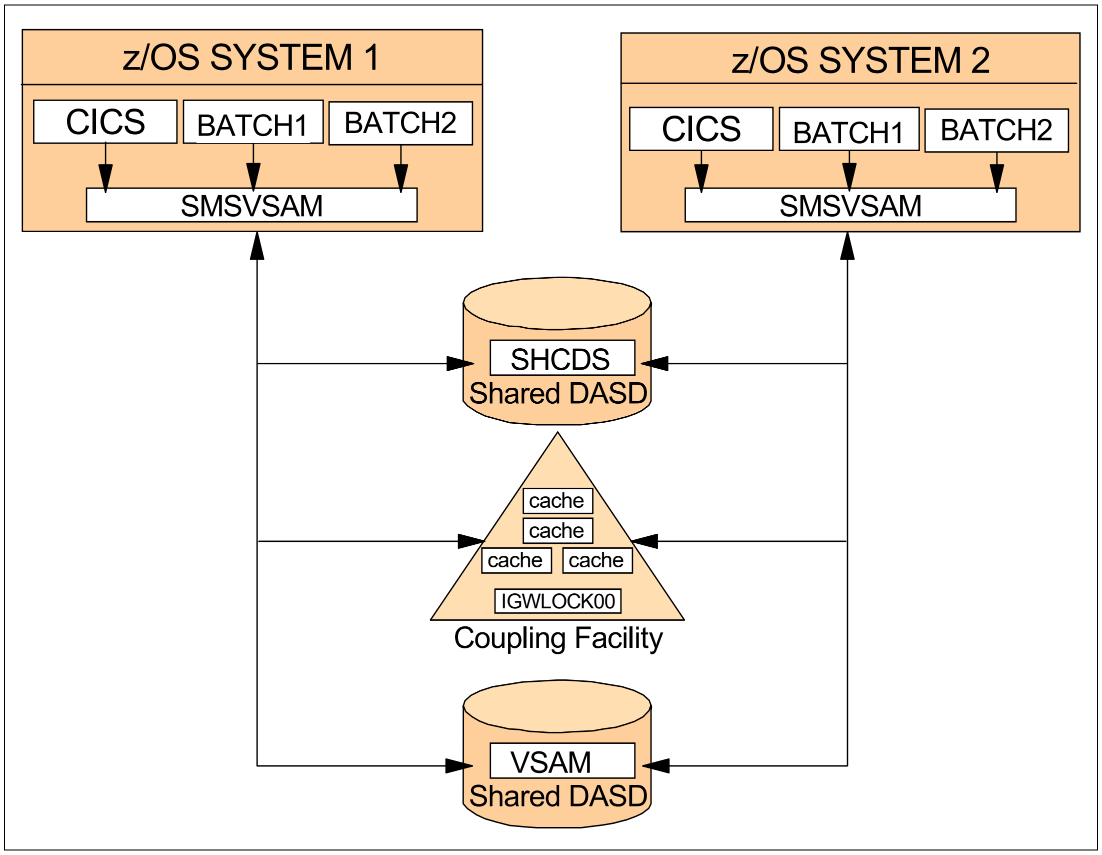


Figure 5-2 RLS in Sysplex

5.1.4 RLS in a single system (monoplex)

You can access VSAM data sets in RLS mode from users that run in just one z/OS image. However, even in a monoplex environment a Coupling facility is required. Figure 5-3 shows an example of such a setup.

The reason for doing that is to use the better serialization that is provided by RLS. For example, in a non-RLS mode two jobs cannot simultaneously access for update a VSAM data set with total read and write integrity. That is, for SHAREOPTIONS (1,3), just one can open the VSAM data set. The other must wait for the first to close the data set. The granularity of the serialization is at data set level.

If the two jobs access the data set in RLS mode, both the OPENs succeed. Both in parallel can then update the VSAM data set. The serialization is at logical record level.

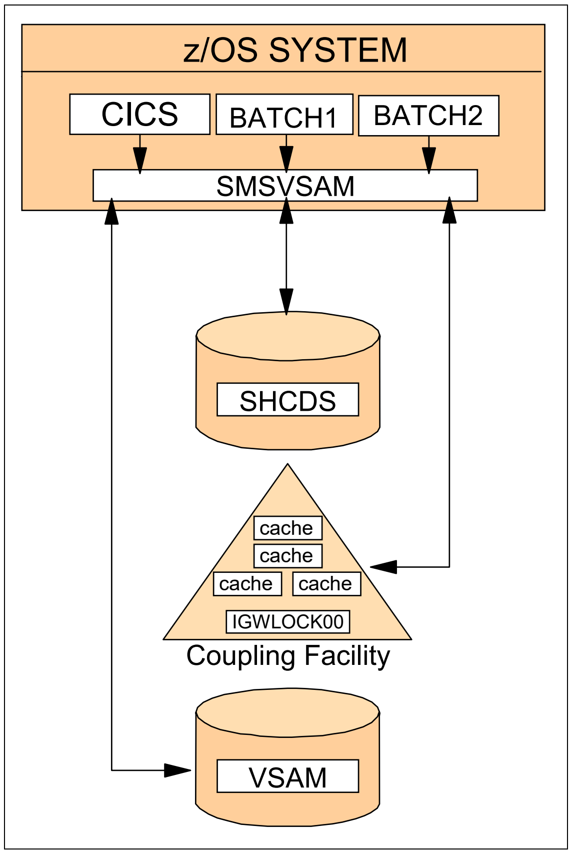


Figure 5-3 RLS in Monoplex

5.1.5 CICS and VSAM RLS

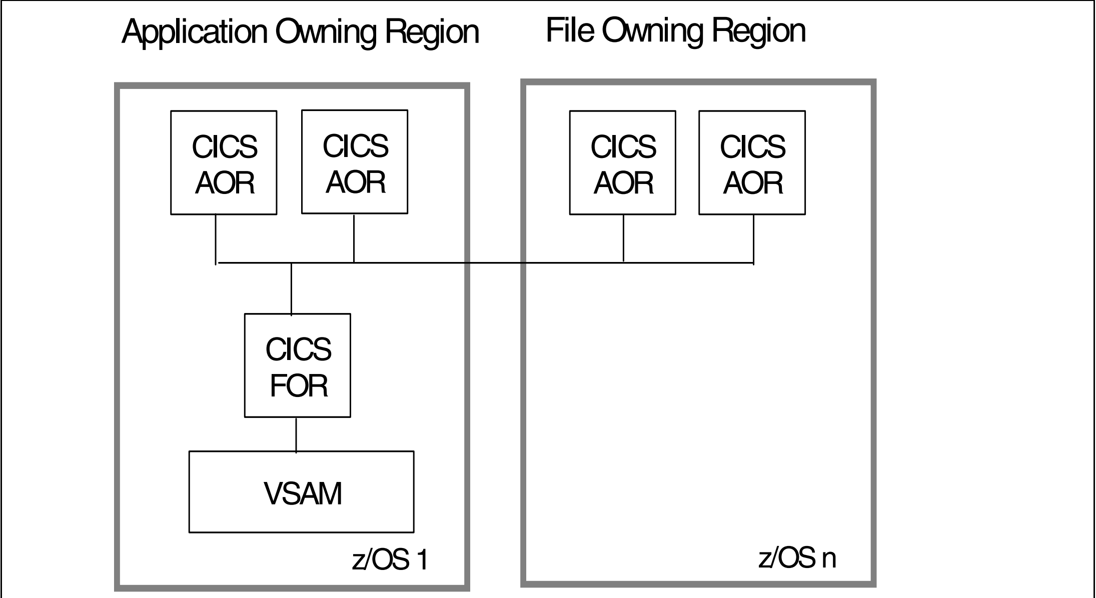
Without VSAM RLS, CICS uses a process that is called *function shipping* to share data sets between CICS address spaces (regions). Function shipping is implemented by using a single CICS region that is called an FOR to own a data set. All access to the data set is through the corresponding FOR, which is a focal point for VSAM access. The role of the FOR is to provide file access. The CICS transactions run elsewhere in application-owning regions (AORs).

These regions run programs which, when they want to access a shared VSAM data set, ship the access request off to the FOR for that data set. They then await the reply from the FOR. This process is illustrated in Figure 5-4.

This implementation has several problems:

- Function shipping between AORs and FORs in distinct systems, which uses resources.
- The FOR is a single point of failure and can also be a processor bottleneck
- Locks are held by CICS/FOR. An in-doubt lock that is held by a CICS task can cause an integrity problem if FOR fails. In this case, there is no way to prevent other applications from accessing uncommitted data. With RLS, locks are held by SMSVSAM, not by CICS/FOR. Therefore, the locks that are held by an in-doubt CICS task can be held if a CICS fails. This configuration prevents other applications from accessing uncommitted data until CICS is restarted or the locks are released by an operator command.
- Need to commit remotely.
- HURBA/HARBA are updated only when CICS closes the data set, causing conflicts along shared updates.

These problems are fully addressed by VSAM RLS.



Problems:

- CICS FOR is a single point of failure
- Multiple system performance is no acceptable (uses VTAM or XCF cross system)
- No exploitation of System/390 Parallel Sysplex

Figure 5-4 CICS before VSAM RLS

CICS with VSAM RLS is shown in Figure 5-5. The RLS data sharing environment is designed to support sharing of VSAM data sets among multiple cloned AORs. With VSAM RLS, you do not need FORs. Each z/OS system contains an SMSVSAM address space that manages locks within a coupling facility. Coupling facility configurations can be highly available. As you no longer have file owning regions, the single point of failure is eliminated. You can also scale by adding z/OS systems to the parallel sysplex, with each new system running an SMSVSAM address space.

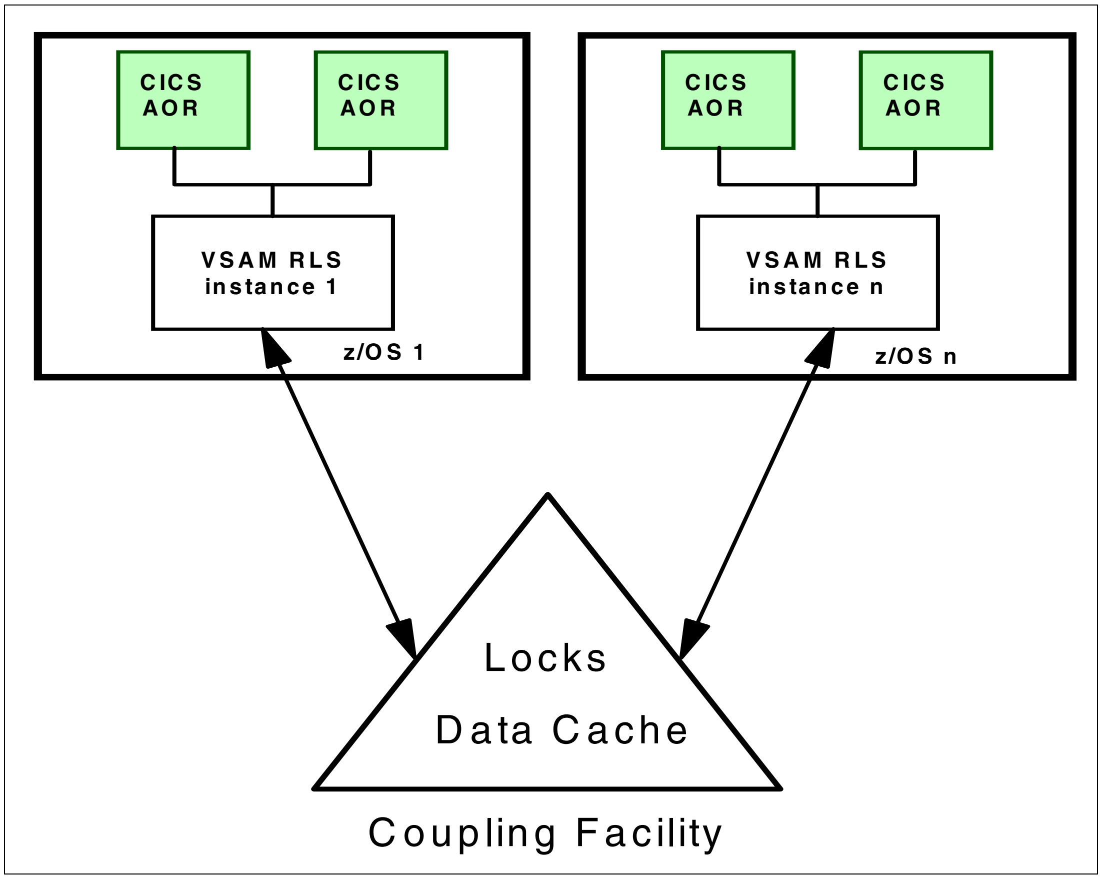


Figure 5-5   CICS after VSAM RLS

For more information about VSAM RLS and CICS, see the following publications:

- *CICS and VSAM Record Level Sharing: Implementation Guide*, SG24-4766
- *CICS and VSAM Record Level Sharing: Planning Guide*, SG24-4765
- *CICS and VSAM Record Level Sharing: Recovery Considerations*, SG24-4768

5.1.6 SHAREOPTIONS and VSAM RLS

VSAM SHAREOPTIONS are ignored when you access your data set in RLS mode. You can have any number of read/write users who can access the VSAM data set in RLS mode.

There is one exception to this rule about SHAREOPTIONS(2 x). VSAM data sets defined with SHAREOPTIONS(2 x) can be accessed by any number of read/write users in RLS mode. They can also be accessed by any number of read users in non-RLS mode. However, the non-RLS users do not have any read integrity. A read user in non-RLS mode can be a batch job for printing the data set. Or it can use the IDCAMS repro to create a copy of the data set for test purposes.

5.1.7 RLS restrictions

RLS does not support the following options and capabilities:

- Linear, KEYRANGE, IMBED, and temporary data sets
- Addressed access to KSDS data sets
- Control Interval Access
- User Buffering (UBF)
- 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, RLS has the following restrictions:

- You cannot specify RLS access when accessing a VSAM data set by using the ISAM interface to VSAM.
- 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 running in these modes: Cross-memory mode, SRB mode, or under an FRR.
- RLS does not support UNIX files.

5.2 RLS terminology

The terminology that is used in RLS books is sometimes different from the terms that are used more widely in z/OS and SMS. Therefore, it is worthwhile to define the RLS terms to avoid confusion.

5.2.1 SMSVSAM

SMSVSAM is the z/OS job name of the RLS address space. You must start an SMSVSAM address space in all z/OS systems that are going to access VSAM data sets in RLS mode. All the SMSVSAM address spaces talk to their peers in the Parallel Sysplex using XCF, sharing locks, information, and data by using the Coupling Facilities.

5.2.2 VSAM sphere

A sphere is the name that is used to represent the components (data/index) and all the AIX of several related VSAM data sets.

5.2.3 RLS client

An RLS client is any address space that starts an RLS function that results in a call to the SMSVSAM address space. Examples of RLS functions are OPEN, CLOSE, GET, PUT, and DELETE. Examples of application that can be registered as RLS client address spaces are CICS, batch jobs, and DFSMShsm.

5.2.4 Subsystem

A subsystem is an RLS client space that “registers” with the SMSVSAM address space as an address space that provides recovery (that is, forward/backward logging). CICS is an example of a recoverable subsystem.

5.2.5 Batch

A batch is an RLS client address space that does not register with SMSVSAM as a recoverable subsystem. A traditional batch job does not provide any logging, but it can be modified to use the DFSMStvs logging facilities. Examples of batch address spaces are non-CICS VSAM applications, DFSMSdss backup and copy, DFSMShsm, and other user-written applications.

5.2.6 Record lock

A record lock is an XES lock resource that is obtained by SMSVSAM on behalf of a user and associated with a logical record. The lock resource name is based on a 16-byte hashed version of the record's key (or RBA, or RRN), and the data set name and component name. There are also other locks to serialize splits, for example.

SMSVSAM maintains two different types of record-level locks in its coupling-facility lock structures: Exclusive locks, and shared locks. Exclusive locks are used for any update request, whereas shared locks are used to support read integrity. The lock includes the name of the corresponding lock owner, which is the application ID (CICS, DFSMStvs or any other), and the unit of work ID.

5.2.7 True contention

True contention is when two different users attempt to access the same record lock at the same time. This situation is listed as true contention on the D SMS,CFLS command and in RMF reports. You must tune your application if you have high true contention rates, or use NRI for reads.

5.2.8 False contention

False contention occurs when VSAM RLS assigns locked resources to an entry value in the lock table, and uses this entry value to quickly check whether a resource is already locked. If the lock structure (and thus the lock table) is too small, many locks can be represented by a single value, making “false” lock contention possible. False lock contention occurs when two different locks on different resources attempt to use the same lock entry. The second lock requester is suspended until VSAM RLS determines that there is no real lock contention on the resource.

This situation is reported as false contention on the D SMS,CFLS command. If you have high false contention rates, increase the size of your lock structures.

5.2.9 True/False or False/False contention

This contention occurs when either true contention or false contention occurs, and the holder releases the record before the RLS contention exit runs. RLS cannot tell between true or false contention. This type of contention is reported as false contention on the D SMS,CFLS command. See Figure 5-31 on page 256.

5.2.10 Deadlocks

Deadlocks occur when two different transactions, each holding a record lock, attempt to obtain the other transaction's record lock. SMSVSAM abnormally terminates one of the waiting lock requests (RPL RC=8 RSN=21), based on the Deadlock_detection value in your IGDSMSxx parmlib member.

5.2.11 Read integrity

SMSVSAM allows the user to decide about read integrity through parameters in DD card or ACB macro. If the option is consistent read, logical records locks are required in shared mode for reads. If the option is no read integrity (NRI), no locks are required along reads.

5.2.12 Retained lock

A retained lock is a record lock obtained by a recoverable subsystem such as CICS, still held at the time of a failure. The failure can be in CICS, RLS CF lock structures, CF, z/OS, other SMSVSAM, or Indoubt transactions. A job can be canceled without retained locks because this job does not register as a subsystem with SMSVSAM.

5.2.13 Lost Locks

A VSAM data set is said to be in "Lost Locks" if the data set was being accessed by a recoverable subsystem when a failure to the lock structure occurs at the same time as a failure in at least one of the SMSVSAM address spaces (double failure scenario). Use the IDCAMS SHCDS LISTSUBSYS(ALL) command to list CICS subsystems that are holding retained or lost locks. This situation can be avoided by duplexing the lock structure.

5.2.14 Recoverable VSAM data set

This is a VSAM data set that is accessed by a recoverable subsystem (that is, CICS) that provides logging for the data set. A recoverable data set is defined by the LOG(UNDO/ALL) attribute in the catalog.

5.2.15 Non-recoverable VSAM data set

This is a VSAM data set for which no logging is required. A non-recoverable data set is defined with the LOG(NONE) attribute in the catalog.

5.2.16 Quiesce a VSAM data set

This special interface is provided by RLS for CICS to close a data set across the sysplex. QUIESCE = YES is set in the catalog after the quiesce request. This command shows the quiesce state of a VSAM data set:

```
D SMS,SMSVSAM,QUIESCE
```

5.2.17 Quiesce transactions

This special interface provided to CICS and DFDSS temporarily halts transactions for a data set in order for DFDSS to take a sharp copy (also called a T0 copy).

5.2.18 Quiesce a volume or a cache

This command to SMSVSAM stops new opens for RLS to a particular volume or cache. The commands are as follows:

```
V SMS,CFVOL(volid),QUIESCE
V SMS,CFCACHE(cachename),QUIESCE
```

5.2.19 Sharing control

This set of routines runs in the SMSVSAM address space that opens, formats, reads, and writes to the SHCDS. SHCDS are linear data sets that are accessed by SMSVSAM in non-RLS mode. They contain “state” type information about RLS client address spaces and VSAM data sets accessed by SMSVSAM.

5.2.20 RLS mode

A data set is said to be in RLS mode when it was last accessed by RLS. RLS mode is indicated by the RLS-IN-USE indicator in the catalog.

5.3 Planning for RLS

This section addresses planning activities that are needed to implement RLS.

5.3.1 Hardware requirements

VSAM RLS uses the coupling facility (CF) to run data set level locking, record locking, and data caching. If your system is running in a sysplex, you already have the necessary hardware to implement VSAM RLS.

To implement system-managed duplexing for RLS lock structures for improved availability, you need more CF storage, CF processor capacity, and z/OS-to-CF link capacity. This additional capacity is need to accommodate duplexed VSAM RLS lock structure instances, and the duplexed VSAM RLS lock structures operations to them from each participating z/OS image.

Consideration: If you want to implement VSAM RLS on a stand-alone z/OS system to share VSAM across multiple address spaces, do not run in XCFLOCAL mode. A coupling facility is still needed for the lock structures even though there is only a single z/OS image.

5.3.2 Software requirements

VSAM data sets must be SMS-managed to be accessed in RLS mode. Also, evaluate if your application programs can tolerate or use VSAM RLS. Regarding RLS, application programs can be classified into three categories:

- VSAM RLS intolerant programs: These programs use facilities that VSAM RLS does not support, access VSAM internal data structures, or are incompatible with VSAM RLS sharing.
- VSAM RLS tolerant programs: The program operates correctly in a multiple updater environment when you specify RLS in the job control language (JCL) or MACRF=RLS in the access method control block (ACB). The RLS parameter (in JCL) allows batch programs to use RLS without requiring the program to be recompiled. You can change the DISP=OLD to DISP=SHR for batch update programs that run against nonrecoverable data sets.
- VSAM RLS using programs: The program recognizes a VSAM RLS accessible data set and uses VSAM RLS to access the data set. CICS uses VSAM RLS.

  A batch program can use VSAM RLS to read-only share a recoverable data set with CICS address spaces that are updating the data set. VSAM RLS provides read integrity for the batch program.

  Batch programs can access recoverable data sets for read/write in RLS mode, by using the logging and commit services that are provided by DFSMStvs.

5.4 Implementing VSAM RLS

This section addresses the step-by-step procedure to implement VSAM RLS.

5.4.1 Defining Sharing Control Data Sets (SHCDS)

This section provides some considerations and examples to define SHCDS.

What is its function?

The SHCDS is critical to maintaining data integrity in the event of failures of the Parallel Sysplex, SMSVSAM address space, or the CF lock structure.

The SHCDS contains the following information:

- The name of the CF lock structure in use
- A list of subsystems
- The status of the subsystems
- A list of open data sets that use the CF
- Other information

How do you define it?

Use the following naming convention when you are defining your SHCDS:

`SYS1.DFPSHCDS.`*qualifier*`.`*Vvolser*

Where:

qualifier is a 1 - 8 character qualifier.

volser is the volume serial number. The V prefix allows you to specify numeric volume serial numbers.

Restriction: The SHCDS naming convention depends on the volume to match the SHCDS name. Therefore, do not move SHCDS across volumes.

How much space to allocate for SHCDS?

Use the following formula to calculate the size of your SHCDS:

```
S = (16 + (N * (16 + C/10))) kilobytes
```

Where:

S is the space that is required for the SHCDS.

N is the number of systems.

C is the number of concurrent OPEN requests that you expect.

Sample JCL to allocate SHCDS

The SHCDS can be defined by using IDCAMS as shown in Example 5-1.

Example 5-1 Allocating SHCDS by using IDCAMS

```
//ALLOCLD1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DEFINE CLUSTER (NAME(SYS1.DFPSHCDS.PRIMARY.VSMS001) LINEAR -
          STORCLAS(GSPACE)                            -
          SHAREOPTIONS(3 3)   CYL(10 10) VOLUME(SMS001) )
    DEFINE CLUSTER (NAME(SYS1.DFPSHCDS.SECONDARY.VSMS002) LINEAR -
          STORCLAS(GSPACE)                            -
          SHAREOPTIONS(3 3)   CYL(10 10) VOLUME(SMS002) )
    DEFINE CLUSTER (NAME(SYS1.DFPSHCDS.SPARE.VSMS003) LINEAR -
          STORCLAS(GSPACE)                            -
          SHAREOPTIONS(3 3)   CYL(10 10) VOLUME(SMS003) )
/*
```

Example 5-2 shows how you can also use DD statements in JCL to allocate these data sets. Use a data class that is defined with a cross-region share options value of 3 and a cross-system share options value of 3.

Example 5-2 Allocating SHCDS by using JCL DD statements

```
//PRISHCDS DD DSN=SYS1.DFPSHCDS.PRIMARY.VSMS001,SPACE=(1,(10,10)),
//            RECORG=LS,STORCLAS=GSPACE,VOL=SER=SMS001,AVGREC=M,
//            DISP=(NEW,CATLG)
//SECSHCDS DD DSN=SYS1.DFPSHCDS.SECONDRY.VSYS002,SPACE=(1,(10,10)),
//            RECORG=LS,VOL=SER=SYS002,AVGREC=M,
//            DISP=(NEW,CATLG)
//SPRSHCDS DD DSN=SYS1.DFPSHCDS.SPARE.VSMS003,SPACE=(1,(10,10)),
//            RECORG=LS,STORCLAS=GSPACE,VOL=SER=SMS003,AVGREC=M,
//            DISP=(NEW,CATLG)
/*
```

SHCDS considerations

Take into account these considerations when you are allocating the SHCDS:

- At a minimum, define and activate two SHCDSs and at least one spare SHCDS for recovery purposes.
- Because the contents of these data sets are highly dynamic, do not back up and restore functions for these data sets.
- The SHCDS is a VSAM linear data set.
- The CISIZE for SHCDS must be 4096.
- When defined, the SHCDS does not need to be cataloged on all systems in the sysplex. If it is cataloged, it must be in a catalog available when SMSVSAM initializes.
- Place the SHCDSs on volumes with global connectivity.
- The share options for SHCDSs must be set to (3,3) so that each system in the Parallel Sysplex can properly share the data sets.
- Use storage classes that are defined with the guaranteed space attribute.
- Avoid placing SHCDSs on volumes that might have extensive volume reserve activity.
- Generally, use secondary space definition. All extents for each data set *must* be on the same volume.
- SMSVSAM remembers the SHCDSs from one SMSVSAM recycle to the next. Do not delete or redefine an active a spare SHCDS without first telling SMSVSAM. Use the V SMS,SHCDS(shcdsname),DELETE command.
- SMSVSAM must be RACF authorized to update SYS1.DFPSHCDS.* data sets. For more information, see “Security definitions” on page 236.
- For some SHCDS errors, an IPL does *not* help because SHCDS is remembered from one IPL to another. The FALLBACK procedure that is documented in “SHCDS recovery procedures” on page 246 is the only way to correct the problem. If you get repeated abends with the prefix '67' in the RSN code, perform a FALLBACK.

Attention: Use the FALLBACK procedure only as the last resort when everything else fails.

5.4.2 Defining CF cache structures

VSAM RLS uses the coupling facility (CF) in the sysplex to maintain cache and lock structures to administer VSAM RLS. CF is accessible from all the z/OS systems. You can define several cache structures, and one or more lock structures.

What is its function?

The CF cache structure maintains RLS local buffer pool consistency at the control interval (CI) level of all SMSVSAM instances in a Parallel Sysplex. CF cache structures contain the control block structure being used as a system buffer pool for VSAM RLS. This configuration provides a level of storage hierarchy between local storage buffers and DASD cache. That is, RLS uses the cache structure for store-through caching. Along writes data is written from local buffer pool to DASD and to the cache structure.

Determining the cache size

The total size of the cache structures should be equal to the sum of the local VSAM LSR buffer pool sizes. After you get that figure, divide that size among the structures that will be defined. You can use the sizing tool available at:

http://www-1.ibm.com/servers/eserver/zseries/cfsizer/vsamrls.html

Use CFRM policy definitions to specify an initial and maximum size for each CF cache structure. DFSMS uses the initial structure size that you specify in the policy each time it connects to a CF cache structure. If more space is required, RLS alters the cache, up to the maximum size specified. Do not specify ALLOWAUTOALT(YES) for RLS cache structures, which allow system-initiated alters. Doing so prevents RLS from managing the cache structure.

Defining the cache structure

Use the IXCMIAPU utility program to define the cache structures in the CFRM couple data set. Coordinate this activity with your z/OS systems programmer. Example 5-3 shows a sample JCL that defines the cache structures.

Example 5-3 JCL to define RLS cache structures

```
//STEP01   EXEC PGM=IXCMIAPU
//STEPLIB  DD   DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=H
//SYSIN    DD   *
     DATA TYPE(CFRM) REPORT(YES)
     DEFINE POLICY NAME(yourpolicy) REPLACE(YES)
       STRUCTURE NAME(CACHE01)
                 SIZE(4000)
                 INITSIZE(3000)
                 PREFLIST(yourCF1)
       STRUCTURE NAME(CACHE02)
                 SIZE(4000)
                 INITSIZE(3000)
                 PREFLIST(yourCF2)
/*
```

Multiple cache structures

After you determine the total CF cache that is required, you might need to break it down into individual cache structures. You can define multiple SMSVSAM cache structures on different CFs. Assign one or more CF cache structures to each *cache set* that is associated with a storage class.

The basic structure has these characteristics:

1. The VSAM data set has a storage class
2. The storage class has the cache set name
3. The cache set (in base SMS ACDS) has the names of the cache structures

Having multiple cache sets allows you to provide different performance attributes for data sets with distinct performance requirements. When more than one CF cache structure is assigned to a cache set, data sets within that storage class are cached in each CF structure in turn. This structure helps SMSVSAM to balance the load.

Considerations

Cache structures are built at open time, and remain even after the close of a data set because of performance reasons (kept attribute). RLS uses LRU algorithms to release the structures.

The cache must be large enough for the CF cache directories to contain an entry for each of the VSAM RLS local buffers across all instances of the RLS server.

Enhancements allow VSAM data sets with a CI size greater than 4-KB records to be cached.

5.4.3 Defining the primary CF lock structure

This section explains how to define the primary lock structure.

What is its function?

The primary CF lock structure maintains the record-level locks and data-set-level locks to enforce global serialization. The CF lock structure includes two parts:

- The lock table, which is used to determine whether there is R/W interest among systems on a particular resource
- Record table space to track information for retained locks and data sets that are processed by VSAM RLS

Defining the primary CF lock structure

The primary CF lock structure is named IGWLOCK00. You can use the same IXCMIAPU utility that you used for CF cache structure to define the lock structure in the CFRM couple data set. See Figure 5-6 for a sample JCL.

```
//STEP01   EXEC PGM=IXCMIAPU
//STEPLIB  DD   DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=H
//SYSIN    DD   *
     DATA TYPE(CFRM) REPORT(YES)
     DEFINE POLICY NAME(CONFIG1) REPLACE(YES)
              STRUCTURE NAME(IGWLOCK00)
                 SIZE(4000)
                 PREFLIST(CF01)
```

Figure 5-6 Defining a CF lock structure

Determining the lock structure size?

You can use the following formula to estimate an initial size for the lock structure.

```
Size in Mega Bytes=10 * number_of_systems * lock_entry_size
```

The lock_entry_size depends on the MAXSYSTEM value that was defined for the sysplex couple data set when it was initially defined by the IXCL1DSU format utility. The MAXSYSTEM value represents the maximum number of systems that can exist in the Parallel Sysplex.

The lock_entry_size is 2 if the MAXSYSTEM is 7 or less. It is 4 if MAXSYSTEM >= 8 and < 24 and 8 if MAXSYSTEM >= 24 and <= 32.

You can issue the following command to determine the MAXSYSTEM value:

```
DISPLAY XCF,COUPLE,TYPE=CFRM
```

Figure 5-7 shows the output of the command.

```
-D XCF,COUPLE,TYPE=CFRM
 IXC358I  15.22.48  DISPLAY XCF 780
 CFRM COUPLE DATA SETS
 PRIMARY    DSN: SYS1.XCF.CFRM04
            VOLSER: SBOX67     DEVN: 3F39
            FORMAT TOD         MAXSYSTEM
            11/29/2001 13:36:18        4
            ADDITIONAL INFORMATION:
             FORMAT DATA
             POLICY(5) CF(4) STR(100) CONNECT(32)
             SMREBLD(1) SMDUPLEX(1)
 ALTERNATE  DSN: SYS1.XCF.CFRM05
            VOLSER: SBOX68     DEVN: 3B39
            FORMAT TOD         MAXSYSTEM
            11/29/2001 13:36:19        4
            ADDITIONAL INFORMATION:
             FORMAT DATA
             POLICY(5) CF(4) STR(100) CONNECT(32)
             SMREBLD(1) SMDUPLEX(1)
             CFRM IN USE BY ALL SYSTEMS
```

Figure 5-7 Displaying MAXSYSTEM

You can also use the CFSIZER sizing tool available at:

`http://www-1.ibm.com/servers/eserver/zseries/cfsizer/vsamrls.html`

For primary and secondary lock structures, the formula and the cfsizer estimated values are guidelines only. When the false contention rate for a lock structure is 0.5% or more, increase the lock structure size. For more information, see 5.8.6, “RLS lock structure sizing” on page 279.

Considerations on lock contentions

Take these considerations into account when you are defining the lock structure:

- You must monitor false contention to adjust the lock structure size. False contention can be monitored by using RMF or the DISPLAY SMS,CFLS command.
- The amount of real lock contention is application-dependent. It depends on record access patterns. False lock contention is almost entirely determined by the size of the lock table. That is, a larger lock table has less false lock contention than a smaller one.

5.4.4 Defining secondary lock structures (optional)

The *multiple lock structures* for VSAM RLS was first delivered in z/OS 1.10. The goal of multiple lock structures is to minimize the effect of a failure in IGWLOCK00 lock structure, and also isolate different workloads in different lock structures. You can assign lock structures to groups of VSAM data sets that belong to different applications.

With multiple lock structures support, IGWLOCK00 is still used as the primary lock structure. The structure contains the following items:

- Record locks and record data for data sets that are not using the new multiple lock structures support
- System special locks: Sphere, component, subsystem locks, and data set record data
- Lock structures lock (associates data sets with lock structures)

The secondary lock structures contain record locks and record data for the data sets that use the multiple lock structures support.

Figure 5-8 shows an RLS environment that uses two secondary lock structures:

- RLSLOCK_PROD00 in CF1
- RLSLOCK_TEST00 in CF2

Production DataSet1 is assigned to SMS Storage Class SCPROD. This storage class is assigned to a lock set where structure RLSLOCK_PROD00 is defined in the SMS configuration.

However, test DataSet2 is assigned to Storage Class SCTEST, which is assigned to a different lock set where structure RLSLOCK_TEST00 is defined. Activities on Test applications on DataSet2 do not interfere with the Productions application on DataSet1.

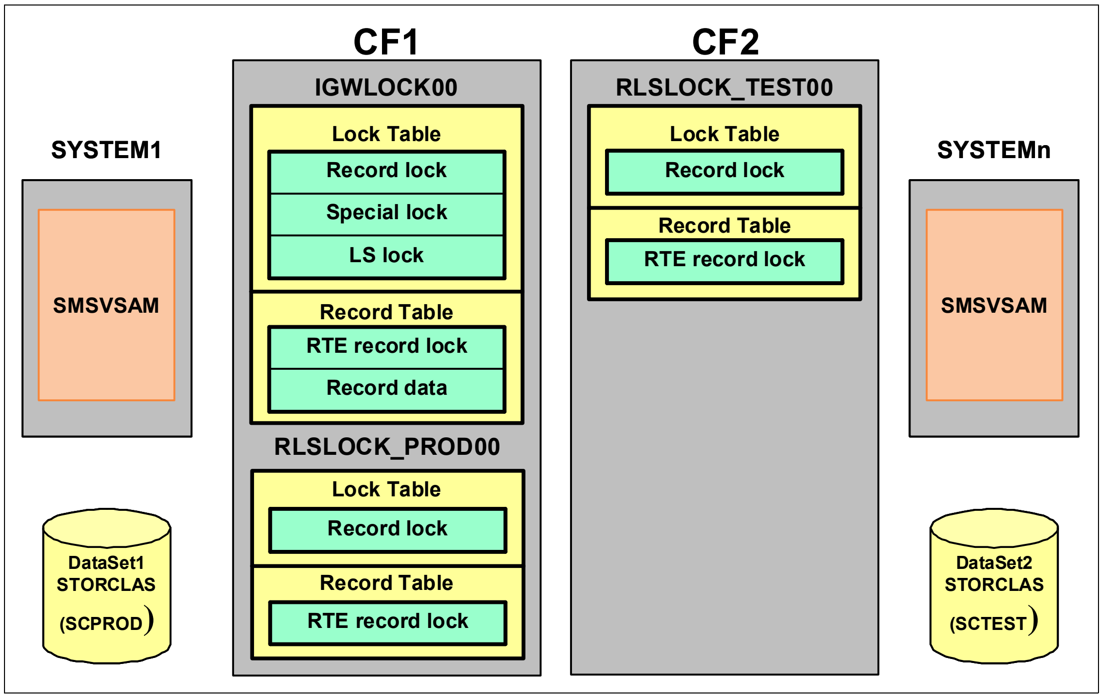


Figure 5-8 Multiple lock structures

You can use the IXCMIAPU utility to define the secondary lock structures in the CFs that you intend to use.

Figure 5-9 shows an example of how to define two lock structures:

- RLSLOCK_PROD00 in CF1
- RLSLOCK_TEST00 in CF2

You can calculate the size of any secondary lock structure by using the same formula as in “Determining the lock structure size?” on page 224. You can also access the CFSIZER tool by using the same criteria that you used for defining the IGWLOCK00 structure. The CFSIZER tool can be found at:

http://www-1.ibm.com/servers/eserver/zseries/cfsizer/vsamrls.html

For primary and secondary lock structures, the formula and the cfsizer estimated numbers are guidelines only. When the false contention rate for a lock structure is 0.5% or more, increase the lock structure size. For more information, see 5.8.6, “RLS lock structure sizing” on page 279.

```
//STEP01   EXEC PGM=IXCMIAPU
//STEPLIB  DD   DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=H
//SYSIN    DD   *
     DATA TYPE(CFRM) REPORT(YES)
     DEFINE POLICY NAME(CFRM41) REPLACE(YES)
              STRUCTURE NAME(RLSLOCK_PROD00)
                 SIZE(4000)
                 PREFLIST(CF1)
              STRUCTURE NAME(RLSLOCK_TEST00)
                 SIZE(4000)
                 PREFLIST(CF2)
```

Figure 5-9 Defining secondary lock structures

After you define these structures, activate the updated CFRM policy, and update your SMS configuration as described in “Defining the secondary lock structures to SMS” on page 229.

5.4.5 SMS definitions

This section defines which CF cache structures and secondary lock structures can be used to assign a VSAM data set.

Defining the CF cache structures names to SMS

SMSVSAM uses the SMS storage class construct that you define to determine which CF cache structures (varying sizes and performance levels) can be used to assign a VSAM data set to a CF cache structure. Based on this determination, group similar (1 - 255) cache structures into cache sets. The storage classes SC54GRT and SCRLS have a pointer to a “cache set” named CSERLS.

These cache sets are defined in the BASE SMS construct, and contain a list of CF cache structures. In the example, these structures are RLS_CACHE1 and RLS_CACHE2.

A VSAM data set that is opened in RLS mode must have a storage class assigned to it as shown in Figure 5-10. The first time an RLS VSAM data set is opened in the Sysplex, SMVSAM selects the *best* cache structure to use for the requested VSAM data set. SMVSAM then attempts to connect to it.

```
Panel  List  Utilities  Scroll  Help
------------------------------------------------------------------------------
                               STORAGE CLASS LIST
Command ===>                                                  Scroll ===> CSR
                                                      Entries 89-99 of 116
                                                      Data Columns 15-18 of 22
CDS Name : SYS1.SMS.MHLRES3.SCDS

Enter Line Operators below:

     LINE       STORCLAS SUSTAINED DATA  CF CACHE  CF DIRECT  CF SEQUENTIAL
     OPERATOR   NAME     RATE (MB/SEC)   SET NAME  WEIGHT     WEIGHT
    ---(1)----  --(2)--- -----(15)-----  --(16)--  --(17)---  ----(18)-----
                SCMSR9              ---  --------         --             --
                SCPAV               ---  --------         --             --
                SCPDSE              ---  --------         --             --
                SCPROD              ---  CSERLS            6              6
                SCRLS               ---  CSERLS            6              6
                SCSAHOO             ---  --------         --             --
                SCSDR1              ---  --------         --             --
                SCSDR12              12  --------         --             --
                SCSDR32              32  --------         --             --
                SCTEST              ---  --------         --             --
                SC54GRT             ---  CSERLS            5              3
```

Figure 5-10 CF Cache Set Name to Storage Class assignment

Figure 5-11 shows the CF cache structure names assigned to cache set CSERLS. In this example, only cache structure RLS_CACHE is associated with cache set CSERLS.

```
Panel  Utilities  Scroll  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                              CF CACHE SET DISPLAY                PAGE 1 OF 1
Command ===>

SCDS Name  : SYS1.SMS.MHLRES3.SCDS
                                         ( 001  Cache Sets Currently Defined )

 Cache Set                      CF Cache Structure Names
 CSERLS    RLS_CACHE




Use UP/DOWN Command to View other Pages when multiple Pages are Shown;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-11 Cache Set to Cache Structure assignment

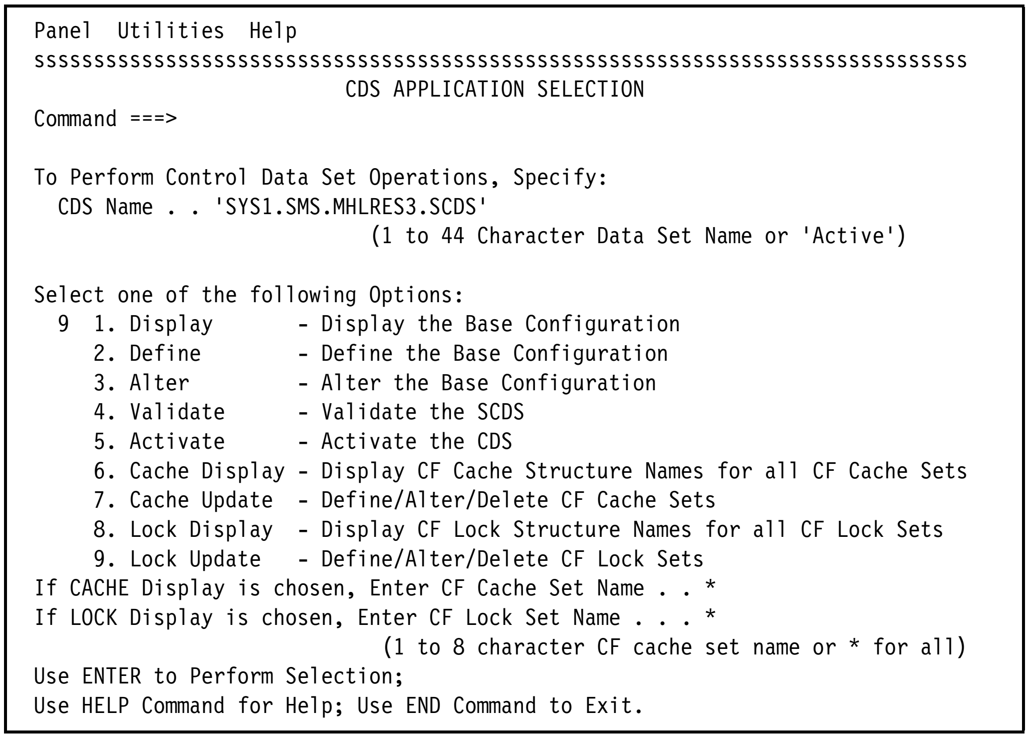
Defining the secondary lock structures to SMS

Like the RLS cache structures, SMSVSAM uses the SMS storage class construct to determine which CF lock structures can be assigned to a VSAM data set. To update your SMS configuration to support these structures, perform these steps:

1. Define the new lock structures in the SMS Control Data Set. Specify the name of a lock set and one lock structure per lock set:

   a. In the ISMF Primary Option menu, select option 8 (Control Data Set).

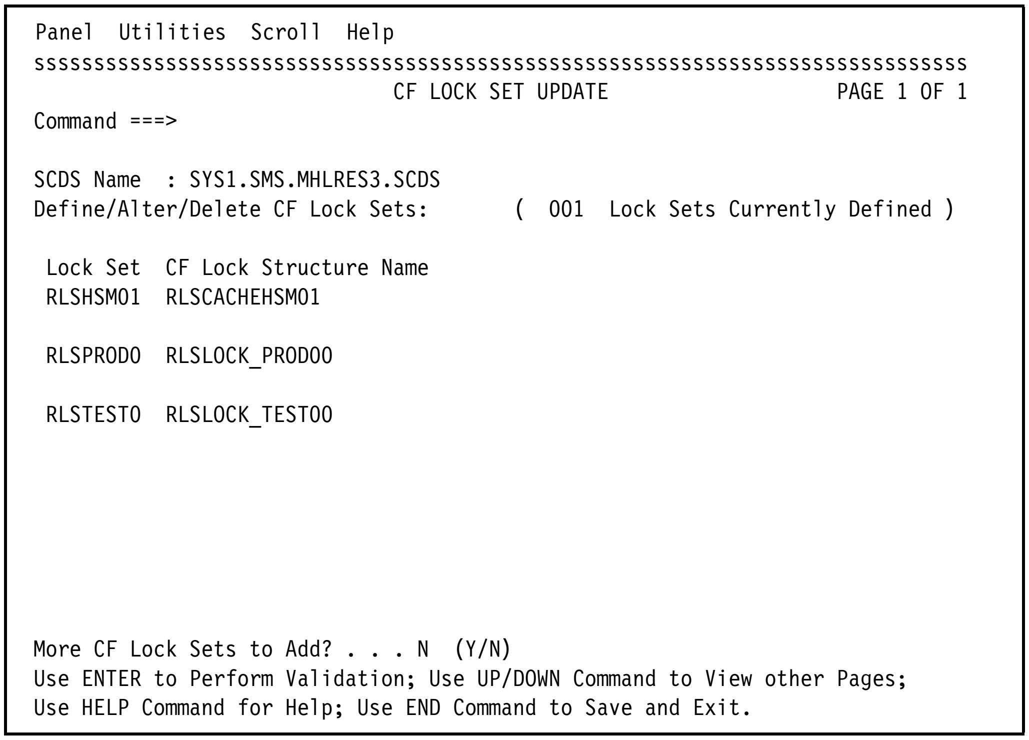
   b. In the CDS Application Selection menu, select option 9 (Lock Update) as shown in Figure 5-12.



```
Panel  Utilities  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                          CDS APPLICATION SELECTION
Command ===>

To Perform Control Data Set Operations, Specify:
  CDS Name . . 'SYS1.SMS.MHLRES3.SCDS'
                            (1 to 44 Character Data Set Name or 'Active')

Select one of the following Options:
  9  1. Display       - Display the Base Configuration
     2. Define        - Define the Base Configuration
     3. Alter         - Alter the Base Configuration
     4. Validate      - Validate the SCDS
     5. Activate      - Activate the CDS
     6. Cache Display - Display CF Cache Structure Names for all CF Cache Sets
     7. Cache Update  - Define/Alter/Delete CF Cache Sets
     8. Lock Display  - Display CF Lock Structure Names for all CF Lock Sets
     9. Lock Update   - Define/Alter/Delete CF Lock Sets
If CACHE Display is chosen, Enter CF Cache Set Name . . *
If LOCK Display is chosen, Enter CF Lock Set Name . . . *
                             (1 to 8 character CF cache set name or * for all)
Use ENTER to Perform Selection;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-12 Selecting option 9 (Lock Update)

c. Specify the name of the lock sets and assign one CF lock structure to one lock set (Figure 5-13). You can assign the same CF lock structure for several lock sets.

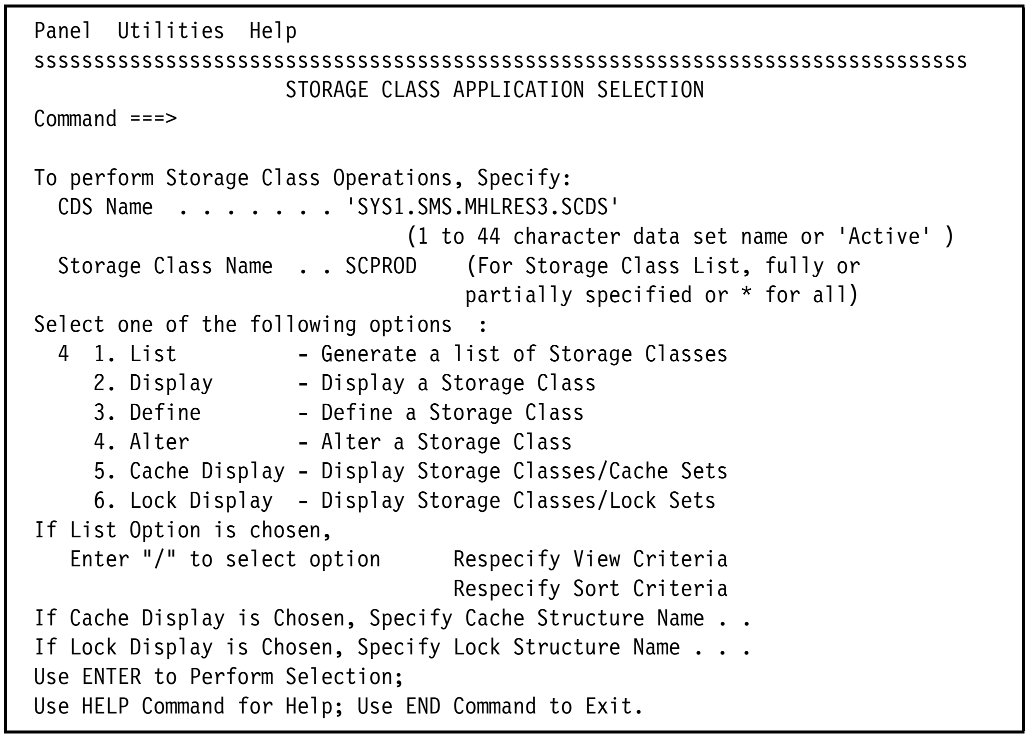


```
Panel  Utilities  Scroll  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                              CF LOCK SET UPDATE                   PAGE 1 OF 1
Command ===>

SCDS Name  : SYS1.SMS.MHLRES3.SCDS
Define/Alter/Delete CF Lock Sets:       (  001  Lock Sets Currently Defined )

 Lock Set  CF Lock Structure Name
 RLSHSM01  RLSCACHEHSM01

 RLSPROD0  RLSLOCK_PROD00

 RLSTEST0  RLSLOCK_TEST00








More CF Lock Sets to Add? . . . N  (Y/N)
Use ENTER to Perform Validation; Use UP/DOWN Command to View other Pages;
Use HELP Command for Help; Use END Command to Save and Exit.
```

Figure 5-13   Define Lock Sets and assign lock structures to them

2. Assign lock sets to Storage Classes:

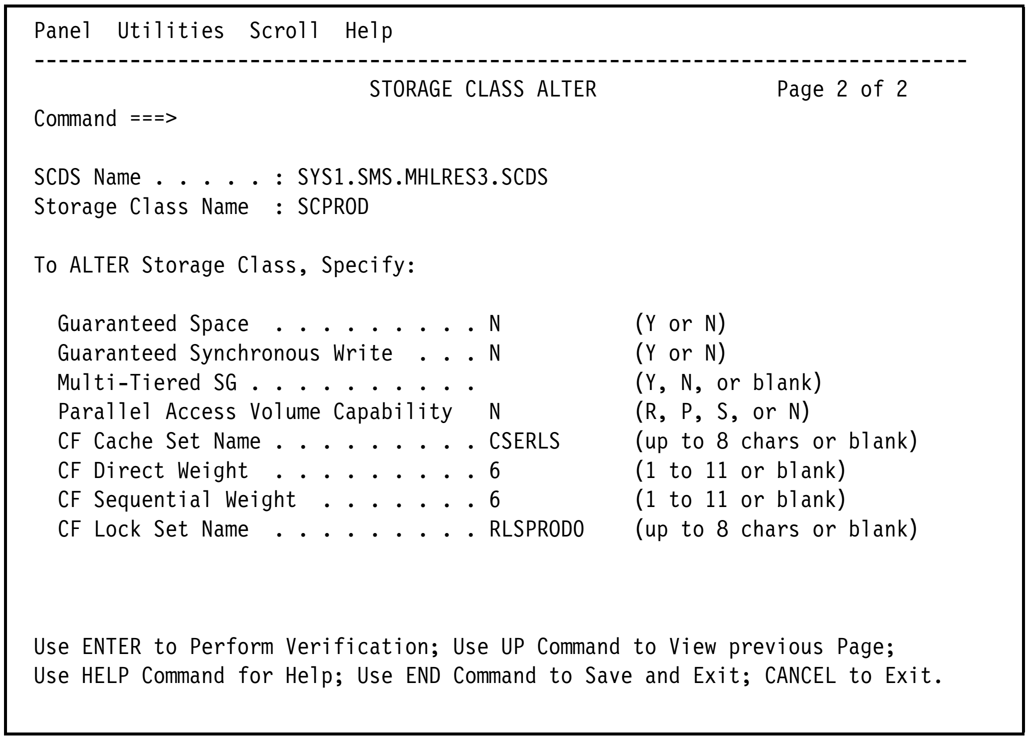
   a. Create or update your existing Storage Classes constructs. In this example, update a storage class called SCPROD (Figure 5-14).



```
Panel  Utilities  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                     STORAGE CLASS APPLICATION SELECTION
Command ===>

To perform Storage Class Operations, Specify:
  CDS Name  . . . . . . . 'SYS1.SMS.MHLRES3.SCDS'
                              (1 to 44 character data set name or 'Active' )
  Storage Class Name  . . SCPROD    (For Storage Class List, fully or
                                   partially specified or * for all)
Select one of the following options  :
  4  1. List         - Generate a list of Storage Classes
     2. Display      - Display a Storage Class
     3. Define       - Define a Storage Class
     4. Alter        - Alter a Storage Class
     5. Cache Display - Display Storage Classes/Cache Sets
     6. Lock Display  - Display Storage Classes/Lock Sets
If List Option is chosen,
   Enter "/" to select option      Respecify View Criteria
                                   Respecify Sort Criteria
If Cache Display is Chosen, Specify Cache Structure Name . .
If Lock Display is Chosen, Specify Lock Structure Name . . .
Use ENTER to Perform Selection;
Use HELP Command for Help; Use END Command to Exit.
```

Figure 5-14   Alter Storage Class

b. Specify the CF Lock Set Name as shown in Figure 5-15.



```
Panel  Utilities  Scroll  Help
------------------------------------------------------------------------------
                            STORAGE CLASS ALTER               Page 2 of 2
Command ===>

SCDS Name . . . . . : SYS1.SMS.MHLRES3.SCDS
Storage Class Name  : SCPROD

To ALTER Storage Class, Specify:

  Guaranteed Space  . . . . . . . . . N           (Y or N)
  Guaranteed Synchronous Write  . . . N           (Y or N)
  Multi-Tiered SG . . . . . . . . . .             (Y, N, or blank)
  Parallel Access Volume Capability   N           (R, P, S, or N)
  CF Cache Set Name . . . . . . . . . CSERLS      (up to 8 chars or blank)
  CF Direct Weight  . . . . . . . . . 6           (1 to 11 or blank)
  CF Sequential Weight  . . . . . . . 6           (1 to 11 or blank)
  CF Lock Set Name  . . . . . . . . . RLSPRODO    (up to 8 chars or blank)



Use ENTER to Perform Verification; Use UP Command to View previous Page;
Use HELP Command for Help; Use END Command to Save and Exit; CANCEL to Exit.
```

Figure 5-15 Specify the CF Lock Set Name

3. Update and translate your ACS routines if you are creating new constructs.
4. Validate the SCDS and activate it.

If your VSAM data set is assigned to a storage class that does not have any lock set specification, RLS processing uses IGWLOCK00 to control its locking.

DFSMS allows you to define up 256 lock sets and lock structure names. However, the maximum number of lock structures that can be connected is 10 - 14. Specify the number in the MAXCAD value of the IEASYSxx parmlib member.

Setting up VSAM striping (optional)

Beginning at z/OS 1.12, VSAM data set stripe is also supported for data sets in RLS mode.

A VSAM data set is striped when it is defined with a stripe count greater than one. Striping a VSAM data set spreads all the CIs across multiple devices. This format allows a single application request for records on multiple CIs to be satisfied by concurrent I/O requests to multiple volumes. This results in improved sequential data access performance by achieving data transfer into the application at a rate greater than a single I/O path.

VSAM derives the stripe count the same way as it always has for SAM stripe data sets. It requires the specification of *Sustained Data Rate* (SDR), and is affected by the specification of Guaranteed Space in the associated Storage Class.

If SDR is blank or 0, SMS assumes a stripe count of 1, so it is allocated as not striped.

If Guaranteed Space is (Y), and SDR is 1 or greater, the data set is striped in as many volumes as the volcount or the number of volsers specified in the allocation request, whichever one is greater.

If Guaranteed Space is (N), the Sustained Data Rate determines the stripe count. The design of SMS assumes a sustained data rate of 4 MBps for a device that emulates a 3390. For example, if you want to have the sustained data rate of two 3390 disks in parallel, specify 8 in the Sustained Data Rate field of the Storage Class construct.

Data Class attributes for RLS

The following SMS Data Class attributes are applicable to RLS:

- RLS CF Cache Value

  Specifies the amount of data that RLS places in the CF cache structures. It is honored only when RLS_MaxCfFeatureLevel(A) is active in the Sysplex.

  You can specify one of the following values:

ALL	VSAM data and index components are cached in the CF. This is the default value.
NONE	Only the index is cached in the CF.
UPDATESONLY	Only write requests are cached in the CF.

- RLS Above the 2 GB Bar

  Specifies whether the SMSVSAM address space can take advantage of 64-bit addressable virtual storage for VSAM RLS buffering.

  You can specify one of the following values:

Y	Indicates that SMSVSAM uses buffers above the 2 GB bar. Specify this value if your applications references more than 1.7 GB of RLS data per hour.
N	Indicates that the SMSVSAM uses buffers below the 2 GB bar. This is the default value.

- LOG

  Specifies whether the VSAM data set is recoverable or not.

NONE	Specifies that the data set does not have an external backout or forward recovery capability, so it is considered not recoverable.
UNDO	Specifies that the changes applied to this data set can be backed out by using an external log, so it is considered recoverable.
ALL	Specifies that this data set can be backward or forward recovered by using an external log, so it is recoverable.
blank	The data set is not recoverable. This is the default value. This data set cannot be accessed in RLS mode.

- Logstream ID

  Identifies the CICS forward recovery log stream.

5.4.6 Modifying the PARMLIB IGDSMSxx Member

Specify options for SYSVSAM in the IGDSMSxx member.

Tip: There is one SMSVSAM address space in each image of z/OS.

Example 5-4 shows the RLS parameters in bold.

Example 5-4 IGDSMSxx parmlib member

```
SMS ACDS(SYS1.SMS.ACDS)
    COMMDS(SYS1.SMS.COMMDS)
    INTERVAL(15)
    DINTERVAL(150)
    CA_RECLAIM(DATACLAS)
    DEADLOCK_DETECTION(15,4)
    DSSTIMEOUT(300)
    SMF_TIME(YES)
    CF_TIME(1800)
    RLSINIT(YES)
    RLS_MAXCFFEATURELEVEL(A)
    RLS_MAX_POOL_SIZE(100)
    REVERIFY(NO)
    ACSDEFAULTS(NO)
    PDSESHARING(EXTENDED)
    TRACE(ON)
    SIZE(128K)
    TYPE(ALL)
    JOBNAME(*)
    ASID(*)
    SELECT(ALL)
```

RLSINIT

To use VSAM RLS, the SMSVSAM address space must be up and running. If you specify RLSINIT=YES, the SMSVSAM address space starts as part of your system initialization at IPL. If you specify NO, you must start it later by using the VARY SMSVSAM,ACTIVE command. The default is NO.

CF_TIME

Here you specify at what interval you want to collect SMF type 42 records, which contain information and statistics on CF cache and lock structures. The default is 3600 (one hour). For more information, see "SMF record 42" on page 298.

DEADLOCK_DETECTION

This parameter specifies how frequently the dead lock detection routine is run to check for local and global deadlocks. The local deadlock detection interval default is 15 seconds.

It also specifies the number of local deadlock cycles that must expire before global deadlock detection is run. This parameter is a one to four digit value in the range 1-9999. The default is four cycles.

DSSTIMEOUT

Specifies the number of seconds that DFSMSdss waits during backup processing for quiesce data set requests to complete. Specify a value from zero to 65536 seconds. If you specify a value from 1 - 299 seconds, the system uses a value of 300 seconds (which equals 5 minutes).

The value specified in the DSSTIMEOUT parameter value is activated when the first instance of the SMSVSAM address becomes active in the sysplex. All subsequent SMSVSAM instances use the same value.

RLS_MaxCfFeatureLevel

Specifies the method VSAM RLS caching uses to determine the size of the data that is placed in the CF cache structure.

If you specify A, SMSVSAM uses the parameter RLS CF Cache Value specification in the SMS data class to decide what to keep in the CF RLS cache structures. See Example 5-4 on page 234.

If you specify A, you can also specify Mx values in the form of AMx to customize IGW500I messages for open VSAM data sets as follows:

M0	IGW500I for all open data sets are suppressed.
M1	IGW500I for all open data sets are displayed.
M2	IGW500I is issued only for the first open data set in the sysplex.
M3	IGW500I for all open data sets are sent to hardcopy.
M4	IGW500I for the first open data set in the sysplex is sent to hardcopy.

If you do not specify a value or specify Z, only VSAM RLS data that have a Control Interval (CI) value of 4K or less are placed in the CF cache structure. The default is Z.

RLS_MAX_POOL_SIZE

This is the maximum buffer pool size SMSVSAM can allocate in the SMSVSAM Data Space, in MB. This is the 31-bit local buffer pool. The default is 100 MB.Generally, do not specify a value greater than 1500.

RlsAboveTheBarMaxPoolSize

Specifies the maximum amount of virtual storage above the 2-gigabyte bar that VSAM RLS can use for RLS buffering in a system in the sysplex. You specify it as follows:

```
[RlsAboveTheBarMaxPoolSize{( sysname, maxrls; ...)|(ALL,maxrls}]
```

You can specify a different above the bar value for each system in the sysplex, or specify the same value for all the systems in the sysplex.

If you do not specify this keyword, SMSVSAM uses only the 31-bit local buffer pool in the SMSVSAM Data Space.

RlsFixedPoolSize

Specifies the amount of real storage (both above and below the 2 GB bar) to be dedicated to VSAM RLS buffering. You specify it as follows:

```
[RlsFixedPoolSize{( sysname, maxrls; ...)|(ALL,maxrls}]
```

You can specify a different value for each system in the sysplex, or use the same value for all systems in the sysplex. The minimum value is 0, which is the default. The maximum value is 80% of the total unpinned available real storage. SMSVSAM initialization issues a message if it reaches this limit.

Buffers are fixed on a first come, first serve basis. If the first data set opened and accessed has a 4K CISIZE, the fixed buffers are 4K in size for the life of that SMSVSAM instance.

SMF_TIME

This keyword specifies whether DFSMS is to use SMF timing. That is, whether SMF type 42 records are to be created at the expiration of the SMF interval period, which is synchronized with SMF and RMF data intervals.

5.4.7 Security definitions

This section provides security definitions for access to the SHCDS data set and access to use the VARY SHCDS command.

Access to SHCDS

Provide update access for SMSVSAM to the SCHCDS data set. Authorize SMSVSAM to update SYS1.DFPSHCDS.* data sets. If you protect SYS1.* data sets, be sure that the user ID associated with SMSVSAM is able to access SYS1.DFPSHCDS.* for update.

You can issue the RACF command RLIST STARTED SMSVSAM,ALL to obtain the user ID associated with SMSVSAM.

Use the following RACF command to give SMSVSAM access to SHCDS:

```
PERMIT SYS1.DFPSHCDS.* ACCESS(UPDATE) ID(smsvsam_userid)
```

Figure 5-16 shows the error messages that are displayed because of SHCDS access failure.

```
 ICH408I JOB(SMSVSAM ) STEP(SMSVSAM ) 744
    SYS1.DFPSHCDS.WTSCPLX2.VSBOX52 CL(DATASET ) VOL(SBOX52)
    INSUFFICIENT ACCESS AUTHORITY
    FROM SYS1.DFPSHCDS.* (G)
    ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
  IEF196I IEC161I 040(056,006,IGGOCLFT)-002,IEESYSAS,SMSVSAM,SYS00
  IEC161I 040(056,006,IGGOCLFT)-002,IEESYSAS,SMSVSAM,SYS00001,,,
  IEF196I IEC161I SYS1.DFPSHCDS.WTSCPLX2.VSBOX52
  IEC161I SYS1.DFPSHCDS.WTSCPLX2.VSBOX52
IEA794I SVC DUMP HAS CAPTURED: 760
  DUMPID=001 REQUESTED BY JOB (SMSVSAM )
  DUMP TITLE=COMPID=DF122,CSECT=IGWXSI20+0490,DATE=04/16/00,MAINT
           ID= NONE   ,ABND=0F4,RC=00000024,RSN=67510404
  RSN67510404 67510404
```

Figure 5-16 SMSVSAM error because of SHCDS access failure

Attention: If you do not provide the necessary RACF access authority to SMSVSAM, it fails to initialize.

Access to use VARY SHCDS command

To use the SHCDS command, you must have access to the facility class STGADMIN.IGWSHCDS.*. Use the following RACF command to provide this access.

```
PERMIT STGADMIN.IGWSHCDS.* CLASS(FACILITY) ID(youruserid)
```

Authority to use the IDCAMS SHCDS command

To use the access method services SHCDS command, you must be authorized to the facility class STGADMIN.IGWSHCDS.REPAIR. The SHCDS command is used to list SMSVSAM recovery that is associated with subsystems and spheres, and to control that recovery.

Also, if you intend to use the SHCDS command under TSO, you must add SHCDS in the AUTHCMD NAMES list in your IKJTSOxx parmlib member.

5.4.8 Using a VSAM data set in RLS mode

When all the previous actions are implemented, the system is ready for VSAM RLS processing.

To enable a VSAM data set for RLS, perform the following tasks:

1. Specify the LOG parameter (NONE, UNDO, or ALL) on the DEFINE CLUSTER or the ALTER CLUSTER command for the data set. Or you can select one of the data classes that contains such a parameter.
2. Set the option RLS in MACRF keyword in ACB. For more information, see "Access method Control Block (ACB)" on page 73. There is also the option RLSREAD keyword in ACB where you indicate the type of read integrity (NRI or CR). Another way is using the keyword RLS= (NRI / CR) in the DD card that defines the data set. For CICS to access a VSAM data set in RLS mode, specify RLSACCESS=YES in the file resource definition.

Tip: Your RLS specification about read integrity on the ACB (RLSREAD) overrides the RLS specification on your JCL, if you specify in both places.

Figure 5-17 shows the messages that are presented in the MVS console when a job that uses RLS starts.

```
IEF403I MHL#64C5 - STARTED - TIME=22.10.03 - ASID=0027 - SC64
ICH70001I MHLRES2  LAST ACCESS AT 21:53:13 ON FRIDAY, MARCH 14, 2003
$HASP373 MHL#63C5 STARTED - INIT 2    - CLASS A - SYS SC63
IGW500I DFSMS CACHE CHARACTERISTICS FOR 705
VSAM COMPONENT NAME: MHLRES2.VSAM.RLSEXT.INDEX
DFSMS CF CACHE STRUCTURE NAME: RLS_CACHE
CI SIZE: 1536
CF CACHING SIZE: 2048
DFSMS DATACLASS NAME: RMMCDS
DFSMS RLSCFCACHE DATACLASS KEYWORD VALUE: ALL
IGW500I DFSMS CACHE CHARACTERISTICS FOR 706
VSAM COMPONENT NAME: MHLRES2.VSAM.RLSEXT.DATA
DFSMS CF CACHE STRUCTURE NAME: RLS_CACHE
CI SIZE: 4096
CF CACHING SIZE: 4096
DFSMS DATACLASS NAME: RMMCDS
DFSMS RLSCFCACHE DATACLASS KEYWORD VALUE: ALL
```

Figure 5-17 Typical RLS messages as seen in a syslog

5.4.9 RLS Recoverable data sets

A recoverable data set has an associated log with it. This log is processed for commit and backout of transactions, and also for forward recovery. This attribute is set in the catalog entry by the DEFINE/ ALTER LOG keyword.

The LOG keyword has the following options: NONE, UNDO, or ALL (UNDO plus REDO), with the following properties:

NONE	This setting means that the data set is non-recoverable, implying no logging. When this option is used, backout and forward recovery logging are not run. If either software or hardware damages the data set, you must restore a backup copy to recover it.

The other two options (UNDO or ALL) imply that the accessing application uses a commit protocol that allows a recovery function through a log, like the CICS logs.

UNDO	This setting means that transactional recovery is required. To support this option, the accessing application must support a two-phase commit and backout protocol. During the life of a transaction, changes are protected by record level locks that prevent the changes from being seen by other applications that also support locking.
REDO	This setting means a forward recovery capability in the case of a lost or damaged data set. If you do not have an application that can run forward recovery of the VSAM data set, such as CICS/VR, there is no point in specifying LOG(ALL). It generates the processor load of creating the redo images of the records, but cannot do anything with that information.

5.5 RLS problem determination and recovery

This section addresses common problems with RLS and suggests remedies. There are a number of informational APARs in IBMLINK that describe common problems in VSAM and what you can do resolve them. These APARs also display what documents you need to collect before you report the problem to IBM. Review APAR II12927 and II14597 as a starting point.

5.5.1 Hang or Wait condition in RLS

A hang or wait condition in RLS may be related to these activities:

- Accessing VSAM data sets in RLS mode
- Starting/stopping of RLS client spaces (CICS regions for example)
- Starting/stopping SMSVSAM instances
- SMSVSAM commands not responding

The following commands can help detect and or pinpoint to hang or wait condition:

- D GRS,C

  Provides an alphabetized list of all visible ENQ resources that are in contention. Each resource is reported with the owners and waiters of the resource.

  Figure 5-18 shows an example where job MHLRES3 (a TSO user ID) owns the MHLRES.VSAM.DATA data set. Job MHLRES3L wants the same resource, but it wants it as an exclusive resource. MHLRES3L waits until MHLRES3 finishes or releases the MHLRES.VSAM.DATA data set.

```
ISG343I 16.38.20 GRS STATUS 512
S=SYSTEMS SYSDSN   MHLRES3.VSAM.DATA
SYSNAME       JOBNAME         ASID     TCBADDR   EXC/SHR    STATUS
SC63      MHLRES3           0095       007FF890   SHARE      OWN
SC63      MHLRES3L          0020       007E5E88 EXCLUSIVE    WAIT
NO REQUESTS PENDING FOR ISGLOCK STRUCTURE
NO LATCH CONTENTION EXISTS
```

Figure 5-18 D GRS,C sample

- D SMS,SMSVSAM,DIAG(CONTENTION):

The GRS latch manager is a service that authorized programs can use to serialize resources within an address space or, by using cross memory capability, within a single MVS system. You can use this command to verify latch contentions when your RLS environment is slower than expected or you suspect a hang condition. For more information about the command output, see 5.5.10, “Latch contention” on page 247.

- D XCF,STR,STRNM=*structurename*

  You can use this command to find the status and sizes of the lock and cache structures that RLS uses. For more information, see “D XCF,STR,STRNAME=structurename” on page 254.

- D SMS,SMSVSAM,QUIESCE

  This command displays the status of all active VSAM record-level sharing (VSAM RLS) data set quiesce events on the system.

- D SMS,SMSVSAM,ALL

  This command displays the status of the SMSVSAM address space on this system, or all the SMSVSAM address spaces and lock table connection status. For more information, see “D SMS, SMSVSAM [,ALL]” on page 250.

Analyze the output displays and look for these indicators:

- Any potential resource deadlocks
- Incomplete structure rebuilds (or rebuild stops)
- SMSVSAM address spaces starting/stopping
- Incomplete quiesce events
- Message IXL041E, which indicates which system is not responding to rebuilds or disconnect events for IGWLOCK00

Console dumps

When a hang or wait condition occurs, console memory dumps and a combined operlog are required to analyze the situation. You should take these logs before any recovery actions (for example cancel/force commands). SMSVSAM is a sysplex-wide function, so SYSPLEX-wide memory dumps, logrec, and a combined operlog are necessary to completely analyze the problem.

To speed up sysplex wide dumping, create the IEADMCxx parmlib member as shown in Example 5-5.

Example 5-5 IEADMCxx example

```
JOBNAME=(SMSVSAM,XCFAS,CATALOG,clientjobname,*MASTER*),
DSPNAME=('SMSVSAM'.*,'XCFAS'.*),
SDATA=(COUPLE,CSA,GRSQ,LPA,LSQA,PSA,RGN,SQA,SUM,SWA,TRT,XESDATA),
REMOTE=(ACTION=SVCD,JOBLIST=(SMSVSAM,XCFAS,CATALOG,clientjobname,*MASTER*),
DSPNAME,SDATA),END
```

For *clientjobname*, you can specify a job region that is being affected by hang or wait conditions.

As soon as the problem is detected, have the console operator issue the following command:

```
DUMP PARMLIB=xx
```

Where xx is the suffix of the IEADMCxx that you created.

After you collecting these memory dumps, you might need to recycle one or more SMSVSAM address spaces by using these commands:

1. V SMS,SMSVSAM,TERMINATESERVER
2. V SMS,SMSVSAM,ACTIVE

If the TERMINATESERVER command does not work, you can cancel and automatically restart SMSVSAM with the following command:

```
FORCE SMSVSAM,ARM
```

If the TERMINATESERVER command does not work, you can cancel and not automatically restart SMSVSAM with the following commands:

1. SETSMS RLSINIT(NO)
2. FORCE SMSVSAM,ARM

If these commands do not terminate SMSVSAM, you can use the FORCE command as a last resort:

```
FORCE SMSVSAM
```

Abend 30D

An Abend 30D occurs after an MVS FORCE command against an address space that is hung in EOM. This message is indication of a hang situation. SYSPLEX-wide console memory dumps and combined operlogs might be required to fully diagnose the problem. Provide the ABEND 30D memory dump, any ABEND0F4 memory dumps, console memory dumps, logrec, and combined operlog to IBM service.

Remember: All address spaces (RLS client or SMSVSAM) should terminate following an MVS cancel command (or for RLS: FORCE SMSVSAM,ARM).

5.5.2 Internal logic errors in RLS

When RLS detects an internal logic error, an ABEND 0F4 with a unique return code and reason code is issued by SMSVSAM. In addition, an SVC memory dump is taken. Other abends (for example a 0C4 abend) might also occur. The RLS function that triggered the abend might fail, or might continue depending on the severity of the error. For some errors, SMSVSAM can automatically recycle.

Collect the memory dumps, combined sysplex-wide SYSLOG, and LOGREC data, and report the problem to IBM.

5.5.3 Data integrity: VSAM Data Trap

You can activate the VSAM RLS Index/Data Trap to investigate occurrences of broken VSAM KSDS data sets. Doing so triggers an 0F4 memory dump when the breakage is about to occur, and fails the write request and attempts to avoid any damage to the data set.

VSAM Data Trap has the following commands available:

- V SMS,MONDS(IGWVSAM.BASE.DATA.TRAP),ON(*data component1,....data component8*)

  This command makes the VSAM data components that are named in the command eligible for data trap. At least one data component name must be provided. Up to eight data components can be provided. The data components become enabled when they are opened. To add more data components to this eligibility list, issue the command with new data component names. The data set that is specified in the command becomes enabled to use data trap. Any data set that was previously enabled is enabled again and remains eligible until the data set is closed.

V SMS,MONDS(IGWVSAM.BASE.DATA.TRAP),ON(*) activates the trap for all VSAM KSDS data sets.

Figure 5-19 shows activating the VSAM Data Trap for the data component of the MHLRES3.VSAM.KSDS data set.

```
-V SMS,MONDS(IGWVSAM.BASE.DATA.TRAP),ON(mhlres3.vsam.ksds.data)
 IGW682I BASE-VSAM DATA TRAP IS ENABLED --
   THE FOLLOWING DATA SETS HAVE BEEN SET TO BE TRAPPED:
   1: MHLRES3.VSAM.KSDS.DATA
   2: ............................................
   3: ............................................
   4: ............................................
   5: ............................................
   6: ............................................
   7: ............................................
   8: ............................................
   WHEN THE TRAP HITS, A CONSOLE MESSAGE MIGHT BE ISSUED
   WITH THE FAILING DATA SET NAME AND REASON,
   AND A DUMP WILL BE GENERATED.
```

Figure 5-19 Activating VSAM Data Trap

- D SMS,MONDS(IGWVSAM.BASE.VSAM.DEBUG.FEATURES)

  Use this command to display the status of VSAM diagnostic features to determine whether they are active for the system. In addition, if data trap is enabled, it displays the data component names that are found in the current data trap list. This list was created when the user last enabled VSAM Data Trap by using the **Vary SMS** command. Any data set that was previously enabled for data trap remains eligible until the data set is closed. See Figure 5-20.

```
-D SMS,MONDS(IGWVSAM.BASE.VSAM.DEBUG.FEATURES)
 IGW675I BASE-VSAM DEBUG INDICATORS:
   VSAM FOOTSTEP:          ON
   INDEX TRAP:             ON
   DATA TRAP:              ON
   THE FOLLOWING DATASETS HAVE BEEN SET TO BE TRAPPED:
   1: MHLRES3.VSAM.KSDS.DATA
   2: ............................................
   3: ............................................
   4: ............................................
   5: ............................................
   6: ............................................
   7: ............................................
   8: ............................................
```

Figure 5-20 VSAM Data Trap status

5.5.4 VSAM_DATA_TRAP Health Check

The check notifies the user if the VSAM Data Trap has been enabled longer than the expected time interval. Generally, run with VSAM Data Trap disabled because it causes performance degradation to the system because of additional validation of data records before they are written to DASD. Enable VSAM Data Trap only when a problem is encountered related to a data CI and more diagnostic data is needed for problem determination. Disable VSAM Data Trap as soon as the data is collected.

Use the keywords in Figure 5-21 to override CHECK values on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command. This statement can be copied and modified to override the check defaults:

```
UPDATE
CHECK(IBMVSAM,VSAM_DATA_TRAP)
ACTIVE
SEVERITY(MED) INTERVAL(00:30) DATE(20090224)
PARM('TimeM(30)')
REASON('IBM recommends running with the VSAM Data Trap disabled.')
```

Figure 5-21 VSAM Data Trap Health Check

5.5.5 Data integrity: VSAM index trap

For VSAM RLS, an index trap checks each index record before writing it. The trap detects the following index corruptions:

- High-used greater than high-allocated
- Duplicate or invalid index pointer
- Out-of-sequence index record
- Invalid section entry
- Invalid key length.

If the index record is corrupted, an ABEND 0F4 with a system memory dump is generated, but the SMSVSAM Address Space is not recycled. Contact IBM service if you see a memory dump with the following title:

```
DUMP TITLE=COMPID=DF122,CSECT=IDAVRBF4+xxxx,DATE=
ID=xxxxxxx ,ABND=0F4,RC=00001008,RSN=61609Dxx
```

This index trap is normally inactive. The following console command dynamically activates the index trap throughout the sysplex:

```
V SMS,MONDS(IGWVSAM.INDEX.TRAP),ON
```

You can issue the following command to deactivate the index trap:

```
V SMS,MONDS(IGWVSAM.INDEX.TRAP),OFF
```

To query the status of the index trap, issue the following command:

```
D SMS,MONDS(SPECIAL.FUNCTION.STATUS)
```

5.5.6 Errors during RLS record management processing

When cancels or internal errors occur in the RLS record management processing, the messages IGW400I or IGW405I are displayed on the console:

- The IGW400I message indicates an abnormal interrupt to an RLS record management request, and is used for debugging purposes by IBM service.
- The IGW405I message indicates that an abnormal interrupt occurred in the RLS record management CI/CA split path, which indicates a potential data integrity problem. Access to the data set is shut down on this system. The data set must be fully closed and reopened.

5.5.7 Transaction recovery

This section includes the following topics:

- Retained locks
- Lost locks

Retained locks

Retained locks occur when active locks are held by recoverable subsystems, such as a CICS region or a TVS instance, for recoverable data sets and one of the following occurs:

- Recoverable data set is closed without ending all active transactions
- SMSVSAM terminates with active locks
- System failure with active locks
- Backout fails (shunted) for one or more transactions that have active locks

When a data set has retained locks, record management requests for the same record locks by RLS applications fail with RC=8 RPLERRCD=24(x'18').

Recovery must be completed against the failed unit of recovery (URs) in order for the retained locks to be released. If the retained lock was caused by a CICS region that terminated, restart that CICS region so the transaction backout or commit can complete.

To determine which subsystem or data set have retained locks, use the IDCAMS SHCDS command, as follows:

1. SHCDS LISTSUBSYS(ALL)

   This command lists all subsystems (CICS, SMSVSAM or TVS instance), showing which one has retained locks held.

Figure 5-22 shows an example where subsystems KMKLK05D, KMKLK05F, and RETLK05A have failed, and have retained locks.

```
SHCDS LISTSUBSYS(ALL)
  ----- LISTING FROM SHCDS ----- IDCSH03

-----------------------------------------------------------------------------------------------
                                    RECOVERY            LOCKS         LOCKS          LOCKS
  SUBSYSTEM NAME   STATUS           NEEDED              HELD          WAITING        RETAINED
  --------------   --------------   --------------   -----------------------------------------
  SMSVSAM          BATCH --ACTIVE   NO                         0             0              0
      DATA SETS IN LOST LOCKS------------         0
      DATA SETS IN NON-RLS UPDATE STATE--         0
      TRANSACTION COUNT------------------         0
  KMKLK05D         ONLINE--FAILED   YES                        0             0              1
      DATA SETS IN LOST LOCKS------------         0
      DATA SETS IN NON-RLS UPDATE STATE--         0
      TRANSACTION COUNT------------------         1
  KMKLK05F         ONLINE--FAILED   YES                        0             0              1
      DATA SETS IN LOST LOCKS------------         0
      DATA SETS IN NON-RLS UPDATE STATE--         0
      TRANSACTION COUNT------------------         1
  RETLK05A         ONLINE--ACTIVE   YES                        0             0             15
      DATA SETS IN LOST LOCKS------------         1
      DATA SETS IN NON-RLS UPDATE STATE--         0
      TRANSACTION COUNT------------------         1
  IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure 5-22 SHCDS LISTSUBSYS example

2. SHCDS LISTSUBSYSDS(*subsystemname*)

   After you determine which subsystem has retained locks held, use the SHCDS LISTSUBSYSDS command to obtain which data sets have locks held for that subsystem. Figure 5-23 shows the data sets that have retained locks by subsystem RETLK05A.

```
SHCDS LISTSUBSYSDS(RETLK05A)
  ----- LISTING FROM SHCDS ----- IDCSH04
  -------------------------------------------------------------------------------
  SUBSYSTEM NAME---- RETLK05A      SUBSYSTEM STATUS----ONLINE--ACTIVE
                                           LOCKS                NON-RLS     PERMIT
  DATA SET NAME /     RETAINED    LOST     NOT       RECOVERY   UPDATE      FIRST TIME
  CACHE  STRUCTURE    LOCKS       LOCKS    BOUND     REQUIRED   PERMITTED   SWITCH
  ----------------    --------    -----    -------   --------   ---------   -----
  SYSPLEX.KSDS.PERMIT.CLUS2
  CACHE01             YES         NO       NO        NO         NO          NO
  SYSPLEX.KSDS.RETAINED.CLUS1
  CACHE01             YES         NO       NO        NO         NO          NO
  SYSPLEX.KSDS.SHARED.CLUS4
  CACHE01             YES         NO       NO        NO         NO          NO
  IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure 5-23 SHCDS LISTSUBSYSDS example

For more information about the SHCDS command and a description of its output, see *DFSMS Access Method Services for Catalogs*, SC26-7394.

Recovery must be completed against the failed unit of recovery (URs) in order for the retained locks to be released.

Lost locks

VSAM data sets can enter the lost locks" state when all of the following are true:

- Any SMSVSAM address space is terminated with either active opens or retained locks for recoverable files
- All other SMSVSAM address spaces are terminated in the sysplex
- An RLS lock structure that contained those locks is deleted

When a data set is in lost locks state, opens by new subsystems fail with the message `IEC161I return code 0248 reason code 0580`.

To clear the lost lock state, the subsystems that were accessing the recoverable files at the time of the failure must be restarted. They must then run recovery for the data sets in lost locks state. To determine which subsystems or data sets are in lost locks, use the IDCAMS SHCDS command. For more information, see "Retained locks" on page 243.

If a TVS instance has data sets in lost locks, the SMSVSAM address space that is associated with that TVS instance must be restarted. Issue the following command to determine to which system this TVS instance is registered:

```
D SMS,TRANVSAM,ALL
```

If the system associated with the TVS instance in lost locks cannot be restarted, another system in the sysplex can perform recovery on behalf of the failed instance. The system that you chose to run the recovery (peer recovery) must have access to these items:

- TVS log data sets
- VSAM RLS data sets
- Catalogs that are required to do the recovery

To start peer recovery, issue the following command from the system you want to perform the recovery:

```
V SMS,TRANVSAM(tvsname),PEERRECOVERY,ACTIVE
```

5.5.8 Lock structure full

A lock structure full condition can occur when the number of locks (or record table entries) exceeds the available entries in the structure. This situation can occur if the lock structure size is two small to handle the normal RLS activity. More commonly, it occurs when an error in an RLS recoverable application occurs and obtains an abnormal number of locks.

To protect against a lock structure full condition, supply a FULLTHRESHOLD value in the CFRM policy for them. If the number of lock entries reaches the threshold value, the XCF component issues message IXC585E. In addition, SMSVSAM issues the following message when a lock structure is 80% full:

```
IGW326W *** Warning *** DFSMS SMSVSAM RECORD TABLE IN lockstructurename IS percent
% FULL.
```

TVS users can also specify the MAXLOCKS(*x,y*) parameter in the IGDSMSxx member in SYS1.PARMLIB. The MAXLOCKS parameter causes SMSVSAM to issue the following warning message when any TVS job is holding more locks than value *x*. It issues the message again when if obtains an additional *y* locks:

```
IGW10074I JOB jobname UNIT OF RECOVERY urid HAS REQUESTED nnn LOCKS  SYSTEM
MAXIMUM IS mmm
```

To check the available number of lock entries, entries in use, and the current lock structure size, issue the MVS command:

```
D XCF,STRUCTURE,STRNAME=strname
```

For an example, see “D XCF,STR,STRNAME=structurename” on page 254.

If there is available space between the current and max size of the lock structure, and more space is required, you can alter its size with this command:

```
SETXCF START,ALTER,STRNM=structurename,SIZE=nnn
```

A lock structure can go from the fullthreshhold to 100% full in a short time. Consider using the ALLOWAUTOALT option for the lock structures in the CFRM policy.

If an application error is the cause of the large volume of locks, cancel the application. The application can then attempt to back out the active URs. It might require more lock structure space to do so.

5.5.9 SHCDS recovery procedures

Always run at least two active and one spare SHCDS. If a permanent I/O error occurs for an active SHCDS, or if an SHCDS becomes inaccessible from one or more systems, it is automatically replaced by a spare SHCDS.

When a system is forced to run with only one SHCDS, it issues a message requesting that you add another active SHCDS and at least one spare SHCDS. If any system does not have access to an SHCDS, all opens for VSAM RLS processing are prevented on that system until an SHCDS becomes available.

If your SHCDS gets a permanent error, perform the steps in “Replacing an SHCDS”. Do *not* try to replace an SHCDS by deleting and redefining new SHCDSs, even if all SMSVSAM instances in the sysplex are down. SMSVSAM always remembers the last SHCDSs that it was using.

Replacing an SHCDS

Perform these steps to replace an SHCDS:

1. Allocate the new SHCDS. For more information, see “Defining Sharing Control Data Sets (SHCDS)” on page 220.
2. Add new SHCDSs to SMSVSAM:
   - For a new active SHCDS, issue the command:
     ```
     VARY SMS,SHCDS(SHCDS_name),NEW
     ```
   - For a new spare SHCDS, issue the command:
     ```
     VARY SMS,SHCDS(SHCDS_name),NEWSPARE
     ```
3. After you add the new SHCDSs, you can delete the old ones by issuing the command:
   ```
   V SMS,SHCDS(SHCDS_name),DELETE
   ```

FALLBACK

For some SHCDS errors, FALLBACK is the only way to correct the problem and to get the SMSVSAM to initialize successfully again. If you get repeated abends with the prefix “67” in the RSN code, you might have to rely on FALLBACK as the last resort. The following errors might require FALLBACK:

- Abend0F4 RC24 RSN675D0355
- Abend0F4 RC24 RSN67260989
- Abend0F4 RC24 RSN675F0398

Attention: If you have serious problems with SHCDS, an IPL will *not* fix the problem because SHCDS name is remembered from one IPL to another.

The FALLBACK procedure is documented in *z/OS V1R12.0 DFSMSdfp Storage Administration Reference*. In a nutshell, FALLBACK reformats the SHCDS and therefore clears all the errors.

FALLBACK involves the following steps:

1. Terminate all SMSVSAMs in the sysplex by issuing the following command:

   ```
   *ROUTE ALL VARY SMS,SMSVSAM,TERMINATESERVER
   ```

2. Reply “C” to any outstanding IGW418D message. If for any reason SMSVSAM does not terminate, try the following command:

   ```
   FORCE SMSVSAM
   ```

3. Complete the FALLBACK by issuing the command:

   ```
   VARY SMS,SMSVSAM,FALLBACK
   ```

4. Ensure that all SMSVSAM address spaces are disabled.
5. Respond to message IGW523 with this command:

   ```
   FALLBACKSMSVSAMYES
   ```

6. Reactivate SMSVSAM in one system by using the command:

   ```
   VARY SMS,SMSVSAM,ACTIVE
   ```

   On the first system, you must specify 2 ACTIVE and 1 SPARE SHCDS.
7. To add an active SHCDS, issue this command:

   ```
   VARY SMS,SHCDS(SHCDS_name),NEW
   ```

8. To add a spare SHCDS, issue this command:

   ```
   VARY SMS,SHCDS(SHCDS_name),NEWSPARE
   ```

9. Reactivate SMSVSAM in the other systems that access VSAM data sets in RLS mode in the sysplex.

5.5.10 Latch contention

There might be cases where your applications that access VSAM data set in RLS mode experience a severe performance slowdown, or even hang. When this situation happens, issue this MVS command to find information about latch contention:

```
DISPLAY SMS,SMSVSAM,DIAG(CONTENTION)
```

Figure 5-24 shows an example of no latch contention.

```
DISPLAY SMS,SMSVSAM,DIAG(CONTENTION)
IEE932I 630
IGW342I VSAM RLS DIAG STATUS (V.01)
NO CONTENTION BY REGISTERED RESOURCES EXISTS
```

Figure 5-24 No latch contention

You can use the output of this command to decide what to do to correct the latch contention:

- If the user holding a latch and causing the contention is a CICS transaction or a batch job on this system, you can cancel the transaction or job
- If the user holding the latch is waiting for an enqueued resource in another system, issue DISPLAY SMS,SMSVSAM,DIAG(CONTENTION) in the other system, to see whether there is a problem there
- If the user holding the latch is a task in the SMSVSAM address space, you might need to restart it by issuing the following command:

  ```
  FORCE SMSVSAM,ARM
  ```

Figure 5-25 shows an example of latch contention in the command output.

```
SYSTEM1          d sms,smsvsam,diag(contention)
SYSTEM1          IGW343I VSAM RLS DIAG STATUS (V.01)
|---RESOURCE----| |------ WAITER ------| |--HOLDER---| ELAPSED
TYPE       ID    JOB NAME ASID   TASK   ASID   TASK     TIME
-------- -------- -------- ---- -------- ---- -------- --------
LATCH    7BAD43B8 SMSVSAM  003A 008D5A48 003A 007F3000 00:22:09
LATCH    07F1B1D0 SMSVSAM  003A 007F3000 003A 008D5A48 00:22:09
LATCH    07F1B1D0 SMSVSAM  003A 008D64F8 003A 008D5A48 00:22:24
LATCH    07F1B1D0 SMSVSAM  003A 008D6CF0 003A 008D5A48 00:23:30
```

Figure 5-25 Contention on latch 07F1B1D0

The output of DISPLAY SMS,SMSVSAM,DIAG(CONTENTION), as shown in Figure 5-25, can be broken into four major fields:

RESOURCE	Shows the type (LATCH) and ID of the resource that has contention on it
WAITER	Shows the job name, asid, and task that is waiting for the latch
HOLDER	Shows which address space is holding the latch
ELAPSED TIME	Shows the amount of time that this task has waited for the latch

In Figure 5-25, task 008D5A48 in SMSVSAM address space has been waiting for more than 22 minutes for latch 7BAD43B8. Latch 7BAD43B8 has been held by task 007F3000, which is also in SMSVSAM:

```
LATCH    7BAD43B8 SMSVSAM  003A 008D5A48 003A 007F3000 00:22:09
```

However, task 007F3000 is waiting for latch 07F1B1D0, which is held by task 008D5A48:

```
LATCH    07F1B1D0 SMSVSAM  003A 007F3000 003A 008D5A48 00:22:09
```

This is a deadlock situation. The only way to get out of this deadlock is to stop or cancel one of latch holders. Because they are tasks in the SMSVSAM address space, try to solve this deadlock by terminating and restarting SMSVSAM by using the following command:

```
FORCE SMSVSAM,ARM
```

5.5.11 RLS rules

RLS enforces some rules when you have a data set open in RLS and non-RLS modes.

RLS OPEN rules

The RLS OPEN command has these restrictions concerning RLS:

- RLS OPEN for input/output fails if the data set is already opened for non-RLS output.
- RLS OPEN for input/output fails if the data set is already opened for non-RLS input, unless the data set is defined as SHAREOPTION(2,x). The non-RLS reader does not have read integrity.
- RLS OPEN for output of a recoverable data set by a batch client fails.
- RLS OPEN for a data set that is either quiesced or is quiescing fails (QUIESCE=YES in catalog).
- RLS OPEN for a VSAM data set that has not been assigned a CF cache by using the SMS STORCLAS construct fails.
- Empty KSDSs: RLS allows you to open an empty KSDS without first loading the data set. In other modes (NSR, RLS), this process is not possible.
- Positioning: RLS does not do implicit positioning to the beginning of the data set for SEQ processing. An explicit POINT is required.

CLOSE and DELETE rules

The CLOSE and DELETE commands have these restrictions concerning RLS:

- An RLS CLOSE is “not successful” if the SMSVSAM address space has recycled since the ACB was opened. One of the following messages are displayed: IEC251I 16-0608 or IEC251I 16-0609.
- A Catalog DELETE deletes all retained or lost locks if the owner is not an RLS subsystem. For more information about the definition of an RLS subsystem, see “RLS terminology” on page 216. The DELETE does not delete CICS log records because CICS is a subsystem.

5.6 Operational procedures for RLS

This section describes some operational procedures and commands you need to become familiar with.

5.6.1 Working with the SMSVSAM address space

The following are useful commands for checking the status of SMSVSAM, and activating and terminating by operator commands:

- D SMS, SMSVSAM [,ALL]
- VARY SMS,SMSVSAM, xxxxx
- D SMS,SMSVSAM,QUIESCE

D SMS, SMSVSAM [,ALL]

This command displays the status of the SMSVSAM server on this system, or all the SMSVSAM servers and primary and secondary lock table connection status.

Figure 5-26 shows the first part of the command. It shows that there are four systems in the sysplex that are able to use VSAM RLS. All of them show the status of SMSVSAM as *SmsVsamInitComplete*, which is their expected status.

```
-D SMS,SMSVSAM,ALL
 IEE932I 173
 IGW420I DISPLAY SMS,SMSVSAM,ALL
 DISPLAY SMS,SMSVSAM - SERVER STATUS
   SYSNAME:  SC65       AVAILABLE ASID: 000A STEP: SmsVsamInitComplete
   SYSNAME:  SC64       AVAILABLE ASID: 000A STEP: SmsVsamInitComplete
   SYSNAME:  SC63       AVAILABLE ASID: 000A STEP: SmsVsamInitComplete
   SYSNAME:  SC70       AVAILABLE ASID: 0066 STEP: SmsVsamInitComplete
   SYSNAME:  ........ ........... ASID: .... STEP: ...................
   SYSNAME:  ........ ........... ASID: .... STEP: ...................
   SYSNAME:  ........ ........... ASID: .... STEP: ...................
   SYSNAME:  ........ ........... ASID: .... STEP: ...................

 DISPLAY SMSVSAM     - JOB STATUS
   SUBSYSTEMS CONNECTED:       0 BATCH:       1
```

Figure 5-26 D SMS,SMSVSAM,ALL showing SMSVSAM servers active

Figure 5-27 shows another part of the output for D SMS,SMSVSAM,ALL. It shows the status of primary lock structure IGWLOCK00. In this example, it is connected to the four systems in the sysplex.

```
DISPLAY SMS,SMSVSAM - LOCK TABLE STATUS
    FOR STRUCTURE  IGWLOCK00
  CONNECT STATUS:
    SYSNAME:  SC65     ACTIVE           RSN: 02010407 RbldNotActive
    SYSNAME:  SC64     ACTIVE           RSN: 02010407 RbldNotActive
    SYSNAME:  SC63     ACTIVE           RSN: 02010407 RbldNotActive
    SYSNAME:  SC70     ACTIVE           RSN: 02010407 RbldNotActive
    SYSNAME:  ........ ................. RSN: ........ ................
    SYSNAME:  ........ ................. RSN: ........ ................
    SYSNAME:  ........ ................. RSN: ........ ................
    SYSNAME:  ........ ................. RSN: ........ ................

  COMPOSITE STATUS:
    ORIGINAL STRUCTURE: NOT VOLATILE       NOT FAILURE ISOLATED



  STRUCTURE STATUS:
    SYSNAME: SC65      Simplex
    SYSNAME: SC64      Simplex
    SYSNAME: SC63      Simplex
    SYSNAME: SC70      Simplex
    SYSNAME: ........  ...........
    SYSNAME: ........  ...........
    SYSNAME: ........  ...........
    SYSNAME: ........  ...........
```

Figure 5-27 IGWLOCK00 status in the D SMS,SMSVSAM,ALL output

Figure 5-28 shows another part of the D SMS,SMSVSAM,ALL command output that shows the status of secondary lock structure RLSLOCK_TEST00.

The structure RLSLOCK_TES00 has the following connect statuses:

- ACTIVE on systems SC63, SC65, and SC70
- FAILED PERSISTANT on system SC64

D SMS,SMSVSAM,ALL does not display the status of the secondary lock structures unless your applications have accessed VSAM data sets that are assigned to use these structures by using their storage classes. The secondary lock structures are allocated when they are first assigned to a VSAM data set.

In this example, batch jobs have been submitted in system SC63, SC65, and SC70 to access a VSAM data set in RLS mode. This data set was assigned to Storage Class SCTEST, which is assigned to lock set name RLSTEST0. Lock set RLSTEST0 is assigned to CF structure RLSLOCK_TEST00. XCF created RLSLOCK_TEST00 in CF2.

System SC64 did not access any VSAM data set that is assigned to RLSLOCK_TEST00 CF structure, so it is not connected to that structure.

```
DISPLAY SMS,SMSVSAM - LOCK TABLE STATUS
    FOR STRUCTURE  RLSLOCK_TEST00
  CONNECT STATUS:
    SYSNAME:  SC64     FAILED PERSISTANT RSN: ........ ................
    SYSNAME:  SC70     ACTIVE            RSN: 00000000 RbldNotActive
    SYSNAME:  SC65     ACTIVE            RSN: 00000000 RbldNotActive
    SYSNAME:  SC63     ACTIVE            RSN: 00000000 RbldNotActive
    SYSNAME:  ........ ................. RSN: ........ ................
    SYSNAME:  ........ ................. RSN: ........ ................
    SYSNAME:  ........ ................. RSN: ........ ................
    SYSNAME:  ........ ................. RSN: ........ ................

  COMPOSITE STATUS:
    ORIGINAL STRUCTURE: NOT VOLATILE       NOT FAILURE ISOLATED


  STRUCTURE STATUS:
    SYSNAME: SC64      NOT CONN    ......... ........
    SYSNAME: SC70      Simplex     ENABLE           0
    SYSNAME: SC65      Simplex     ENABLE           0
    SYSNAME: SC63      Simplex     ENABLE           0
    SYSNAME: ........  ........... ......... ........
    SYSNAME: ........  ........... ......... ........
    SYSNAME: ........  ........... ......... ........
    SYSNAME: ........  ........... ......... ........
```

Figure 5-28 Status of a secondary lock structure

Figure 5-28 indicates that System SC64 is not connected ("FAILED PERSISTANT" and "NOT CONN") to secondary lock structure RLSLOCK_TEST00. "FAILED PERSISTANT" might be replaced by more appropriate text in a future release.

Figure 5-29 shows another part of the D SMS,SMSVSAM,ALL output, showing information about the collecting of SMF type 42 records and its RLS-related subtypes. You can see that in this environment SC70 is the system that is writing SMF records to the SYS1.MANxx data sets.

```
DISPLAY SMS,SMSVSAM - SMF RECORD 42 STATUS
                    SMF_TIME  CF_TIME ----- SUB-TYPE  SUMMARY -----
                                             15  16  17  18  19
  SYSNAME: SC64      YES- 4    1800- 4     YES YES YES YES YES
  SYSNAME: SC65      YES- 2    1800- 2     YES YES YES YES YES
  SYSNAME: SC63      YES- 3    1800- 3     YES YES YES YES YES
  SYSNAME: SC70     *YES- 1    1800- 1     YES YES YES YES YES
  SYSNAME: ........ ....-..  ......-..     ... ... ... ... ...
  SYSNAME: ........ ....-..  ......-..     ... ... ... ... ...
  SYSNAME: ........ ....-..  ......-..     ... ... ... ... ...
  SYSNAME: ........ ....-..  ......-..     ... ... ... ... ...

  *SMSVSAM SMF 42 RECORDS ARE WRITTEN FROM THIS SYSTEM.
   RECORDS ARE WRITTEN WHEN SMF GLOBAL INTERVAL EXPIRES.
```

Figure 5-29 SMF 42 recording

VARY SMS,SMSVSAM, xxxxx

Use this command to manage SMSVSAM data sets or the SMSVSAM address space. Specify one of the following parameters in place of “xxxxx”:

- ACTIVE

  Restarts the SMSVSAM server and re-enables the automatic restart facility for the server. This command does not function if the SMSVSAM address space is terminated with a FALLBACK command.

- SPHERE

  Clears the VSAM-quiesced state for the specified sphere. Normally, this operation is done under application program control. This command is required only in rare circumstances.

- FALLBACK

  This parameter is used as the last step in the disablement procedure to fall back from SMSVSAM processing. For the SMSVSAM fallback procedure, see the z/OS DFSMSdfp Storage Administration Reference.

- TERMINATESERVER

  Abnormally terminates an SMSVSAM server. The server will not automatically restart after the termination. After completing your recovery action, restart the SMSVSAM server with the V SMS,SMSVSAM,ACTIVE command. Use this command only for specific recovery scenarios that require the SMSVSAM server to be down and not restart automatically.

- FORCEDELETELOCKSTRUCTURE

  Deletes the lock structure from the coupling facility, and deletes any data in the lock structure at the time the command is issued. You must reply to the confirmation message with the response FORCEDELETELOCKSTRUCTURESMSVSAMYES before the command takes effect. Use this command only in the event of a volume loss.

D SMS,SMSVSAM,QUIESCE

This parameter displays the status of all active VSAM record-level sharing (VSAM/RLS) data set quiesce events on the system that the command is entered on. This is not a sysplex-wide command.

5.6.2 Working with the CF structures

The following commands display and alter CF RLS structures:

- D XCF,STR,STRNAME=structurename
- SETXCF START,ALTER,STRNM=structurename,SIZE=nnn
- D SMS,CFLS(structurename|ALL)
- D SMS,CFCACHE(structurename or *)
- D SMS,CFVOL(volid)
- VARY SMS,CFCACHE(structurename)
- VARY SMS,CFVOL(volid)
- VARY SMS,CFLS(lockstructurename)

D XCF,STR,STRNAME=*structurename*

Use this command to display the status and size of any CF structure. Figure 5-30 on page 255 shows a sample output of the command that was issued to check the status of primary lock structure IGWLOCK00.

The output of the D XCF,STR,STRNAME=IGWLOCK00 command includes the following useful information:

- STATUS: ALLOCATED
- POLICY SIZE: 28600K
- Systems that are connected to this structure:
  - SC63
  - SC64
  - SC65
  - SC70

```
-D XCF,STR,STRNAME=IGWLOCK00
  IXC360I  13.48.30  DISPLAY XCF 914
  STRNAME: IGWLOCK00
   STATUS: ALLOCATED
   EVENT MANAGEMENT: POLICY-BASED
   TYPE: LOCK
   POLICY INFORMATION:
    POLICY SIZE    : 28600 K
    POLICY INITSIZE: 14300 K
    POLICY MINSIZE : 0 K
    FULLTHRESHOLD  : 80
    ALLOWAUTOALT   : NO
    REBUILD PERCENT: 75
    DUPLEX         : ALLOWED
    ALLOWREALLOCATE: YES
    PREFERENCE LIST: CF1      CF2
    ENFORCEORDER   : NO
    EXCLUSION LIST IS EMPTY

   ACTIVE STRUCTURE
   ----------------
    ALLOCATION TIME: 01/08/2011 10:37:03
    CFNAME         : CF1
    COUPLING FACILITY: 002094.IBM.02.00000002991E
                       PARTITION: 0F   CPCID: 00
    ACTUAL SIZE    : 14 M
    STORAGE INCREMENT SIZE: 512 K
    USAGE INFO      TOTAL     CHANGED    %
     ENTRIES:       32514           6    0
     LOCKS:       1048576
    PHYSICAL VERSION: C7266F3D AE0ED78B
    LOGICAL  VERSION: C7266F3D AE0ED78B
    SYSTEM-MANAGED PROCESS LEVEL: 8
    XCF GRPNAME    : IXCLO001
    DISPOSITION    : KEEP
    ACCESS TIME    : NOLIMIT
    NUMBER OF RECORD DATA LISTS PER CONNECTION: 16
    MAX CONNECTIONS: 8
    # CONNECTIONS  : 4

  CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME  ASID STATE
  ---------------- -- -------- -------- -------- ---- ----------------
  SC63             01 0001012C SC63     SMSVSAM  000A ACTIVE
  SC64             02 00020135 SC64     SMSVSAM  000A ACTIVE
  SC65             03 00030149 SC65     SMSVSAM  000A ACTIVE
  SC70             04 000400C7 SC70     SMSVSAM  0066 ACTIVE

  DIAGNOSTIC INFORMATION:  STRNUM: 00000001 STRSEQ: 00000000
                           MANAGER SYSTEM ID:  00000000

 EVENT MANAGEMENT: POLICY-BASED
```

Figure 5-30 Display of primary RLS lock structure IGWLOCK00

SETXCF START,ALTER,STRNM=*structurename*,SIZE=*nnn*

You can use this command to increase a structure size to a new *nnn* value. This command works only if there is space available between the current and the maximum size of the structure. You can verify whether there is space available by looking at the output of the D XCF,STR,STRNAME=*structurename* command.

If there is not any available space, you can create and activate a new CFRM policy to increase the Max Size for the structure.

D SMS,CFLS(*structurename*|ALL)

This command displays the following information about a CF lock structure:

- *LockRate*: The number of locks that are required per second (shared and exclusive)
- *ContRate*: The percentage of lock requests globally managed in the CF. If you are running your VSAM RLS data set in just one system, this number is zero.
- *FContRate:* The percentage of lock requests falsely globally managed. It indicates the percentage of false contention in the structure.
- *WaitQLen:* The average number of requests that are waiting behind locks. This field usually increases when LockRate increases, passing from a larger period of observation to a small period (1 hour to 1 minute, for example). If WaitQueue increases just by itself, it indicates a performance problem.

See Figure 5-31 for a sample output of the command. This example displays information about a secondary lock structure called RLSLOCK_TEST00. This structure was implemented less than a day before the command was issued, so the field 1 Day shows only dashes.

```
-D SMS,CFLS(RLSLOCK_TEST00)
 IEE932I 580
 IGW320I 13:01:00 Display SMS,CFLS(RLSLOCK_TEST00 )
 PRIMARY STRUCTURE:RLSLOCK_TEST00 VERSION:C7C037437705E106 SIZE:4096K
 RECORD TABLE ENTRIES:10275 USED:0
 LOCK STRUCTURE MODE: SIMPLEX  STATUS: ENABLE
 System   Interval   LockRate   ContRate  FContRate   WaitQLen
 SC63     1 Minute     1081.9      1.670      0.887       0.00
 SC63      1 Hour       101.5      2.734      0.589       0.03
 SC63      8 Hour        12.7      2.734      0.589       0.00
 SC63       1 Day  ---------- ---------- ---------- ----------
   (03)   1 Minute      683.5      1.177      0.334       0.11
   (03)    1 Hour        72.4      1.709      0.276       0.02
   (03)    8 Hour         9.0      1.709      0.276       0.00
   (03)     1 Day         0.0      0.000      0.000       0.00

 ***************** LEGEND ******************
   LockRate = number of lock requests per second
   CONTRATE = % of lock requests globally managed
   FCONTRATE = % of lock requests falsely globally managed
   WaitQLen = Average number of requests waiting for locks
```

Figure 5-31 D SMS,CFLS(structurename) sample output

If you enter D SMS,CFLS alone, without a lock structure name, z/OS shows you the primary lock structure, IGWLOCK00.

D SMS,CFCACHE(*structurename* or *)

This command displays SMS-related information about cache structures in the CF. See Figure 5-32 for a sample output.

```
-D SMS,CFCACHE(*)
  IEE932I 937
  IGW530I DFSMS CF STRUCTURES

  DFSMS CF CACHE STRUCTURE TO SYSTEM CONNECTIVITY

  SYSTEM         ==>  00000000011111111112222222222333
  IDENTIFIER     ==>  12345678901234567890123456789012

  RLS_CACHE           +...............................
  ................    ................................
  ................    ................................

  SYSTEM  1 = SC63     SYSTEM  2 = ........ SYSTEM  3 = ........
  SYSTEM  4 = ........ SYSTEM  5 = ........ SYSTEM  6 = ........
  SYSTEM  7 = ........ SYSTEM  8 = ........ SYSTEM  9 = ........
  SYSTEM 10 = ........ SYSTEM 11 = ........ SYSTEM 12 = ........
  SYSTEM 13 = ........ SYSTEM 14 = ........ SYSTEM 15 = ........
  SYSTEM 16 = ........ SYSTEM 17 = ........ SYSTEM 18 = ........
  SYSTEM 19 = ........ SYSTEM 20 = ........ SYSTEM 21 = ........
  SYSTEM 22 = ........ SYSTEM 23 = ........ SYSTEM 24 = ........
  SYSTEM 25 = ........ SYSTEM 26 = ........ SYSTEM 27 = ........
  SYSTEM 28 = ........ SYSTEM 29 = ........ SYSTEM 30 = ........
  SYSTEM 31 = ........ SYSTEM 32 = ........

  DFSMS CF CACHE                 SPHERES CONNECTED  ACTIVE CACHE
  STRUCTURE STATUS:              THIS-SYS:   TOTAL: FEATURE LEVEL:

  RLS_CACHE       = ENABLE            0        3 -NONE-
  ................ = ............      -        -
  ................ = ............      -        -
  ................ = ............      -        -


  DFSMS CF CACHE FEATURE STATUS:
    Z  = No Caching Advanced functions are available
    A  = SMS RlsCfCache Data Class values honored
    Mx = IGW500I Open Sphere Message Status:
       M0 = Suppress All Open Sphere IGW500I Messages
       M1 = Display All Open Sphere IGW500I Messages
       M2 = Display First Open Sphere IGW500I message in sysplex
       M3 = All Open Sphere IGW500I messages in sysplex to HardCopy
       M4 = First Open Sphere IGW500I message in sysplex to HardCopy
```

Figure 5-32 Displaying RLS cache structures

The output of D SMS,CFCACHE command contains the following useful information:

- The names of the Cache Structures. The example sysplex has one cache structure named RLS_CACHE
- A matrix that shows which system this structure is connected. In the example sysplex, only partition SC63 is connected to this structure
- SPHERES CONNECTED:
  - THIS SYS: 0
  - TOTAL: 3 (three VSAM data sets are connected to this structure, probably being accessed in RLS mode by SC63)

Attention: If there is the name of the structure without the system names, the structure is defined to SMS but does not exist in the CF.

D SMS,CFVOL(volid)

This command displays a list of coupling facilities cache structures that contain data for the specified volume (volid) and the status of the volume.

VARY SMS,CFCACHE(*structurename*)

Issue this command to change the state of a cache structure and specify the name of the cache structure.

ENABLE	If you specify ENABLE, VSAM RLS data can be stored in cache structure. This is the normal state of operations, and is the state the coupling facility cache structure is in after sysplex IPL.
QUIESCE	If you specify QUIESCE, you cannot store any VSAM RLS data in the cache structure. The QUIESCE operation is not complete until the state of the volume is quiesced. Use the D SMS,CFVOL command to determine the state of the volume.

VARY SMS,CFVOL(*volid*)

Use this command to change the state of a volume as it relates to coupling facility cache structures. Specify the volume (volid).

ENABLE	If you specify ENABLE, data that are contained on this volume can be stored in a coupling facility cache structure. This is the normal state of operations.
QUIESCE	If you specify QUIESCE, you cannot store any data from the volume on the coupling facility cache structure.

VARY SMS,CFLS(*lockstructurename*)

You can use this command to change the state of a secondary lock structure. You can Enable and Quiesce the structure:

ENABLE	If you specify ENABLE, VSAM RLS secondary lock structures can be accessed. When the lock structure is enabled, SMSVSAM attempts to connect to the structure. This structure is marked available when a VSAM data set eligible to connect to this lock structure is opened in RLS mode.
QUIESCE	If you specify QUIESCE, VSAM RLS secondary lock structures cannot be accessed. Any VSAM data set that is not open for VSAM RLS access is not allowed to select the specified lock structure name. All

existing usage of this secondary lock structure is not affected. When all data sets across all systems that are assigned to this secondary lock structure close, the secondary lock structure transitions from Quiescing to Quiesced.

When the secondary lock structure transitions to Quiesced state, SMSVSAM does not issue the MVS command to have the lock structure UNALLOCATED in the coupling facility. To deallocate a secondary lock structure, issue this operator command:

```
V SMS,FORCEDELETELOCKSTRUCTURE(lockstructurename)
```

5.6.3 Working with the SHCDS

Share Control Data Sets are key elements in maintaining data integrity in a VSAM RLS environment. You can use the following commands to work with them:

- D SMS,SHCDS
- VARY SMS,SHCDS(shcdsname)

D SMS,SHCDS

This command displays the following information about the sharing control data sets:

- Name
- Size
- Amount of free space for the active and spare SHCDS
- Whether the data set is usable

Figure 5-33 shows sample output. Note that it displays only the last two qualifiers of the SHCDS names. Read it as *SYS1.DFPSHCDS.WTSCPLX2.VSBOX48*, and so on.

```
-D SMS,SHCDS
  IEE932I 959
  IGW612I 14:44:47    DISPLAY SMS,SHCDS
  Name                      Size    %UTIL Status  Type
  WTSCPLX2.VSBOX48       10800Kb      4%  GOOD    ACTIVE
  WTSCPLX2.VSBOX52       10800Kb      4%  GOOD    ACTIVE
  WTSCPLX2.VSBOX49       10800Kb      4%  GOOD    SPARE
  -----------------          0Kb      0%  N/A     N/A
  -----------------          0Kb      0%  N/A     N/A
  -----------------          0Kb      0%  N/A     N/A
  -----------------          0Kb      0%  N/A     N/A
  -----------------          0Kb      0%  N/A     N/A
  -----------------          0Kb      0%  N/A     N/A
  -----------------          0Kb      0%  N/A     N/A
```

Figure 5-33 Displaying SHCDS status

VARY SMS,SHCDS(*shcdsname*)

Issue this command to add or delete a sharing control data set (SHCDS). It has the following options:

NEW	If you specify NEW, a new active SHCDS named (shcdsname) is added.
NEWSPARE	If you specify NEWSPARE, a new spare SHCDS named (shcdsname) is added.
DELETE	If you specify DELETE, an SHCDS named (shcdsname) is deleted. This SHCDS can be either an active or a spare SHCDS.

The sharing control data set (SHCDS) is identified by the dsname SYS1.DFPSHCDS.qualifier.Vvolser. When you specify its name (shcdsname) in this command, do not use the fully qualified name. Use only *qualifier.Vvolser* as the shcdsname, without the SYS1.DFPSHCDS prefix, when you specify the name of the SHCDS.

Attention: For more information about a procedure to replace one or all of your SHCDS, see "Replacing an SHCDS" on page 246.

5.6.4 Monitoring data sets

Use the following commands to activate and display coupling facility statistics monitoring for data sets:

- D SMS, MONDS(specmask or *)
- VARY SMS,MONDS(dsname{,dsname...}),ON|OFF

D SMS, MONDS(specmask or *)

This command displays the data set names specifications eligible for coupling facilities statistics monitoring. This monitoring is activated by the command VARY SMS,MONDS. You can specify a full or partial data set name (specmask) to view a subset of the data set specifications. You must specify at least one high-level qualifier. A wildcard in the data set name cannot be followed by more qualifiers. Specify '*' to display all the data sets specifications eligible for coupling facilities statistics monitoring.

VARY SMS,MONDS(dsname{,dsname...}),ON|OFF

This command specifies the data set name (dsname) or data set names (dsname{,dsname...}). If you want to be eligible for coupling facility statistical monitoring, specify ON. The commands are now generic for the CF structures.

To indicate that the specified data set is no longer eligible for statistical monitoring, specify OFF. Monitoring is tracked through SMF record 42 subtype 16. You can specify a full or partial data set name with at least one high-level qualifier. An asterisk cannot be followed by other qualifiers. You can specify up to 16 data set names with each command. This command affects activity for the specified data sets across all systems in the sysplex.

5.6.5 Changing SMS parameters dynamically

You can update your IGDSMSxx parmlib member and issue the SET SMS=xx command to make your update effective immediately.

You can also use the SETSMS command to dynamically change a subset of SMS parameters from the console, without changing the parmlib member IGDSMSxx. You can use the following commands to alter SMS parameters that are closely related to RLS:

- SETSMS RLS_MAXCFFEATURELEVEL({A|Z})

  This command specifies the method that VSAM RLS uses to determine the size of the data that is placed in the CF cache structure. If you specify A, caching proceeds using the RLSCFCACHE keyword characteristics specified in the SMS data class that is defined for the VSAM data set. If you do not specify a value, or specify Z, only VSAM RLS data that have a Control Interval (CI) value of 4K or less are placed in the CF cache structure. The default is Z. For more information, see 5.4.6, “Modifying the PARMLIB IGDSMSxx Member” on page 233.

- SETSMS BMFTIME(nnnnn)

  This command specifies that SMS allows nnnnn seconds (1 - 86399) to elapse between the production of SMS BMF records. The default is 3600 seconds.

- SETSMS CF_TIME(nnn or 3600)

  This command specifies the number of seconds between recording SMF 42 records that are related to SMSVSAM coupling facility use (subtypes 15, 16, 17, 18 and 19). SMF_TIME(YES) overrides this parameter.

- SETSMS DEADLOCK_DETECTION(iiii,kkkk)

  This command specifies the deadlock detection intervals that are used by SMS.

  iiii is a 1 - 4 digit numeric value in the range 1-9999 that specifies the length in seconds of the local deadlock detection interval. The default for *iiii* is 15 seconds.

  kkkk is a 1 - 4 digit numeric value in the range 1-9999. It specifies the number of local deadlock cycles that must expire before global deadlock detection is run. The default for kkkk is four local cycles.

- SETSMS DSSTIMEOUT(nnnn)

  Use this command to alter the DSSTIMEOUT value dynamically. Its range is from 0 to 65536.

- SETSMS RLS_MAX_POOL_SIZE(nnn)

  You can use this command to alter the maximum size in megabytes of the SMSVSAM data space 31-bit local buffer pool. SMSVSAM tries not to exceed the buffer pool size you specify, although more storage might be temporarily used. Because SMSVSAM manages buffer pool space dynamically, this value does not set a static size for the buffer pool.

- SETSMS RLSABOVETHEBARMAXPOOLSIZE

  You can use this command to alter the total size in megabytes of the SMSVSAM above the bar local buffer pool. You can change this value for all the systems the sysplex, or assign a different value for each system in the sysplex:

  – SETSMS RLSABOVETHEBARMAXPOOLSIZE(ALL,*size*)

    Changes the size of the above the bar buffer pool in all the systems in the sysplex;

  – SETSMS RLSABOVETHEBARMAXPOOLSIZE(*system1,size1,system2,size2*.....)

    Changes the size of the above the bar buffer pool for each individual system that you specify in the command. This process allows you to assign a different size for each system.

- SETSMS RLSFIXEDPOOLSIZE

  You can issue this command to specify the amount of real storage that is fixed for the buffers either below or above the bar. You can change this value for all the systems in the sysplex, or assign a different value for each system in the sysplex:

  – SETSMS RLSFIXEDPOOLSIZE(ALL,size)

    Specifies the amount of fixed real storage for all systems in the sysplex.

  – SETSMS RLSFIXEDPOOLSIZE(system1,size1,system2,size2,....)

    Specifies the amount of fixed real storage for each system that is referenced in the command.

5.6.6 Fallback procedure

Use the following procedure if you want to stop using RLS in a sysplex:

1. Ensure that no applications that access VSAM data sets in RLS mode are running.
2. Ensure that there are no outstanding recovery requirements.

   You can use IDCAMS SHCDS LISTRECOVERY or LISTSUBSYSDS to find any recoveries that must be completed before you continue with the fallback procedure. You must resolve situations, such as retained locks, that might compromise data integrity before you continue.
3. Change all cache definitions in the SMS Storage Classes constructs to blank. Also, change all the lock set definitions in the Storage Classes to blank. Activate this SMS configuration.
4. Quiesce all RLS CF cache structures by issuing the following command:

   ```
   VARY SMS,CFCACHE(CF_cache_structure_name),QUIESCE
   ```

   Issue this command for all RLS CF cache structures. Use the D SMS,CFCACHE command to verify that all structures are quiesced.
5. Reset the RLS indicators in all applicable catalogs by using the SHCDS CFRESET command.
6. Update the CICS file definitions for the affected VSAM data sets. Remove RLSACCESS=YES from their file definitions.
7. Change RLSINIT to NO in the parmlib member IGDSMSxx of all systems in the sysplex. Activate this configuration.
8. Terminate all SMSVSAMs in the sysplex by issuing the following command:

   ```
   ROUTE *ALL,VARY SMS,SMSVSAM,TERMINATESERVER
   ```
9. Reply “C” to any outstanding IGW418D message:
   a. If for any reason SMSVSAM does not terminate, try the following command:

      ```
      FORCE SMSVSAM,ARM
      ```
   b. If the previous command does not terminate SMSVSAM, try this command:

      ```
      FORCE SMSVSAM
      ```
10. Complete the FALLBACK by issuing the following command:

    ```
    VARY SMS,SMSVSAM,FALLBACK
    ```
11. Ensure that all SMSVSAM address spaces are disabled.

12. Respond to message IGW523 with this command:

```
FALLBACKSMSVSAMYES
```

13. Delete the secondary lock structures by issuing the following MVS command:

```
SETXCF FORCE,STRUCTURE,STRNAME(strname1,strname2,....)
```

Remember: Do not use this procedure for normal or abnormal termination of SMSVSAM address space.

5.6.7 Switching between RLS and non-RLS mode

Switching between RLS mode is mostly appropriate for CICS users who are using VSAM RLS recoverable data sets, without DFSMStvs implemented.

Batch applications that use VSAM RLS can read the VSAM data while CICS is active. If the VSAM data set is defined as recoverable, batch applications that do not use TVS are not allowed to update the data set.

VSAM RLS does not change how CICS and batch jobs are scheduled. It continues to be done by using batch window scheduling. To run batch jobs against your recoverable data sets, you must switch the access of the VSAM recoverable data sets from RLS to non-RLS mode. Then, run your batch applications, and switch the data sets back to RLS mode. Use the Quiesce protocol to switch your VSAM data sets between RLS and non-RLS mode.

CICS uses the QUIESCE command to broadcast and coordinate the quiesce across all CICS AORs across the sysplex. The QUIESCE command is issued from a single AOR.

When QUIESCE completes, you can run your batch jobs, accessing your data sets in non-RLS mode.

Recoverable/nonrecoverable data set attribute does not have any meaning to NSR/LSR, which is the way most job batches work. It is therefore ignored.

Figure 5-34 on page 264 shows the flow of the QUIESCE command:

1. CICS administrator or operator issues the command to quiesce a VSAM data set that has been accessed by CICS in RLS mode:

   ```
   F cicsname,CEMT SET DSN(datasetname.*),QUI
   ```

2. SMSVSAM calls the QUIESCE exit to all registered CICS regions in this system. Through XCF communication, it informs the other SMSVSAM in the sysplex that the data set is to be quiesced.
3. CICS QUIESCE exit schedules all the close tasks to close the data set. This process happens in all CICS regions that have opened that data set.
4. Each CICS region tells SMSVSAM that the data set is quiesced.
5. After all CICS regions declare that the data set has completed QUIESCE processing, SMSVSAM updates the catalog entry for data set with the QUIESCED status. The data set can now be opened in non-RLS mode.
6. SMSVSAM returns the QUIESCE command to the CICS administrator or operator.

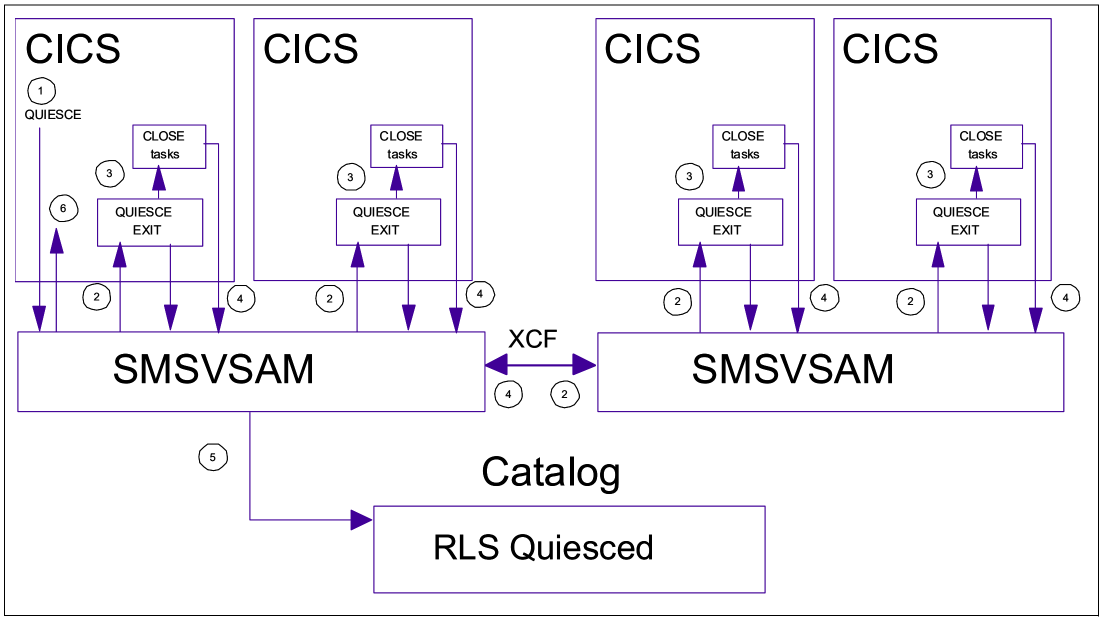


Figure 5-34 QUIESCE a data set

This communication between all the SMSVSAM instances in the sysplex always occur, even if the system has not opened any data set in RLS mode. The SMSVSAM that first received the QUIESCE must confirm that the other instances are not accessing the same data set. A partition that is defined with low priority and SMSVSAM active can increase the time for the QUIESCE processing.

After you run the batch jobs, you must unquiesce the data set. That is, CICS must be able to open the same data set in RLS mode again. The following command unquiesces a VSAM data set:

```
F cicsname,CEMT SET DSN(datasetname.*),QUI
```

For more information about QUIESCE processing and sample programs that you can include in the automation of the QUIESCE function, see *CICS and VSAM Record Level Sharing: Implementation Guide*, SG24-4766.

5.6.8 Working with the IDCAMS SHCDS command

Use the SHCDS command to list SMSVSAM recovery that is associated with VSAM RLS data sets, and to control that recovery. This command works both in batch and in the TSO/E foreground.

Figure 5-35 shows an example of SHCDS command in batch. This command information is about all subsystems (CICS, SMSVSAM, or TVS instance), such as about retained locks held.

```
//MHLRES3S JOB (999,POK),'MHLRES3',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=6M
//ALTER   EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  SHCDS LISTSUBSYS(ALL)
/*
```

Figure 5-35 SHCDS batch example

Under TSO/E, you can issue the same command under ISPF command shell, as shown in Figure 5-36. The results are displayed on your terminal.

```
 Menu  List  Mode  Functions  Utilities  Help
 sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                               ISPF Command Shell
 Enter TSO or Workstation commands below:

 ===> SHCDS LISTSUBSYS(ALL)



 Place cursor on choice and press enter to Retrieve command

 => SHCDS LISTSUBSYS(ALL)
 =>
 =>
 =>
 =>
```

Figure 5-36 SHCDS command under TSO

Figure 5-22 on page 244 shows an output example for the SHCDS LISTSUBSYS(ALL) command.

5.7 RLS enhancements

This section includes descriptions of the following RLS enhancements:

- VSAM extended addressability
- VSAM RLS CF lock structure duplexing and rebuilding
- RLS CF caching enhancements
- VSAM RLS data buffers above the 2-GB bar
- VSAM data striping
- Reliability and availability enhancements
- Multiple lock structures

5.7.1 VSAM extended addressability

VSAM data sets that are defined with extended addressability can be accessed in RLS mode. As in VSAM non-RLS extended addressability, VSAM RLS extended addressable data sets must have their data classes that are defined with these parameters:

```
Data Set Name Type =EXT
If Ext =P or R
Extended Addressability =Y
```

5.7.2 VSAM RLS CF lock structure duplexing and rebuilding

VSAM RLS goes to lost locks state when a CF lock structure is lost and a system is also lost, which is a disruptive condition.

This process consists of the lock structure rebuild with all of the active VSAM RLS instances. The process includes allocating a new structure instance, propagating the lock data to the new structure, and switching over to using the new structure instance.

Currently, z/OS supports four types of lock structure rebuild processes:

- User-managed rebuild
- User-managed duplexing rebuild
- System-managed rebuild
- System-managed duplexing rebuild

User-managed rebuild

A CF lock structure might fail and need to be rebuilt if the CF lock structure named IGWLOCK00 does not exist. It might also fail if it is not connected to the system attempting to open a VSAM data set in RLS mode. In either case, DFSMS internally initiates a rebuild that queues all applications that are using VSAM RLS during the rebuild process. The rebuild is transparent to these applications.

If both the CF lock structure and the system fail, all recoverable data sets open for VSAM RLS processing at the time of failure are converted to “lost locks”. The data sets become unavailable to any processing besides recovery processing (such as backouts). No new sharing is allowed until the recovery processing is complete. A new CF lock structure must be available to perform the recovery processing.

In this case, you can perform one of these remedies:

- Redefine or replace the CF lock structure
- Correct the problem that caused the unavailability
- Move the work requiring VSAM RLS to another system that has connectivity to an available CF lock structure

System-managed duplexing rebuild

Using the system-managed duplexing rebuild function, you can create and maintain a duplex (secondary) copy of the lock structure in case of failure. When the structure is in duplex mode and a failure occurs, VSAM RLS switches from the failed lock structure to the active lock structure. Record-level sharing can then proceed. The lock structure reverts to simplex mode, and continues to operate as it would in user-managed mode.

The VSAM RLS duplexing support applies only to the lock structures, and not to the cache structures.

To use the VSAM RLS system-managed duplexing function, perform the following tasks:

1. Review the CF configuration

   Evaluate the CF configuration (storage, links, processor capacity) and make any necessary configuration changes to accommodate the new VSAM RLS lock structure instances that result from system-managed duplexing.

2. Format the CFRM couple data set.

   Format this set as system-managed and duplexing-capable. Bring that CFRM couple data set into use in the sysplex to enable CFRM for system-managed duplexing.

3. Modify the CFRM policy.

   Modify this policy to control the sizing, placement (through the CF preference list), and the DUPLEX parameter indication for the VSAM RLS lock structures. These structures are duplexed by using system-managed duplexing. Finally, activate this CFRM policy.

Restriction: When a lock structure is in duplex mode, user command rebuild requests are rejected. You must use the alter function to change the lock structure.

5.7.3 RLS CF caching enhancements

There are two RLS caching enhancements:

- Dynamic cache reassignment
- Data CIs larger than 4 KB in cache structure

Dynamic cache reassignment

RLS cache structures do not support duplexing, but they can be dynamically rebuilt and reassigned if there is a failure. This capability allows you to better use the CF resources.

For the CF structure rebuild process to work, extra space in a CF must be reserved. If there is no extra space, the process fails. All I/O requests for those VSAM data sets then fail.

If an RLS cache structure fails, z/OS tries to rebuild it. If the failure is a result of a CF problem, z/OS tries to rebuild the structure in another CF according to the CFRM policy preference list for the structure.

If the rebuild fails, SMSVSAM Dynamic cache reassignment code examines each VSAM data set that is assigned to the failing cache. Data sets that are assigned to a storage class that points to a cache set where more than one cache structure name is reassigned. The same algorithm that is used in the original data set allocation is used. For the VSAM data sets that were able to be reassigned, all requests continue to work. For the VSAM data sets in which the reassignment failed, those requests are failed with the reason code `cache not available`.

Data CIs larger than 4 KB in cache structure

SMSVSAM supports caching of RLS data CI with any size, including CIs greater than 4 KB. You can cache all, some, or none of the VSAM RLS data in the CF cache structures that are defined to SMSVSAM. This caching is done through the RLS CF Cache Value in the SMS Data Class construct. The VSAM index structure is ALWAYS cached. The directory entry for a CF data entry also is always cached, regardless of what option is specified.

You can implement this support by using these techniques:

- Specifying RLS_MaxCfFeatureLevel(A) in the SYS1.PARMLIB member IGDSMSxx
- Updating RLS CF Cache Value field in the SMS Data Class construct according to your needs

For more information about the RLS CF Cache Value field of SMS Data Class, see "Data Class attributes for RLS" on page 233. For more information about the RLS_MaxCFFeature Level keyword of IGDSMSxx, see "RLS_MaxCfFeatureLevel" on page 235.

5.7.4 VSAM RLS data buffers above the 2-GB bar

Each image in the sysplex has one 31-bit local buffer pool, which is in the SMSVSAM data space, with a current maximum size of 1.7 GB. In addition to this 31-bit buffer, you can make SMSVSAM to allocate a 64-bit buffer pool in the SMSVSAM address space. Allocate the larger pool by specifying RlsAboveTheBarMaxPoolSize in the IGDSMSxx member of SYS1.PARMLIB.

For more information, see "RlsAboveTheBarMaxPoolSize" on page 235.

5.7.5 VSAM data striping

With z/OS V1R12, you are now able to access VSAM striped data sets in RLS mode. The considerations and restrictions are the same as for non-RLS VSAM striped data sets.

You must take care in a sysplex that has z/OS V1R12 images and pre-z/OS V1R12 images, if you are going to share VSAM RLS data sets. You can open the VSAM striped data set in RLS mode in the z/OS images, but this generates an error (message IEC161I) on pre-z/OS V1R12 images.

5.7.6 Reliability and availability enhancements

IBM has been improving the VSAM RLS code to enhance reliability and availability in these areas:

- SMSVSAM termination improvements
- Enhanced RLS recovery
- Health Checks

SMSVSAM termination improvements

The processing of command V SMS,SMSVSAM,TERMINATESERVER has been improved to avoid unnecessary scans in the XES lock table. SMSVSAM terminates more quickly by skipping these scans.

Enhanced RLS recovery

With the original RLS, following an internal logic error, most of the RLS subcomponents were used to decide to terminate the SMSVSAM address space. This process prevents possible data integrity problems and correct environmental errors. By terminating the SMSVSAM address space, the damaged system environment is cleared and a new clean SMSVSAM address space is created. Additionally, the process of terminating and restarting the SMSVSAM address space was automated so that you do not have to manually perform these steps.

Although this process provided protection, it also created some availability problems. A logic error that occurs during a single read/write request to a single data set can result in a system outage that affects all files opened in the system. Because most of the subcomponents adopted the same strategy, it started to terminate the SMSVSAM address space for reasons other than data integrity.

The VSAM RLS development team redesigned the code in the error recovery area. For errors with individual subcomponents, the subcomponent fails, but SMSVSAM stays up if integrity is not compromised. Messages, error codes, and memory dumps are still triggered, depending on the type of failure.

Health Checks

The following are Health Checks improvements for VSAM RLS:

- VSAMRLS_DIAG_CONTENTION

  This command checks for VSAM RLS resource contention periodically. If contention is detected, it displays a contention table.

- VSAMRLS_SINGLE_POINT_FAILURE

  Allocate each SHCDS in a different volume to avoid losing your RLS environment because of a loss of access to one volume. This Health Check looks for the volumes where the SHCDS are allocated. If it finds two SHCDS on the same volume, it issues the message IGWRH202E.

5.7.7 Multiple lock structures

Since z/OS V1R10, you can define multiple, secondary lock structures for VSAM RLS workloads to reduce locking constraints.

You can use a new SMS Storage Class attribute that is called the lock set to specify a DFSMS lock structure to be used for VSAM records. An installation can define up to 256 lock sets per sysplex. When an application opens a VSAM data set, RLS processing determines which lock structure to use by checking the Storage Class that is assigned to the data set. If the Storage Class specifies a secondary lock structure (a Lock Set Name), RLS processing uses the secondary lock structure for serializing access to records in the data set. Otherwise, RLS processing uses IGWLOCK00 for all record locking.

A secondary lock structure connection persists beyond data set closure. Secondary lock structures are disconnected when the SMSVSAM address space is terminated.

To implement multiple lock structures, all z/OS images in the Sysplex must be at z/OS V1R10 or later.

The decision to change to an multiple lock structures-based environment might be based upon a need to separate workloads from interfering with each other. For example, you might separate test and production workloads, or batch and online applications.

For more information about defining multiple lock structures, see 5.4.4, “Defining secondary lock structures (optional)” on page 225.

5.8 RLS performance

The main reason to implement RLS is availability. However, there are performance considerations that are caused by the change from non-RLS to RLS mode.

Performance is a matter of subjectivity and relativity. In each performance evaluation task, you must have clear objectives, and decide which measured numbers to relate to what objectives.

This section therefore includes these two types of goals:

- Compare theoretically: Comparing a IBM CICSPlex® without RLS data sharing with a CICSPLEX of the same environment but with RLS. For more information, see “CICSPlex RLS performance comparison” on page 286.
- Compare practically (with experiments) batch workloads that are accessing VSAM clusters in non-RLS and in RLS mode. For more information, see “Using RLS mode in an ESDS organization” on page 287.

The performance monitor in the examples is Resource Measurement Facility together with some MVS commands and SMF records reports related to RLS performance.

The major difference between these two modes (RLS and non-RLS) is the use of the coupling facility (CF) by RLS, which implies performance losses and gains. The losses can be caused by the time the transaction needs to wait for CF link transport and CFCC processing, such as ensuring integrity. The gains can come from the use of CF caching capabilities to avoid an I/O operation, improving throughput.

This section addresses all of the factors that are introduced by RLS data sharing that can affect the performance of a VSAM I/O request. Some of them are connected to specific ways of implementing data sharing. Each description includes suggestions on how to improve performance that is related to that factor.

5.8.1 Speed of the CF hardware and software

RLS data sharing performance is directly affected by the speed of the hardware that is associated with the coupling facility (CF):

- The CF and host processor speeds have the following relationship:
  - The faster the CF processor, the less usage on the host processor because the host processor waits less.
  - The faster the host processor, the higher the usage because of the time spent waiting for the CF processor. MIPS are not being productively used by the host processor.
- Between dedicated or shared CF processors, generally use dedicated processors in production environments.
- For the number of CF processors in the CF logical partition, two is better than one.
- The CF type of link, such as IC, ISC, or ICB, affects the rate (MBps).

The other factor is the load that is submitted to the hardware:

- Total demand in the CF and CF links, which might cause high queue time.
- How heavy the actual RLS request is, which might cause high service time.

For more information about evaluating the performance of your CF hardware, see “Coupling Facility Usage Summary report” on page 291, and “Coupling Facility structure activity report” on page 293.

Guidelines: If you see performance problems in the CF through RMF analysis, buy faster hardware for the CF:

- Faster and more than one (if needed) dedicated CF processors
- Faster CF links
- Do not overload the CF and CF links capacity

5.8.2 DASD performance of the SHCDS data sets

The DASD performance of the SHCDS data sets is important to maintain good general RLS performance. For more information about the role that is played by this data set in SMSVSAM, see 5.4.1, “Defining Sharing Control Data Sets (SHCDS)” on page 220.

To be sure that the I/Os towards SHCDS is running well, you can look at the RMF Postprocessor Device Activity report. This report contains valuable data such as the Device Activity Rate and the Average Response Time for all the DASD volumes.

Guidelines:

- Allocate data sets in faster controllers.
- Avoid allocating the SHCDS in volumes with high activity rate.

5.8.3 CF synchronous and asynchronous requests

A key XES decision that affects performance is what to do with the MVS processor when CFCC is processing the request. There are two possibilities:

- With *synchronous processing*, the MVS processor is placed in a loop by XES, waiting for the completion of the CF request. The synchronization is between the requiring task (not placed in wait state) and the execution of the request in the CF. This type of processing has these considerations:
  - It is efficient because the requesting CP spins until response is received from CF. This process saves processor usage because the task is undispatched, which saves status, register swaps, and being dispatched again.
  - This process can burn MIPS if CF takes a long time to respond.

  Use synchronous processing for short requests.
- *Asynchronous processing*: Instead of holding the MVS processing, XES releases it after the CF requests are sent. Here are some observations on this approach:
  - This process consumes less processor time than long running synch requests.
  - This process frees up processor time for other work while the request is processed.
  - This process uses instructions in the dispatcher component because the task must be undispatched, and then redispatched.

  Use asynchronous processing for long requests.

Usually the CF requests for the IGWLOCK00 are synchronous.

XES uses the *CF Synch/Asynch heuristic algorithm*, which selects the most efficient way (in terms of *used* processor cycles) of handling *each* request. This algorithm depends on these factors:

- The current actual synch response times for each CF request type. For duplexed structures, XES keeps the time from the start of the first operation to the end of the last one.
- The speed of the processor that z/OS is running on, which determines the number of instructions that can be run while CP spins on synch request.

Guideline: Consult the RMF reports that cover synchronous and asynchronous requests. For more information about improving the performance of Synch/Asynch CF requests, see 5.8.1, “Speed of the CF hardware and software” on page 270. You cannot influence the decision between one type and the other.

5.8.4 Local buffer pool sizing

This section addresses some aspects of sizing the local buffer pools.

If SMSVSAM can find the required CI in either the 31-bit buffer pool in the SMSVSAM data space or the 64-bit buffer pool, no I/O is necessary. The CI must be valid for this to occur.

Each buffer in this buffer pool has the size of a CI. You can have 16 different types of buffer pools with different sizes. The CI size can vary from 2 KB to 32 KB.

VSAM RLS GET I/O path example

An application program direct GET request can be served by SMSVSAM in three places, in this sequence:

1. Local buffer pools
2. SYSVSAM CF cache structures
3. DASD I/O

Local buffer pool sizes: considerations

These SMSVSAM buffers CI contents are managed by Buffer Management Facility (BMF), an SMSVSAM component, by using a least recently used (LRU) algorithm. There is not a sequential look-ahead for any type of RLS access. LRU keeps the CIs that are the most referenced in the buffer pool, or the best selection of CIs as direct access is concerned. The number of buffers in the buffer pool varies dynamically. It increases when a buffer is needed by an incoming CI and decreases when a CI is deleted or stolen by the LRU algorithm. The least referenced CIs are stolen by an LRU buffer aging routine. This routine runs with a dynamic frequency. As the system works faster the more often the CIs are stolen, the less exact their measured unreferenced pattern is, and the more CPU cycles are spent.

You can influence the LRU algorithm by setting limits through the IGDSMSxx Parmlib member specification:

- RLS_MAX_POOL_SIZE
- RLSAboveTheBarMaxPoolSize.

For more information about these parameters, see 5.4.6, “Modifying the PARMLIB IGDSMSxx Member” on page 233.

You can make your VSAM RLS data sets eligible for the 64-bit buffer pool by modifying your SMS to assign them a Data Class construct. Use the following command:

```
RLS Above the 2-GB Bar . . . : YES
```

Pool Size values are a goal that the LRU tries to maintain. If more buffers are required, the pool can temporarily exceed the values set.

The LRU algorithm for the local buffers operates in four modes:

- Normal Mode
- Maintenance Mode
- Accelerated Mode
- Panic Mode

The age of a buffer is determined by the time interval since it was last referenced by the LRU buffer aging routines.

For the 31-bit buffer pool, this age is measured in unreferenced interval count (UIC). A UIC is the time interval between two LRU buffer aging routines. A buffer aging routine initially sets a goal for the UIC, and changes this goal according to the mode the LRU algorithm is in. Any buffer that has a UIC greater than the goal UIC is eligible for being released to give room to another CI (the buffer is stolen). This goal UIC is technically called *Initial_Free_UIC*.

For the 64-bit buffer pool, the age is measured in minutes.

Periodically, BMF checks the total buffer pool size utilization and compares it to the RLS_MAX_POOL_SIZE and RLSAboveTheBarMaxPoolSize values. Based on that comparison, it can change the mode of LRU operation. The behavior of the LRU algorithm is different for the buffers in the SMSVSAM data space and the 64-bit buffers.

The SMSVAM data space buffers (31-bit buffers) have these modes:

- Normal Mode

  The LRU algorithm is in Normal Mode when the 31-bit buffer total size is less than 80% of the RLS_MAX_POOL_SIZE. Normal Mode has these characteristics:

  – Invalid and paged out buffers are released
  – Buffer UIC is increased
  – Buffers can stay indefinitely in this mode

- Maintenance Mode

  LRU algorithm enters Maintenance Mode when the 31-bit buffer total size is between 80% and 120% of RLS_MAX_POOL_SIZE. Maintenance Mode has these characteristics:

  – Initial_Free_UIC is decreased by 1
  – Buffers with UIC greater than Initial_Free_UIC are released

- Accelerated Mode

  LRU algorithm enters Accelerated Mode when the 31-bit buffer total size is greater than 120% and less than twice the value of RLS_MAX_POOL_SIZE. Accelerated Mode has these characteristics:

  – Initial_Free_UIC is decreased by 4
  – Buffers with UIC greater than Initial_Free_UIC are released
  – Requests for new buffers are first served by stolen buffers
  – If a buffer is not available to steal, a get for new storage is done

- Panic Mode

  LRU enters Panic Mode when the 31-bit buffer pool total size is greater than twice the RLS_MAX_POOL_SIZE value, or 1728 Megabytes. Panic Mode has these characteristics:

  – Initial_Free_UIC is reduced by 8

  - Buffers with UIC greater than Initial_Free_UIC are released
  - Requests for new buffers are first served by stolen buffers
  - If a buffer is not available to steal, the request is put to sleep until the next LRU run.

For the 64-bit buffer pool, the LRU algorithm has the following behaviors:

- Normal Mode

  The LRU algorithm is in Normal Mode when the total 64-bit buffer pool size is less than 80% of the RLSAboveTheBarMaxPoolSize value. Buffers stay indefinitely when LRU algorithm is in Normal Mode.

- Maintenance Mode

  LRU is in this mode when the total 64-bit buffer pool size is greater than 80% and less than 90% of RLSAboveTheBarMaxPoolSize value. In this mode, buffers that are 60 minutes or older are released.

- Accelerated Mode

  The total 64-bit buffer pool size is greater than 90% and less than 100% of RLSAboveTheBarMaxPoolSize value. Accelerated Mode has these characteristics:

  - Buffers 30 minutes or older are released
  - Requests for new buffers are first served by stolen buffers
  - If a buffer is not available to steal, a get for new storage is done

- Panic Mode

  The 64-bit buffer pool size is greater than the RLSAboveTheBarMaxPoolSize. Panic Mode has these characteristics:

  - Buffers 5 minutes or older are released
  - Requests for new buffers are first served by stolen buffers
  - If a buffer is not available to steal, the request is put to sleep until the next LRU run

Fixed buffer pools

Fix real storage by either specifying a value for RLSFixedPoolSize in the parmlib member IGDSMSxx, or by issuing the SETSMS RLSFIXEDPOOLSIZE command.

Buffers are fixed on first come, first served basis. If the first data set that is accessed in RLS mode has a 4K CI size, all the fixed buffers are 4K. This setting lasts for the life of that SMSVSAM instance.

Fixed buffers can provide a significant performance improvement because the processor usage of RSM to pin and unpin real storage is avoided.

Guidelines

The amount of storage that you use for the local buffer pools, and so the values that you specify in parmlib member IGDSMSxx, depends on the amount of data that you will be accessing in RLS mode. It also depends on the rate and pattern of this access. The following are some guidelines about local buffer pool sizes:

- If you are a new RLS user, use the 31-bit buffer pool first.

  Use the default value, 100 MB, and monitor its size by using RMF Monitor III *VSAM LRU Overview* report, also called RLSLRU report.

- Monitor local buffer pool sizes in the VSAM LRU Overview panel of RMF Monitor III.

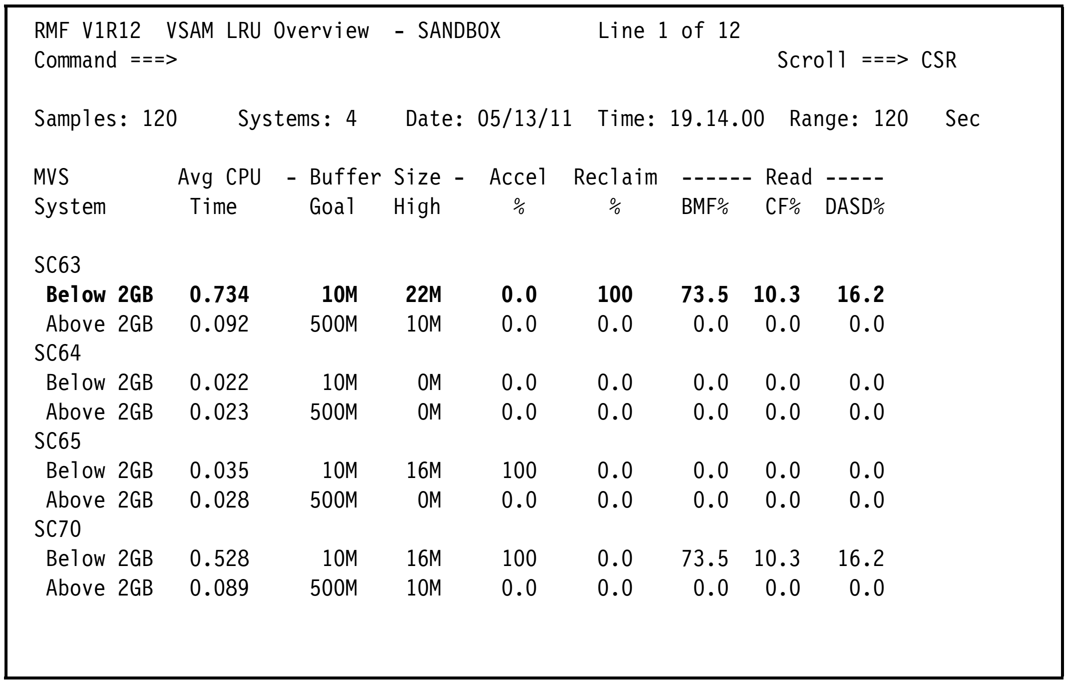
  Try to prevent LRU from entering Panic Mode frequently. This change happens in the following circumstances:

  - Buffer Size High being two times the size of Buffer Size Goal in the Below 2 GB lines

– Buffer Size High being the same size as the Buffer Size Goal in the Above 2 GB lines

Figure 5-37 shows an example of a system that exceeds two times the Goal size for the Below 2 GB buffer in system SC63. Its LRU therefore has reached Panic Mode. In this example, the 31-bit local buffer pool was decreased to 10 MB to test the practical effect of a local buffer being too undersized. Therefore, SC63 reached 22 MB, which is two times the size of the Goal.

In this example, two batch jobs that access a VSAM RLS KSDS data set were submitted, one in SC63, and the other in SC70. These jobs had an increase of the elapsed time when compared to another experiment with 100 MB of local buffer pool.



```
RMF V1R12  VSAM LRU Overview  - SANDBOX        Line 1 of 12
Command ===>                                                Scroll ===> CSR

Samples: 120     Systems: 4    Date: 05/13/11  Time: 19.14.00  Range: 120   Sec

MVS         Avg CPU  - Buffer Size -  Accel  Reclaim  ------ Read -----
System       Time      Goal   High      %       %     BMF%   CF%  DASD%

SC63
 Below 2GB   0.734      10M    22M     0.0     100    73.5  10.3   16.2
 Above 2GB   0.092     500M    10M     0.0     0.0     0.0   0.0    0.0
SC64
 Below 2GB   0.022      10M     0M     0.0     0.0     0.0   0.0    0.0
 Above 2GB   0.023     500M     0M     0.0     0.0     0.0   0.0    0.0
SC65
 Below 2GB   0.035      10M    16M     100     0.0     0.0   0.0    0.0
 Above 2GB   0.028     500M     0M     0.0     0.0     0.0   0.0    0.0
SC70
 Below 2GB   0.528      10M    16M     100     0.0    73.5  10.3   16.2
 Above 2GB   0.089     500M    10M     0.0     0.0     0.0   0.0    0.0
```

Figure 5-37 SC63 entering Panic Mode

- Keep RLS_MAX_POOL_SIZE under 850.

  This setting allows for SMVSAM to double the pool to 1700 MB before it enters Panic Mode.

- Implement the 64-bit buffer pool if your systems are reaching Panic Mode frequently.

  Note that you can assign a different RLSAboveTheBarMaxPoolSize for each system in the sysplex.

- The total amount of buffer pools should not exceed the amount of real storage available.

  A paged out buffer is immediately freed by LRU.

- Avoid batch sequential RLS processing because BMF does not implement look ahead along read sequential access for the local buffer pools. If there is a reading JOB accessing the CICS/RLS data sets, use SHROPT(2 x) and no RLS mode.

5.8.5 RLS cache structures sizing

The SYSVSAM cache structure in the CF is used when the data CI is not in local buffer pool, or is there but invalid because of cross invalidation. No I/O is necessary.

The CF cache structure is also managed by an LRU algorithm, which attempts to keep the most referenced CIs readily accessible.

To increase the probability of finding the CI in this cache, allow large cache structures. They must be at least the sum of all the local RLS buffer pools.

The ideal cache structure size would be the result of this formula:

```
Total_cache_structure_size = ((RLS_MAX_POOL_SIZE)*Number_of_SMSVSAMs_in_sysplex) +
(RLSAboveTheBarMaxPoolSize(system1) +.... + RLSAboveTheBarMaxPoolSize(systemn))
```

As an alternative to this formula, you can use the sizing tool at:

http://www-1.ibm.com/servers/eserver/zseries/cfsizer/vsamrls.html

This tool calculates the size of your cache structures based on the sum of your current LSR pool size and hiperspace pool size.

Control Interval cross invalidation

Buffer coherency must be maintained when sharing data. There are two things that drive this coherency:

- The SMSVSAM address space must to be updated to reflect that a CI in its local buffer was updated by other SMSVSAM. That is, the CI is no longer valid.
- There must be a place where any SMSVSAM can find the most current copy of a CI in its local buffer pool

To address these problems, the CF logic that is shown in Figure 5-38 is implemented.

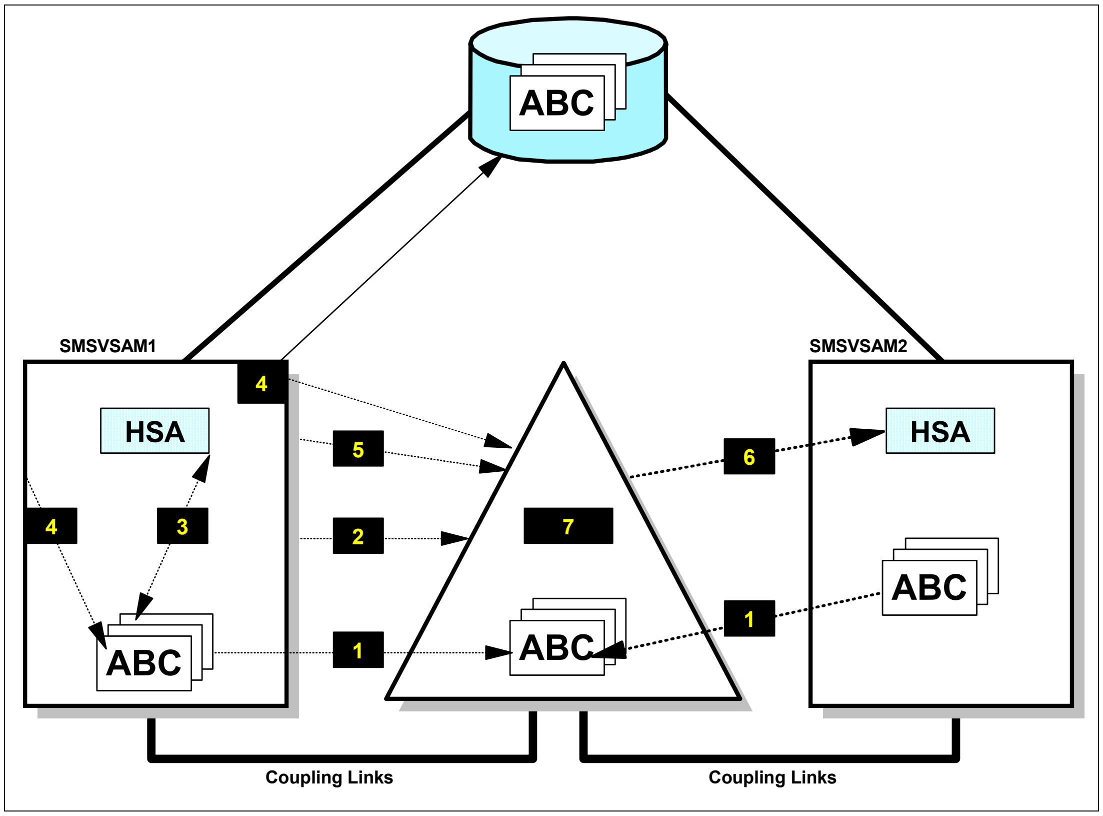


Figure 5-38 Cross invalidation and locks

These numbers refer to the numbers in the figure:

1. Every time a CI enters a local buffer pool, SMSVSAM indicates its interest in the CI to CFCC. So there are entries in the directory that do not point to a cache element, but rather to a specific SMSVSAM. The ratio between the number of directory entries and cache elements is determined by the first SMSVSAM to connect to the structure through the IXLCONN macro. The ABC control interval is in both local buffers.
2. A transaction in system 1 is requiring an update to a record in an ABC control interval. The lock is required by SMSVSAM1 to CFCC and granted.
3. SMSVSAM1 verifies that the copy in local buffer pool is valid by inspecting the HSA valid bit.
4. The CI is modified in a local buffer pool, and the CI is copied in DASD (store-through) and optionally in a CF cache element.
5. CFCC is informed by SMSVSAM1 about the change.
6. By searching the directory, CFCC discovers the other SMSVSAMs that manifested interest in the CI. These SMSVSAMs are the ones that now have an out of date copy of the ABC updated control interval. CFCC switches on the invalid bit accordingly by using IXLVECTR XES service. This action is called a *cross-invalidation* (XI).

There is an unusual case of XI called *directory reclaims XI.* It occurs when a directory entry is needed and one is not available. To reclaim an entry, its previous content is released and the corresponding pair SMSVSAM/CI affected has the HSA invalid bit set to on. This process is not because the copy of the affected CI is not valid anymore, but because CFFC is not able to track it.

Later an SMSVSAM2 receives a transaction request to be filled by the modified CI. The old copy is still in local buffer pool, but SMSVSAM2 detects the invalid bit on, using the IXLVECTR local vector service TestLocalCache. This causes a cross invalidation, and the SMSVSAM searches by the fresh copy in the CF cache elements or in DASD. When found, the new copy is brought to local buffer and made available to the transaction.

RLS cache structure balancing

Consider load balancing when you are defining multiple RLS cache structures. You can assign different SMS Storage Classes for different groups of data sets to make them use different cache structures. To do so, you must assign a different Cache Set Name for each Storage Class.

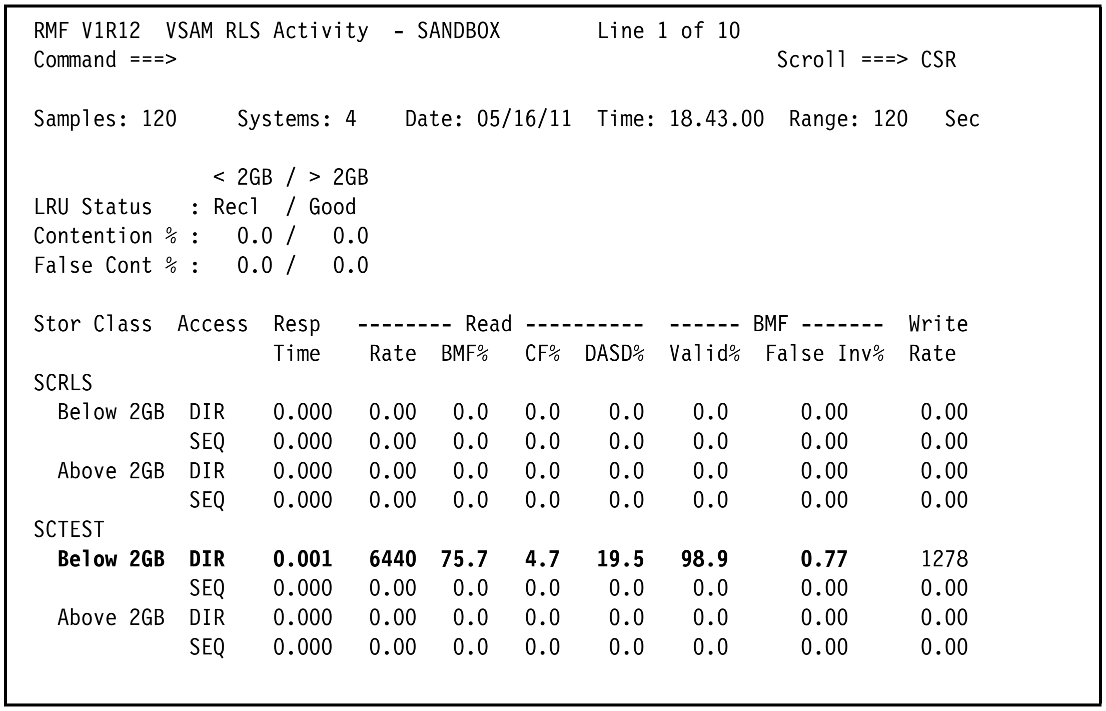
For example, you can assign smaller cache structures to VSAM data sets that are used by less important applications.

If there is more than one cache structure in a cache set, RLS OPEN attempts to balance the load between the cache structures in the cache set. The RLS balancing algorithm is not exact. It cannot predict when the activity in a data set will increase or decrease.

You can use RMF III SYSPLEX CF detail report to check cache access and reclaim rates.

Guidelines

Use the RMF III *VSAM RLS Activity by storage class* (also called RLSSC) report to identify BMF False Inv% as shown in Figure 5-39. This field shows the false invalid rate that occurs when buffers in the local buffer pools become invalid because of space reclaims in the CF cache. Ideally, the BMF false invalid rate should be as close to zero as possible. Increasing the size of the CF cache structure can decrease the false invalid rate.



```
RMF V1R12  VSAM RLS Activity  - SANDBOX       Line 1 of 10
Command ===>                                              Scroll ===> CSR

Samples: 120     Systems: 4    Date: 05/16/11  Time: 18.43.00  Range: 120   Sec

               < 2GB / > 2GB
LRU Status   : Recl  / Good
Contention % :   0.0 /   0.0
False Cont % :   0.0 /   0.0

Stor Class  Access  Resp   -------- Read ----------  ------ BMF -------  Write
                    Time    Rate  BMF%   CF%  DASD%  Valid%  False Inv%  Rate
SCRLS
  Below 2GB  DIR    0.000   0.00   0.0   0.0    0.0    0.0      0.00      0.00
             SEQ    0.000   0.00   0.0   0.0    0.0    0.0      0.00      0.00
  Above 2GB  DIR    0.000   0.00   0.0   0.0    0.0    0.0      0.00      0.00
             SEQ    0.000   0.00   0.0   0.0    0.0    0.0      0.00      0.00
SCTEST
  Below 2GB  DIR    0.001   6440  75.7   4.7   19.5   98.9      0.77      1278
             SEQ    0.000   0.00   0.0   0.0    0.0    0.0      0.00      0.00
  Above 2GB  DIR    0.000   0.00   0.0   0.0    0.0    0.0      0.00      0.00
             SEQ    0.000   0.00   0.0   0.0    0.0    0.0      0.00      0.00
```

Figure 5-39 RLSSC report

Check RMF III SYSPLEX CF detail report for cache access and reclaim rates.

Considerations for RLS_MaxCfFeatureLevel

You can control the size of data that is cached in the CF cache structures by specifying these parameters:

- RLS_MaxCfFeaturelevel(*A or Z*) in the IGDSMSxx parmlib member
- The RLS CF Cache value in the SMS Data Class

Use the following guidelines to determine what to specify for this parameter:

- RLS_MAXCfFeatureLevel(Z)
  - This setting caches CIs that are 4096 bytes or less in size. It therefore saves space in the CF cache structures by not caching large CIs.
  - This setting can be an advantage if your applications that access VSAM RLS data are mostly read only, and the local buffer pools are still valid.
- RLS_MaxCfFeatureLevel(A)
  - This setting caches CIs up to 32768 bytes in size. It therefore requires more space in the CF cache structures.
  - Use this specification if your RLS data sets are updated across the images in your sysplex.

With the RLS CF Cache Value, you can determine the amount of data that is cached:

ALL	All the data and index components are cached
NONE	Only the index is cached
UPDATESONLY	Only write requests are cached.

5.8.6 RLS lock structure sizing

This topic addresses considerations about the sizing of the VSAM RLS lock structures.

True and false contention in lock structures can cause problems. Usually you have many more records in a medium to large data set to serialize through global locks than the number of entries in theCF structure lock table. This numerical difference is because you cannot have enormous lock structures in the CF.

The solution is to share a lock entry with several logical records locks as shown in Figure 5-40. A hashing algorithm that is applied in the Key, RBA, or RRN is used to map a lock associated with the record to a lock entry. When more than one lock register maps to the same lock table entry (synonyms), there is a false contention.

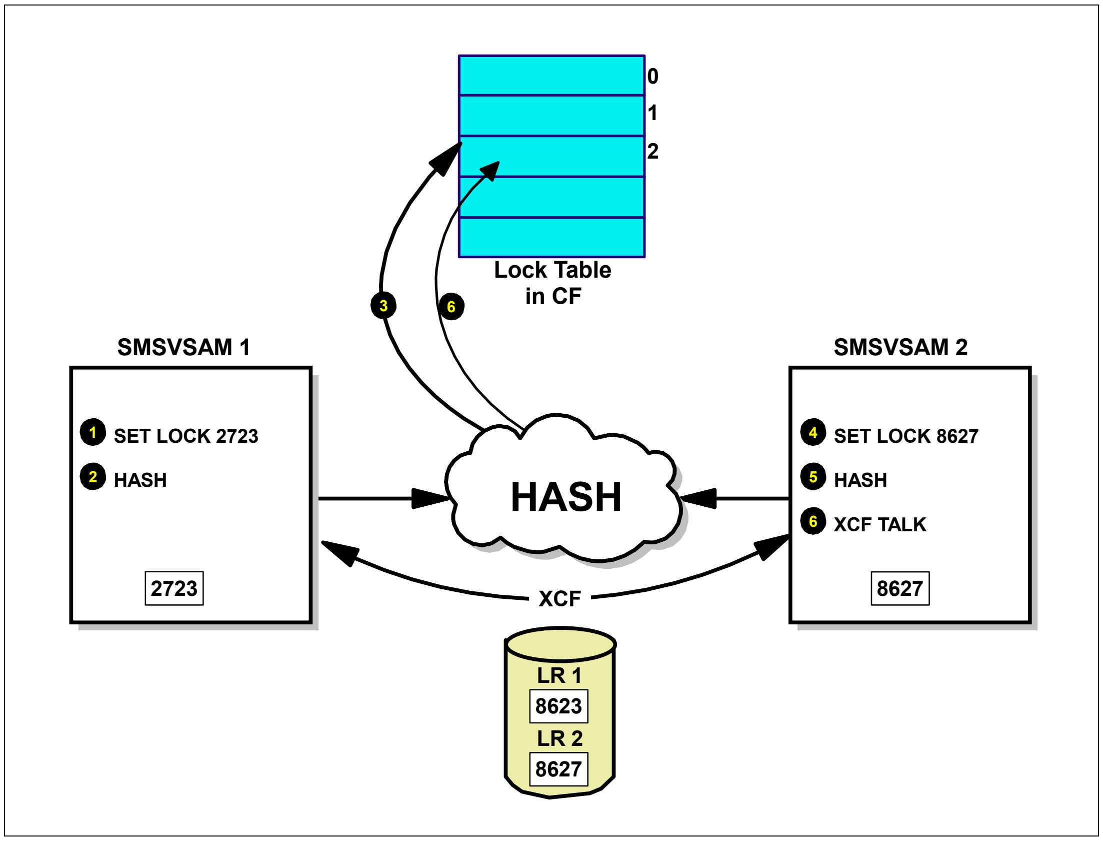


Figure 5-40 Example of two locks synonyms

In the example, it is a false contention because the lock table entry #2 is held by an exploiter in SMSVSAM1 because of a logical record 2723 lock. Another exploiter in SMSVSAM2 requires a logical record 8627 lock. It cannot get the lock because the common entry is held by the first (false contention). This contention causes a performance penalty. The second lock requester is suspended until SMSVSAM2 determines through XCF conversation (using the XCF group name IXCLO001) with SMSVSAM1, that there is no real lock contention on the resource. If the contention is true, the requesting task stays in wait state. If it is false, it is placed immediately in ready state.

To avoid performance degradation because of false contention, decrease the possibility of synonyms by using one of these methods. False contention should be less than 0.5% for an RLS lock structure.

- Decrease the length of each lock table entry by decreasing MAXSYSTEMS to 7 or less, if you have seven or less systems in your sysplex. If you have more than seven, define MAXSYSTEMS as less than 24.
- Increase the size of the lock table. Use the following formula to calculate its size:

  ```
  10M *number_of_systems *lock_entry_size
  ```

  The `loc_entry_size` depends on the MAXSYSTEM value as seen in the previous section:

7 or less:	2 bytes
>= 8 and < 24:	4 bytes
>= 24 and <= 32:	8 bytes

For more information about how to derive the size of the structure, see *DFSMSdfp Storage Administration Reference*, SC26-7402.

If you have real contention that affects more than 2% of the total locks requested), use the *no ready integrity (*NRI) option in DD card or ACB. This setting prevents shared locks from being obtained for reads. This configuration means that the reader can read before the commit of an updater. This is sometimes called a dirty read because the reader might see an uncommitted change made by another transaction. However, the buffer pool coherency is achieved, even with NRI.

You can also verify the existence of long transactions not committing. these transactions can hold many locks, which can flood the CF lock structure. With the multiple lock structure support, you can assign a different secondary lock structure to those problematic transactions.

For more information about lock contention, see “VSAM RLS Monitor III reports: RLSSC and RLSDS” on page 289 and “D SMS,CFLS(structurename|ALL)” on page 256.

Guidelines: If you have more than 0.5% false contention:

- Decrease the length of each lock table entry
- Increase the size of the lock table
- Consider using multiple lock structure

If you have more than 2% real contention:

- If possible, use NRI
- Avoid long transactions that run with scarce commits
- Avoid rerunning a JOB applying duplicate updates instead of updating from the last commit point
- Avoid backing out changes that are made by an application error (or bad/invalid output) in a specific job that runs in parallel with heavy workload against the cluster
- Consider using multiple lock structures, as an attempt to avoid floods in the primary lock structure

5.8.7 Data set level parameters

This section addresses the following performance aspects that are related to the characteristics of VSAM data sets:

- Splits
- CA reclaim
- Data component CISIZE for KSDS

Splits

CI and CA splits occur as a result of data record insertions (or increasing the length of an already existing record) in KSDS and VRRDS organizations. Keep in mind that splits are not bad. Splits are how VSAM deals with a lack of space. They generate free space that helps prevent more splits. After a split occurs, it is no longer a performance issue.

The cost of processing a split usually is in the order of:

- Some milliseconds for a CI split
- Some tens of milliseconds for a CA split
- Some hundreds of milliseconds for an EOV processing

A CI split that also results in CA split that in turn results in an allocation of another extent, can affect your performance significantly.

Guidelines:

- Do not reorganize your data sets because of CI/CA splits, unless they have many small secondary extents. In this case, consider increasing the secondary allocation amount.
- Optimize the space amount that you specify when defining/creating a new VSAM data set. If the data set that you are defining is going to be one cylinder or larger in size, do not specify a primary or secondary space amount of less than one cylinder or 15 tracks in size. If either the primary or secondary allocation is smaller than one cylinder, the smaller of the two size values is used as the CA size. A VSAM data set with CA size of one cylinder generally suffers fewer CA splits than a data set with a CA size of three tracks, for example.

CA reclaim

CA reclaim is implemented for VSAM and VSAM RLS KSDS data sets to reuse free CA space on DASD. When the freed space in placed in a free chain to be reused, the index structure can be shrunk to facilitate quicker data accesses. When space is needed for a new CA, a free CA space in the free chain is reused. This process reduces the number of calls to EOV to extend the KSDS. CA reclaim is supported on z/OS 1.12 and later.

When CA reclaim is used, space fragmentation caused by erasing records from a KSDS is minimized. This configuration reduces the need to reorganize the data set, which is detrimental to your data set’s availability.

Guideline: Implement CA reclaim to minimize the need to reorganize your VSAM KSDS data sets because of space-related problems, and to improve performance.

Data component CISIZE for KSDS

RLS access has the following considerations regarding large CI sizes for the data component of a VSAM KSDS:

- When an application needs a record, the whole CI is brought from disk into central storage and to an RLS cache structure in a Coupling Facility. Therefore, larger CI sizes tend to use more local buffer and more cache structure space.
- Larger CI sizes result in more records per CI. This configuration in turn can cause more contention on disk because larger pieces of data must be read/written to disk.
- Larger CI sizes are better for sequential processing, which is typical of batch workloads. Smaller CI sizes are better for direct processing, like online transactions.

Guidelines: Monitor the local buffer pool utilization through the RMF III RLSLRU report. For more information about sizing the local buffer pools, see 5.8.4, “Local buffer pool sizing” on page 272.

Also, consider increasing the size of your CF RLS cache structures if the BMF False Inv% value increases. For more information, see 5.8.5, “RLS cache structures sizing” on page 275.

5.8.8 Serialization at record level

In non-RLS mode, the following levels of serialization are used by DFSMS lock manager:

- At data set level (at OPEN time), using share options.
- At CI level (at GET/PUT time) using intra-address space sharing.

In RLS mode, the serialization at logical record level (instead at CI level) has the following important advantage:

- Significantly decreases the real contention because of a higher granularity

It has the following disadvantages:

- Increases the number of locks and the associated effort to manage them
- Because there are more locks, the number of false contentions is higher than if the serialization would be done at CI level.

Generally speaking, serialization at logical record favors smaller components. However, the cross invalidation process that is described in “Control Interval cross invalidation” on page 276, is still done at CI level. The following scenario can happen:

Two different logical records in the same CI might be updated at the same time on two different SMSVSAM (X and Y), which is good. X updates the first logical record in a CI holding its lock. The same CI is in the local buffer pool of Y. When Y tries to update the second logical record (also holding its lock), it detects that the buffer is now invalid. However, the second logical record was not updated by X. The CI is then reread by Y from DASD (or CF). The new CI, containing both updated records, is now written by Y to the CF and then to DASD. The invalid bit is set in X.

A logical record lock is only free at commit time, not at the end of the logical processing.

Guidelines: f you are facing high level of real and false contentions, or BMF false invalids, see 5.8.5, “RLS cache structures sizing” on page 275 and 5.8.6, “RLS lock structure sizing” on page 279. Do not forget the possibility of using NRI.

5.8.9 Effect of a write dominant I/O workload

Writes are always a source of performance, integrity, and availability problems in data processing.A write dominant I/O workload can cause performance degradation because of the following reasons:

- Excess of CI invalidations. For more information, see “Control Interval cross invalidation” on page 276.
- All the *random* writes against a CI in the local buffer pool are in store-through mode. This means that the DASD I/O operation is always run. There is no option of a defer write (store-in mode). In CICS FOR, there is such an option. However, the store-through mode favors a faster recovery. The performance gets worse because of these reasons:
  - The requesting task waits until the end of the DASD I/O operation.
  - If the same CI is updated *N* times when in the buffer pool, with the store-through mode you have *N I/O* operations. However, with defer write you have just *one* I/O operation, saving *N-1 I*/O operations.
- Many locks that are held in exclusive mode (because of the writes) cause more real contention at the logical record level.

Guidelines:

- If you have many DASD I/O writes, improve the performance of your disk volume or allocate the VSAM data set to a fast disk subsystem.
- Decrease XIs as addressed in “RLS lock structure sizing” on page 279.

5.8.10 RLS lock structure duplexing

An error in software or hardware might corrupt the contents of a CF structure or make them inaccessible because of the lack of CF links. The RLS lock structures are sensitive to structure failures. To keep continuous availability, implement System Managed Duplexing. For more information, see 5.7.2, “VSAM RLS CF lock structure duplexing and rebuilding” on page 266.

Cost of SM duplexing differs depending on structure type, usage, and the percent of requests that get duplexed. When duplexing an RLS lock structure, all the writes to the structure are duplexed.

The performance effect of implementing SM duplexing is more likely to be a lack of capacity in the CF than a transaction response time in the MVS CPU time component.

SM duplexing has the following additional costs:

- The MVS CPU cost
- The subchannel cost
- The link cost

There is no additional cost for simplex requests because your system already has the SM duplexing capability.

Guidelines:

- Evaluate your current environment regarding system managed duplexing. You might need to acquire more CF processor, storage, and link capacity resources to use system managed duplexing.
- If system managed duplexing is not possible, the preference list that you use when you define the structures in the IXCMIAPU utility must point to a set of Coupling Facilities. These facilities must be failure isolated from each other.

5.8.11 Recoverable VSAM data sets

Recovery affects performance because of the extra work that must be done to allow a cluster to be recoverable. If you do not have procedures in place and recovery products, such as CICS/VR, to implement, such as forward recovery (REDO), do not specify LOG(ALL). This option generates the processor load of creating record images for UNDO/REDO in the forward recovery log, but you cannot do anything with the information. For more information, see 5.4.9, “RLS Recoverable data sets” on page 237.

Guidelines: Do not declare your VSAM data set recoverable if you do not have products to enforce this recoverability.

5.8.12 Dealing with deadlocks

Deadlocks are a performance and problem determination issue. When in CICS RLS, messages display transactions that are timed out, which may be caused by VSAM RLS deadlocks.

You can try to avoid deadlocks by changing the way programs are written:

- Avoid locking more than one logical record concurrently.
- Create a hierarchy of locks. When more than one lock is required, have them required in a pre-determined sequence as defined by the hierarchy.

Your installation can activate the deadlock detection routine through the IGDSMSxx keywords DEADLOCK_DETECTION(nnnn|15,kkkk|4) and RLSTMOUT({nnnn|0}).

The default for DEADLOCK_DETECTION is (15,4). You might want to change this values to (1,1) to minimize the time that is required to detect a deadlock condition. Obviously there is a trade-off between processor time spent detecting deadlocks and the effect that deadlocks are causing to applications. Carefully monitor these effects in your environment before you change the supplied defaults.

You can get more information about what is causing a deadlock by issuing the following command:

```
DISPLAY SMS,SMSVSAM,DIAG(CONTENTION)
```

For more information about interpreting the output of this command, see 5.5.10, “Latch contention” on page 247.

Guidelines:

- Avoid deadlocks by enforcing the NRI option, implementing a lock hierarchy, or never asking for more than one lock concurrently.
- Activate the deadlock detection routine more often if you are facing several deadlock situations.

5.8.13 RLS performance gains

Performance gains are coming from data sharing and the use of CF:

- Balancing the system through dynamic workload distribution
- Using the CF cache structure as a buffer

Balancing the system through dynamic workload distribution

In CICS/RLS, the major performance objective (besides availability) is to balance the load between several MVS images that share data, and so provide better overall response time and throughput.

The queuing theory shows that two images at 80% and 20% busy have a much longer average queue time than two images at 50% busy each. In a controlled test environment, the VSAM "I/O time" (which includes the buffer and cache hits and the real I/O) might get worse (bigger) with RLS than without RLS. This performance degradation is because of CF processor usage caused by cross invalidation and locking. However, the caching function saves read access to DASD. Nevertheless, if the results are acceptable after tuning, in a production environment the performance will probably be better because of subsystems balance.

In a batch environment, throughput tends to be better because a much larger level of parallelism is obtained with RLS. Before RLS data sharing, ensuring integrity required that only one updater could open a cluster at a time. The serialization was at cluster level. With RLS the serialization is at record level and many updaters can access the same cluster at different logical records.

Using the CF cache structure as a buffer

Because of the better load balance of the systems accessing a VSAM cluster in RLS mode and the inexistence of the FOR bottleneck, a major rate of GET/PUT requests per second is expected. These GET/PUT requests can be served out of DASD, in local buffers, and in the cache. The concern is whether this large rate causes an increased number of DASD I/Os per second.

The number of DASD I/Os is dependent on the local buffer hit ratio. The buffer pool hit ratio is dependent on the type of workload and the read/write ratio. Remember that each write causes an I/O. To avoid growth in I/Os when adding systems, scale the buffer pool and coupling facility cache size to maintain a constant hit ratio in the buffer pool. For more information, see 5.8.4, "Local buffer pool sizing" on page 272 and 5.8.5, "RLS cache structures sizing" on page 275.

If the cache size is set up correctly for a read load, you should experience a drop in the DASD I/O rate per transaction, which improves the overall performance. If the Coupling Facility cache is sized large enough to avoid reclaims, the XIs caused by the updates will not cause an increase in the DASD I/O rate. This improvement is because the updated record can still be found in the CF. Because the RLS buffer manager manages its buffers more efficiently than local shared resource (LSR) pools, the buffer pool hit ratios are better for the RLS managed buffers.

5.8.14 CICSPlex RLS performance comparison

The paramount concern is what will happen with your transaction response time and throughput (number of transactions per second) when you change from a CICS/VSAM non-shared environment to a VSAM RLS.

As a general statement, running transactions under CICS TS in CICSPlex (multiple z/OS) accessing VSAM data sets in RLS mode does not degrade transaction response time. Experience has shown that response time remains the same and the total throughput is linear based on the number of images. That is, the throughput for three images is triple the throughput for one. These results are because of the following theoretical reasons:

- CICSPlex dynamic workload balance
- No more CICS function shipping
- No file owning region bottleneck
- Avoiding remote two-phase commit
- Avoiding I/O because of the CF cache structure

CICSPlex dynamic workload balance

Balancing the load across AORs in different z/OS images in a Parallel Sysplex is beneficial for CICS performance in general. For more information, see “Balancing the system through dynamic workload distribution” on page 285.

No more CICS function shipping

In the environment of CICSPlex without RLS, CICS transactions coming from the network are welcomed by a TOR address space. Based on WLM suggestion, they are sent to an AOR, where the transaction logic is run by a program. If such logic requires a VSAM logical record, through a CICS function shipping mechanism (through XCF, if the FOR is remote), the request is sent to a FOR address space. FOR is in charge of accessing non-RLS VSAM data sets on behalf of CICS transactions.

However, function shipping has a price. IBM benchmarks suggest an increase of about 16% to 20% in the processor time that is used by CICSPlex compared with just one system. This increase is because of function shipping between the AOR and the FOR.

With VSAM RLS, this processor load does not exist because AOR passes the control to SMSVSAM in its own system. This gain can counterbalance the MVS CPU cycles for accessing the CF.

No file owning region bottleneck

Without RLS, all CICS requests to the VSAM cluster flow through the CICS file-owning region (FOR). This address space is mono task. It can be a performance bottleneck, and a single point of failure for availability. With RLS, the FOR is replaced by several SMVSAM address spaces, one per system in the Parallel Sysplex. This new configuration addresses problems of availability.

Avoiding remote two-phase commit

Before SMSVSAM, it was possible for a unit of recovery to be formed by updates in VSAM data sets that were accessed by different CICS FOR address spaces. In this case, the sync point manager (CICS) must implement a remote two-phase commit. The commit was a lengthy process, causing a transaction response time to increase. With SMSVSAM, all the writes from the same recovery unit can be run in the same SMSVSAM, eliminating this performance problem.

Avoiding I/O because of the CF cache structure

The RLS cache structures in the CF can avoid many read I/O operations. For more information, see "Using the CF cache structure as a buffer" on page 285.

5.8.15 Using RLS mode in an ESDS organization

Usually the first implementations of RLS mode in customers are done in VSAM KSDS data sets. There are some concerns in using RLS in other VSAM data organizations such as ESDS, RRDS. It was tested by IBM, and works as described in the manuals.

5.9 RMF and VSAM RLS

There are several RMF reports that display the VSAM RLS performance such as Monitor III: RLSLRU, RLSSC, and RLSDS. There are also the traditional CF reports that describe the CF structures and their links. Use these reports for information about the SYSVSAM structures, such as IGWLOCK00 and the RLS cache structures:

- RMF III reports
- RMF Post Processor reports
- SMF records covering VSAM RLS

5.9.1 RMF III reports

The following RMF Monitor III reports contain key information that you can use for performance management of each field:

- VSAM RLS Monitor III reports: RLSLRU
- VSAM RLS Monitor III reports: RLSSC and RLSDS

VSAM RLS Monitor III reports: RLSLRU

The RLSLRU report provides Local Buffer Manager LRU statistics for each system. The data in this report can help you to adjust the goal / limit for the local buffer pool. Before going through this report, which is shown in Figure 5-41 on page 288, see 5.1.3, "How does RLS work?" on page 210.

The RLSLRU report has the following fields that need more explanation:

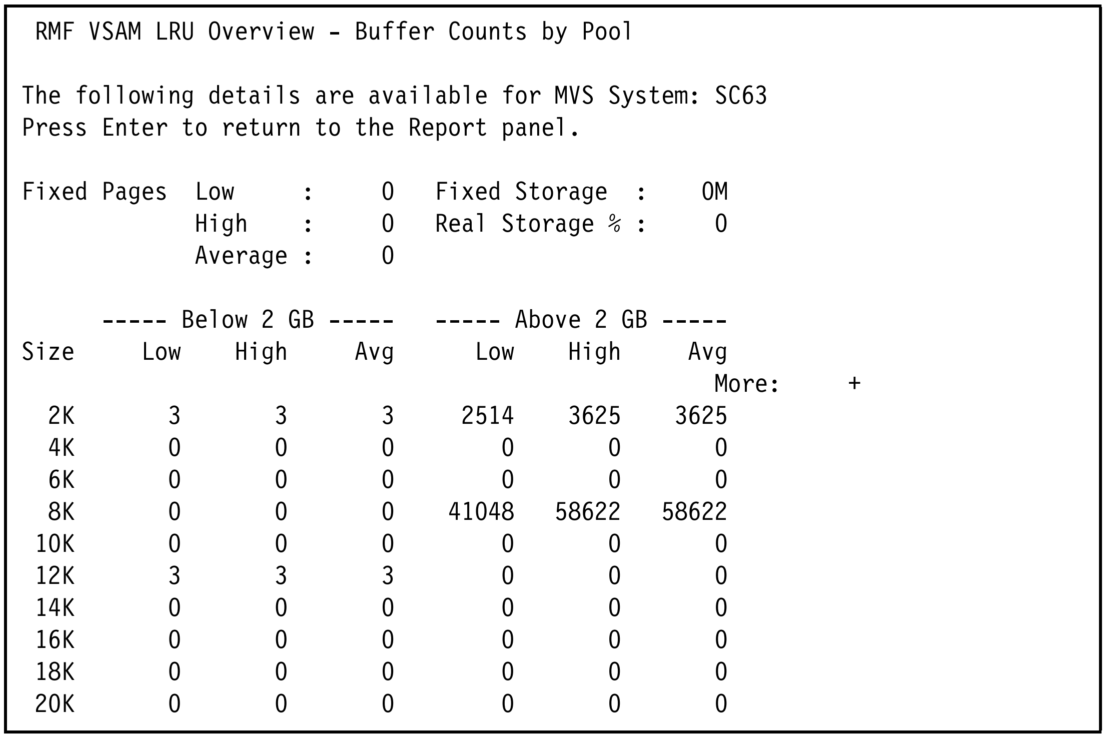
- *Avg CPU Time:* This is the average processor time that was spent by BMF LRU processing during each reporting interval in milliseconds. This field can be used for comparison between different RLS workloads. These workloads can be more or less buffer friendly, or produce more cross invalidation that causes a change in processor consumption.
- *Buffer Size Goal*: This is the buffer size goal (bytes) as stated in IGDSMSxx RLS_MAX_POOL_SIZE for 31-bit buffers, and RLSAboveTheBarMaxPoolsize for 64-bit buffers. If goal is greater than 1.5 GB, then the word MAX is displayed.
- *Buffer Size High:* This is the buffer size actual high value in bytes. A cursor-sensitive control on a system line displays a pop-up panel with buffer counts by pool for the selected system. There are 16 storage pools (2K, 4K,..., 32K) available. Each one is presented with the low, high, and average number of buffers. See Figure 5-42 on page 289 for an example, showing the local buffer pool distribution on system SC63.

- *Accel%:* This is the percentage of Buffer Manager LRU cycle routine when local buffer pool size is over the goal and buffer aging algorithms were accelerated.
- *Reclaim%:* This is the percentage of Buffer Manager LRU cycle routine when BMF was over the goal and buffer aging algorithms were bypassed to reclaim buffers.
- *BMF Read%:* This is the percentage of GET hits in local buffer pool when the CI is in local buffer pool and *valid*. Generally, you want this reading to be around 80% for direct. If the percentage is less than 80%, evaluate the level of cross invalidation. If the level of invalidation is OK, increase your local buffer pool maximum size. If you already have enough virtual storage, see the guidelines in "Control Interval cross invalidation" on page 276. Chances are that your workloads be cache unfriendly with revisits to the same data.
- *Cache Read%:* This is the percentage of GET hits in the CF cache structure.
- *DASD Read%*: This is the percentage of GET I/O requests in DASD.

```
RMF V1R12  VSAM LRU Overview  - SANDBOX        Line 1 of 12
Command ===>                                                Scroll ===> CSR

Samples: 120     Systems: 4    Date: 05/18/11  Time: 19.33.00  Range: 120   Sec

MVS         Avg CPU  - Buffer Size -  Accel  Reclaim  ------ Read -----
System       Time      Goal   High      %       %      BMF%   CF%  DASD%

SC63
 Below 2GB   0.306     100M    22M     0.0     0.0     0.0   0.0    0.0
 Above 2GB   0.129     500M   248M     0.0     0.0    80.3   2.4   17.2
SC64
 Below 2GB   0.030     100M     0M     0.0     0.0     0.0   0.0    0.0
 Above 2GB   0.020     500M     0M     0.0     0.0     0.0   0.0    0.0
SC65
 Below 2GB   0.031     100M     0M     0.0     0.0     0.0   0.0    0.0
 Above 2GB   0.030     500M     0M     0.0     0.0     0.0   0.0    0.0
SC70
 Below 2GB   0.029     100M     8M     0.0     0.0     0.0   0.0    0.0
 Above 2GB   55.49     500M   404M     0.0     0.0    85.5   2.3   12.2
```

Figure 5-41 RLSLRU report

Figure 5-42 shows a sample RLSLRU report for a 64-bit local buffer pool.



```
 RMF VSAM LRU Overview - Buffer Counts by Pool

The following details are available for MVS System: SC63
Press Enter to return to the Report panel.

Fixed Pages  Low     :     0   Fixed Storage  :    0M
             High    :     0   Real Storage % :     0
             Average :     0

      ----- Below 2 GB -----   ----- Above 2 GB -----
Size     Low    High     Avg      Low    High     Avg
                                                    More:     +
  2K       3       3       3     2514    3625    3625
  4K       0       0       0        0       0       0
  6K       0       0       0        0       0       0
  8K       0       0       0    41048   58622   58622
 10K       0       0       0        0       0       0
 12K       3       3       3        0       0       0
 14K       0       0       0        0       0       0
 16K       0       0       0        0       0       0
 18K       0       0       0        0       0       0
 20K       0       0       0        0       0       0
```

Figure 5-42 64-bit local buffer pool usage by system SC63

VSAM RLS Monitor III reports: RLSSC and RLSDS

RLSSC and RLSDS reports show the same fields. The difference is the scope. RLSSC groups data in storage classes and RLSDS groups it in VSAM data sets. Both reports cover the use of local buffers and CF cache.

To gather the data about data sets, the option VSAMRLS must be set in Monitor III ERBRMFxx Parmlib member:

```
VSAMRLS ADD(data set name mask)
DELETE(data set name mask)
```

This option controls the collection of VSAM RLS activity data. When you specify VSAMRLS or allow the default value to take effect, activity data is gathered for VSAM RLS by storage class. In addition, data set masks can be specified to collect data by VSAM data sets as well. To suppress the gathering of VSAM RLS data, specify NOVSAMRLS.

The collection of VSAM RLS activity data by VSAM data sets can be controlled by following suboptions:

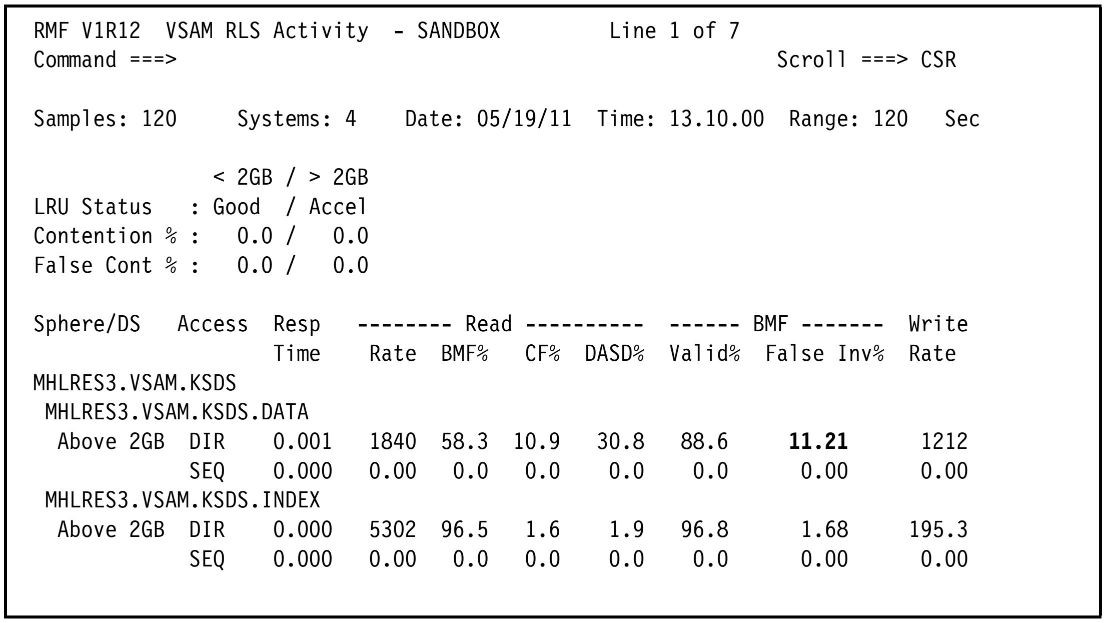
- ADD: Start collection for all VSAM data sets that are covered by the mask.
- DEL: Stop collection for all VSAM data sets that are covered by the mask.

If you stop and start SMSVSAM, RMF Monitor III Data gatherer (RMGAT) must be restarted to capture VSAM RLS data again.

These reports provide VSAM RLS activity about GET and PUT requests accessing the local buffers, the CF cache structures, and DASD. These data are useful to answer questions such as these:

- Are there problems with Least Recently Used algorithms (LRU) or buffer pool sizes?
- Are the CF cache structures too small?

Figure 5-43 shows a sample RLSDS report. See Figure 5-39 on page 278 for a sample RLSSC report.



```
RMF V1R12  VSAM RLS Activity  - SANDBOX        Line 1 of 7
Command ===>                                                Scroll ===> CSR

Samples: 120     Systems: 4    Date: 05/19/11  Time: 13.10.00  Range: 120   Sec

               < 2GB / > 2GB
LRU Status   : Good  / Accel
Contention % :   0.0 /   0.0
False Cont % :   0.0 /   0.0

Sphere/DS   Access  Resp   -------- Read ----------  ------ BMF -------  Write
                    Time    Rate  BMF%   CF%  DASD%  Valid%  False Inv%  Rate
MHLRES3.VSAM.KSDS
 MHLRES3.VSAM.KSDS.DATA
  Above 2GB  DIR    0.001   1840  58.3  10.9   30.8   88.6      11.21      1212
             SEQ    0.000   0.00   0.0   0.0    0.0    0.0       0.00      0.00
 MHLRES3.VSAM.KSDS.INDEX
  Above 2GB  DIR    0.000   5302  96.5   1.6    1.9   96.8       1.68     195.3
             SEQ    0.000   0.00   0.0   0.0    0.0    0.0       0.00      0.00
```

Figure 5-43 RLSDS report

The report that is shown in Figure 5-43 has these fields that need more explanation:

- *LRU status:* This indicates the status of local buffers, both the 31-bit buffer pool (< 2 GB) and the 64-bit buffer pool (> 2 GB). It can have one of these statuses:
  - *Good*: The local buffer pool size is below its goal on all systems.
  - *Accelerated*: The local buffer pool size is over the goal on at least one system, and the buffer aging algorithms are accelerated.
  - *Reclaimed*: The local buffer pool size is over the goal on at least one system, and the buffer aging algorithms were bypassed to reclaim buffers. This status is the same as accelerated, but much more dramatic.
- *Contention%:* This is the percentage of true lock contentions. It is the percentage of requests that wee issued by exploiters that were delayed because of real contention on a lock in relation to the total number of requests. Generally, you want this percentage to be less than 2.0%.
- *False Contention%:* This is the percentage of false LOCK contentions. It is the percentage of requests that were issued by exploiters that were delayed because of false contention on a lock in relation to the total number of requests. Generally, you want this percentage to be under 0.5%. If you see a greater percentage on your installation, consider increasing the size of the RLS CF lock structures
- *Resp Time:* This is the average I/O response time in seconds of all I/O requests generated by the line portrayed entity, such as storage class, data set, and system (cache structure). It includes IOSQ time, pending time, disconnect, and connect time.

- *Read:* This describes the behavior of GET I/O operations. It displays the rate and the distribution of where the GET is served, which can be the local buffer pool, SMSVSAM cache structure in the CF, or DASD. This field has the following subheadings:
  - Rate: This is the average number of logical records read through a GET, per second.
  - BMF%: This is the percentage of GET hits in a local buffer pool, when the CI is in local buffer pool and *valid*. Sometimes the CI is invalid because another logical record (not the one required) was updated by another SMSVSAM. Generally, you want this percentage to be around 80% for direct. If it is less than this, evaluate the level of cross invalidation. If that figure is normal, increase your local buffer pool maximum size if you have enough virtual storage.
  - CF%: This is the percentage of GET hits in the CF cache structure.
  - DASD%: This is the percentage of GET I/O requests in DASD.
- *BMF:*
  - *VALID%:* This is the percentage of BMF GETs hits that were valid. If a buffer is found in the local cache and determined to be valid according to the information in local control blocks, it counts as a BMF valid READ hit.
  - *FALSE INV%:* This is the percentage of READ requests when the copy in the BMF local cache was invalid. The copy is invalid because the coupling facility lost track of the integrity status of the buffer. In other words, there is no copy of the CI being referenced in a CF cache structure. The example in Figure 5-43 on page 290 shows a high value for BMF false invalids. This level happened because the size of the local buffer pools was increased, but the RLS CF cache structure remained at its original size.
- *WRITE RATE:* This is the total number of BMF PUTs requests per second.

5.9.2 RMF Post Processor reports

These reports are created by RMF Monitor III data gatherer in an SMF record format. Later, these records are processed by the Postprocessor, generating the sysout reports. There are also CF reports online that are produced by RMF Monitor III data gatherer and shown under TSO Monitor III data reporter. These reports are not covered here because their fields are a repetition of the Post Processor fields.

The CF Postprocessor reports are generic, showing the performance aspects of all the CFs and all the structures. This section covers only fields and examples of RLS structures. The example reports are modified to highlight these structures. For a full description of all fields, see *OS/390 MVS Parallel Sysplex Capacity Planning*, SG24-4680.

Coupling Facility Usage Summary report

This report shows the use of the CF in number of requests, storage, and processor. You can use this section to evaluate the status of all structures and how the CF as a whole is being used. See Figure 5-44 on page 292.

Structure summary

The structure summary lists all structures, listed by type and name. For each structure, the status at the end of the RMF data collection interval is presented:

- ACTIVE: Storage is allocated and there is connection to at least one system. There is also information about duplexing such as primary (PRIM) or secondary (SEC).
- INACTV: Storage is allocated but there is no connection to any system. These structures are called persistent. IGWLOCK00 is persistent.

- UNALLOC: The structure was active at some point in time during the interval, but did not exist at the end of the RMF interval.

```
C O U P L I N G   F A C I L I T Y   A C T I V I T Y
                                                                                                                     PAGE   1
      z/OS V1R12            SYSPLEX SANDBOX           DATE 05/18/2011          INTERVAL 010.00.000
                            RPT VERSION V1R12 RMF     TIME 18.30.00            CYCLE 01.000 SECONDS

   ------------------------------------------------------------------------------------------------------------------------------
    COUPLING FACILITY NAME = CF2
    TOTAL SAMPLES(AVG) =   600  (MAX) =   600  (MIN) =   600
   ------------------------------------------------------------------------------------------------------------------------------
                                              COUPLING  FACILITY  USAGE  SUMMARY
   ------------------------------------------------------------------------------------------------------------------------------
    STRUCTURE SUMMARY
   ------------------------------------------------------------------------------------------------------------------------------

                                                  % OF             % OF    % OF     AVG     LST/DIR  DATA      LOCK      DIR REC/
           STRUCTURE                       ALLOC  CF           #   ALL     CF       REQ/    ENTRIES  ELEMENTS  ENTRIES   DIR REC
    TYPE   NAME              STATUS CHG    SIZE   STOR       REQ   REQ     UTIL     SEC     TOT/CUR  TOT/CUR   TOT/CUR   XI'S


    LOCK   RLSLOCK_TEST00    ACTIVE          4M    0.2     1104K   41.1    23.0   1839.6      10K        0        262K       N/A
                                                                                                1        0          3       N/A

    CACHE  DB9CG_GBP0        ACTIVE          8M    0.4       129    0.0     0.0     0.21    6008      1201        N/A         0
                                                                                               15        5        N/A         0
           DB9DG_GBP0        ACTIVE          8M    0.4       148    0.0     0.0     0.25    6008      1201        N/A         0
                                                                                               65        5        N/A         0
           DB9DG_GBP8K0      ACTIVE          8M    0.4        74    0.0     0.0     0.12    3339      1334        N/A         0
                                                                                                0        0        N/A         0
           RLS_CACHE         ACTIVE         50M    2.6     1559K   58.0    73.0   2597.9      83K      11K        N/A       86K
                                                                                              82K      10K        N/A       84K
           SYSZWLM_991E2094  ACTIVE         12M    0.6       716    0.0     0.1     1.19    1052      2093        N/A         0
                                                                                                4       24        N/A         0

    PROCESSOR SUMMARY
   ------------------------------------------------------------------------------------------------------------------------------

    COUPLING FACILITY         2094       MODEL S18      CFLEVEL  15          DYNDISP OFF

    AVERAGE CF UTILIZATION  (% BUSY)            3.2      LOGICAL PROCESSORS:   DEFINED   1    EFFECTIVE    1.0
                                                                               SHARED    0    AVG WEIGHT   0.0
```

Figure 5-44 CF usage summary report

For each active structure, the section shows the following data:

- Allocated structure size
- The total number of requests issued by the exploiter to the XES API for the structure
- The average number of requests/second

The report also has four final columns. The information in these columns can be used to validate the size of various types of structures. The TOT/CUR (total/current) in-use values indicate how many are allocated (TOT) and how many are in use (CUR) at end of the RMF interval. Therefore, it is not a high water mark.

- Cache structures (as RLS_CACHE)
  - *DIR ENTRIES* displays how many cache directory entries are allocated/in-use. The example report shows that the RLS_CACHE directory is almost full with 82K entries in use, from a total of 83K.
  - *DATA ELEMENTS* displays how many cache elements (data) are allocated/in-use. In the example, this is 10K, from a total of 11K.
  - The *DIR RECLAIMS* can be used as an indicator whether the directory of a cache structure is over-committed. Whenever a shortage of directory entries occurs, the CF reclaims in-use directory entries that are associated with unchanged data. These reclaimed items are used to satisfy new requests for directory entries for SMSVSAM.

    All users of the data item that is represented by the directory entry must be notified that their copy of the data item is invalid. When the SMSVSAM needs access to the now invalidated data item, the item must be reread from DASD and registered with the CF.

  Avoid directory reclaims because this activity results in increased I/O activity to the cluster to reacquire a referenced data item. Reacquiring data items and re-registering them to the CF can result in increased processor utilization and prolonged transaction response times whenever a read miss occurs in the local buffer pool.

  Directory reclaims can be eliminated by increasing the number of directory entries for a particular structure. You can do so by using these techniques:

  - Increasing the size of the structure. For structures such as IGWLOCK00 with data elements and directory entries, both increases in the ratio are specified by SMSVSAM.
  - Changing the proportion of the structure. Some cache structure exploiters such as DB2 and IMS allow the installation to specify the ratio of directory elements to data elements. IGWLOCK00 does not have this capability. It constantly adjusts the data-to-directory ratio to minimize the number of buffer invalidations.

- Lock structures (IGWLOCK00)

  LST ENTRIES displays how many lock lists that point to the lock data are allocated/in-use. In the report, you can see that the RLSLOCK_TEST00 secondary lock structure is almost empty, with 10K allocated and only 1 used.

  LOCK ENTRIES contains record data entries. In the report, you can see that the RLSLOCK_TEST00 has 262K record data that are allocated, and only 3 in use.

  However, the critical data about lock structures can be found in the FALSE CONTENTION data for the lock structure in “Coupling Facility structure activity report” on page 293.

Processor summary

The processor summary section display the following data:

- The CF model and version.
- Average CF Utilization: This indicates the real CP utilization (discarding the CFCC ethernal loop effect) of the LP that hosts the CF.
- Logical processors:
  - Defined: How many logical processors are defined to this CF.
  - Effective: The number of effective available logical processors in a shared environment. CFCC measures the time of real command execution and the time spent waiting for work. The reported value shows the ratio between the LPAR dispatch time (CFCC execute and loop time) and the RMF interval length.

Coupling Facility structure activity report

The structure activity report, an example of which is shown in Figure 5-45 on page 294, presents these data for all structures:

- The REQUESTS columns list the number of synchronous (SYNC), asynchronous (ASYNC), and changed (CHNGD) requests. For more information, see “CF synchronous and asynchronous requests” on page 271. It also lists the average service times and standard deviation for these requests (SERV TIME(MIC)). The CHNGD line applies to the synch requests changed to asynch because the subchannel was busy.

  The service time average of 4.4 microseconds shown in the example report indicates a fast CF hardware.

- The DELAYED REQUESTS columns list the number of requests that were delayed because of the following conditions:
  - NO SCH: Subchannel contention. Lists the number of *asynchronous* requests that were delayed because of subchannel busy conditions.
  - PR WT: Peer subchannel wait contention. Applies to duplexed requests, which always require two subchannels. The number of requests a subchannel for the operation that is targeted to the peer (secondary) structure is not available. Only those in the primary structure are listed. This amount is always about 100% for secondary monitor delay times.
  - PR CMP: Waiting-for-peer-completion contention. Applies to duplexed requests, which always require two subchannels. This is the number of requests that one of the two duplexed operations has completed, but the completed subchannel remains unavailable for use until the peer operation completes. It shows disparity in response times of CFs.
  - DUMP (not shown in this example): Lists the items that are waiting for the DUMP structure serialization.
- The /DEL field indicates the average delay time for each delayed request. The /ALL field indicates the average delay time, taking in consideration all the requests, including the ones that do not suffer delay.
- The EXTERNAL REQUEST CONTENTIONS gives information about lock contention for serialized list and lock structures, as well as data access statistics for cache structures. A discussion for the SMSVSAM structures follows.

```
C O U P L I N G   F A C I L I T Y   A C T I V I T Y
                                                                                                                       PAGE   9
        z/OS V1R12            SYSPLEX SANDBOX            DATE 05/18/2011          INTERVAL 010.00.000
                              RPT VERSION V1R12 RMF      TIME 18.30.00            CYCLE 01.000 SECONDS

    ------------------------------------------------------------------------------------------------------------------------------
     COUPLING FACILITY NAME = CF2
    ------------------------------------------------------------------------------------------------------------------------------
                                               COUPLING  FACILITY  STRUCTURE  ACTIVITY
    ------------------------------------------------------------------------------------------------------------------------------

     STRUCTURE NAME = RLSLOCK_TEST00    TYPE = LOCK   STATUS = ACTIVE
               # REQ    -------------- REQUESTS -------------   -------------- DELAYED REQUESTS -------------
     SYSTEM    TOTAL            #     % OF  -SERV TIME(MIC)-    REASON    #    % OF  ---- AVG TIME(MIC) -----   EXTERNAL REQUEST
     NAME      AVG/SEC         REQ    ALL     AVG   STD_DEV              REQ    REQ   /DEL    STD_DEV   /ALL    CONTENTIONS

     SC63        249K   SYNC   249K   22.6     4.6      6.2     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL      243K
                415.0   ASYNC    0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0   REQ DEFERRED  5540
                        CHNGD    0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0    0.0        0.0    0.0   -CONT         5540
                                                                                                             -FALSE CONT    870

     SC65           0   SYNC     0     0.0     0.0      0.0     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL        0
                 0.00   ASYNC    0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0   REQ DEFERRED     0
                        CHNGD    0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0    0.0        0.0    0.0   -CONT            0
                                                                                                             -FALSE CONT      0

     SC70        855K   SYNC   855K   77.4     4.3      4.3     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL      849K
                 1425   ASYNC    0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0   REQ DEFERRED  5501
                        CHNGD    0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0    0.0        0.0    0.0   -CONT         5501
                                                                                                             -FALSE CONT   1042

    ------------------------------------------------------------------------------------------------------------------------------
     TOTAL       1104K  SYNC  1104K    100     4.4      4.8     NO SCH    0    0.0    0.0        0.0    0.0   REQ TOTAL     1092K
                 1840   ASYNC    0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0   REQ DEFERRED    11K
                        CHNGD    0     0.0                      PR CMP    0    0.0    0.0        0.0    0.0   -CONT           11K
                                                                                                             -FALSE CONT   1912
```

Figure 5-45 CF structure activity report (RLSLOCK_TEST00)

Lock Structures contentions

The RMF reports these data:

- REQ TOTAL, which is the number of requests issued by SMSVSAM to the XES API for the RLS lock structures
- REQ DEFERRED, which is the number of requests deferred because of contention

REQ TOTAL should be numerically close to # REQ in the same report, the difference is explained by the way XES account requests for unlocking.

A request can be deferred because of:

- True lock contention, which is measured by (CONT - FALSE CONT).
- False lock contention, which is measured by FALSE CONT.
- XES internal processing delays. If REQ DEFERRED is equal to CONT, there is no such type of delay.

As a result of lock contention you might see increased XCF activity that is caused by SMSVSAMs negotiating the lock status.

RLSLOCK_TEST00 shows 1% (11K/1092K) for true lock contention. This is an acceptable true contention, as the rule is to keep it under 2 percent. True contention is application-dependent, so you must identify the workload that uses the lock structure. For example, long-running batch jobs that do not commit the locks can cause high true contention on lock structures.

The false lock contention is slightly more than 0.1% (1912/1092K) of the total requests. The best value for IGWLOCK00 is under 0.5 percent.

Cache Structures for Data Access

For this analysis, in Figure 5-46 on page 296, review the data about the RLS_CACHE structure. The service time for this structure is around 8 microseconds, which indicates a fast CF hardware.

DATA ACCESS statistics for cache structures are acquired from counters in the CF. They cannot be broken down into individual systems. Only SMSVSAM knows how efficiently the local buffer and cache structure are being used. Depending on the cache structure implementation, one or more counts can be zero.

- READS and WRITES indicate how often the CFCC returned data to SMSVSAM (read) and how often SMSVSAM placed changed data into the RLS_CACHE structure.
- CASTOUTS is a count of the number of times an explorer (demanded by CFCC) retrieves a changed data entry. This process writes the data to DASD, causing the changed attribute to be reset to unchanged in the structure. This process occurs when the structure occupancy reaches a CFCC internal threshold. Because SMSVSAM uses a store-through algorithm, no DASD castouts are needed to make room in the structure.
- XIs shows the number of times a data item in a local buffer pool was marked invalid by the CF during the RMF interval. The counter reflects the amount of data sharing among the users of the cache structure, and the amount of write/update activity against the databases. For more information, see “Control Interval cross invalidation” on page 276.

```
STRUCTURE NAME = RLS_CACHE        TYPE = CACHE  STATUS = ACTIVE
              # REQ    -------------- REQUESTS -------------   -------------- DELAYED REQUESTS -------------
    SYSTEM    TOTAL            #     % OF  -SERV TIME(MIC)-    REASON    #   % OF  ---- AVG TIME(MIC) -----
    NAME      AVG/SEC         REQ    ALL     AVG   STD_DEV              REQ    REQ   /DEL    STD_DEV   /ALL

    SC63        302K  SYNC    302K   19.4     8.0      5.2     NO SCH    0    0.0    0.0        0.0    0.0
               503.4  ASYNC     0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0
                      CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0    0.0        0.0    0.0
                                                               DUMP      0    0.0    0.0        0.0    0.0

    SC65           0  SYNC      0     0.0     0.0      0.0     NO SCH    0    0.0    0.0        0.0    0.0
                0.00  ASYNC     0     0.0     0.0      0.0     PR WT     0    0.0    0.0        0.0    0.0
                      CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0    0.0        0.0    0.0
                                                               DUMP      0    0.0    0.0        0.0    0.0

    SC70       1257K  SYNC   1257K   80.6     7.6      4.1     NO SCH    0    0.0    0.0        0.0    0.0
                2094  ASYNC     1     0.0   155.0      0.0     PR WT     0    0.0    0.0        0.0    0.0
                      CHNGD     0     0.0  INCLUDED IN ASYNC   PR CMP    0    0.0    0.0        0.0    0.0
                                                               DUMP      0    0.0    0.0        0.0    0.0

   -------------------------------------------------------------------------------------------------------------------------------
    TOTAL      1559K  SYNC   1559K    100     7.6      4.4     NO SCH    0    0.0    0.0        0.0    0.0    -- DATA ACCESS ---
                2598  ASYNC     1     0.0   155.0      0.0     PR WT     0    0.0    0.0        0.0    0.0    READS       72923
                      CHNGD     0     0.0                      PR CMP    0    0.0    0.0        0.0    0.0    WRITES     747501
                                                               DUMP      0    0.0    0.0        0.0    0.0    CASTOUTS        0
                                                                                                             XI'S       152117
```

Figure 5-46 CF Structure Activity report (RLS_CACHE)

Subchannel Activity

This report is presented in Figure 5-47 on page 297. To review, a CF request is first processed by XES. It is then transferred to the CF if there is a free link, processed by a CF CP, and sent back to z/OS through the CF link. This report addresses the possible delays for those requests.

Each CF link in peer mode has seven subchannels, and seven buffers, for each image that it is serving (MIF). Subchannels are busy from the time MVS sends the request until the time MVS processes the response. Links are only busy for the time it takes for the data to travel between CPCs.

CF Links do not support CPMF, so there is no reporting anywhere of actual CF link utilization. Actual link utilization is rarely an issue.

The data that are used to build this report are not available on a per structure basis. Some of the fields have the same meaning as the ones found in the report that is shown in Figure 5-46. The following are the fields that are different:

- PTH BUSY: Measures the number of times that a synchronous request gets a free subchannel but encounters the all path busy condition. This situation can occur because the CF link is shared (MIF) between several z/OS images. The request “spins” until the link becomes available. If this figure is much greater than zero, dedicate the CF link to the LP. The number is the example is zero, which is ideal.
- DELAYED REQUESTS: These columns have the same meaning of the CF Structure report, but are organized by the type of the structures.

```
C O U P L I N G   F A C I L I T Y   A C T I V I T Y
                                                                                                                                   PAGE  13
        z/OS V1R12              SYSPLEX SANDBOX             DATE 05/18/2011            INTERVAL 010.00.000
                                RPT VERSION V1R12 RMF       TIME 18.30.00              CYCLE 01.000 SECONDS

   ------------------------------------------------------------------------------------------------------------------------------
    COUPLING FACILITY NAME = CF2
   ------------------------------------------------------------------------------------------------------------------------------
                                                          SUBCHANNEL  ACTIVITY
   ------------------------------------------------------------------------------------------------------------------------------
             # REQ                                    ----------- REQUESTS -----------   ------------------ DELAYED REQUESTS --------------
    SYSTEM   TOTAL   -- CF LINKS --   PTH                  #    -SERVICE TIME(MIC)-                #     % OF   ------ AVG TIME(MIC) ------
    NAME     AVG/SEC TYPE  GEN  USE   BUSY                REQ     AVG     STD_DEV                 REQ    REQ    /DEL    STD_DEV     /ALL

    SC63     570743  ICP     2    2      0   SYNC    551851     6.5        5.9   LIST/CACHE    0   0.0    0.0        0.0       0.0
              951.2  SUBCH  14   14          ASYNC    11722   540.0      28990   LOCK          0   0.0    0.0        0.0       0.0
                                             CHANGED      0  INCLUDED IN ASYNC   TOTAL         0   0.0
                                             UNSUCC       0     0.0        0.0
    SC65       5695  ICP     2    2      0   SYNC       484    15.4       10.9   LIST/CACHE    0   0.0    0.0        0.0       0.0
                9.5  SUBCH  14   14          ASYNC     2470   931.9      816.6   LOCK          0   0.0    0.0        0.0       0.0
                                             CHANGED      0  INCLUDED IN ASYNC   TOTAL         0   0.0
                                             UNSUCC       0     0.0        0.0
    SC70      2136K  ICP     2    2      0   SYNC     2112K     6.3        4.5   LIST/CACHE    0   0.0    0.0        0.0       0.0
             3559.7  SUBCH  14   14          ASYNC    10012   322.4      890.4   LOCK          0   0.0    0.0        0.0       0.0
                                             CHANGED      0  INCLUDED IN ASYNC   TOTAL         0   0.0
                                             UNSUCC       0     0.0        0.0
```

Figure 5-47 Subchannel activity

CF to CF Activity

This report is shown in Figure 5-48. This report is introduced because of SM duplexing. In this type of duplexing, there is a conversation between the two CFs. The fields that are pictured in the report show details of the conversation. All of these fields are already covered in this topic in the other reports.

```
C O U P L I N G   F A C I L I T Y   A C T I V I T Y
                                                                                                                                   PAGE  14
        z/OS V1R12              SYSPLEX SANDBOX             DATE 05/18/2011            INTERVAL 010.00.000
                                RPT VERSION V1R12 RMF       TIME 18.20.00              CYCLE 01.000 SECONDS

   ------------------------------------------------------------------------------------------------------------------------------
    COUPLING FACILITY NAME = CF2
   ------------------------------------------------------------------------------------------------------------------------------
                                                           CF TO CF ACTIVITY
   ------------------------------------------------------------------------------------------------------------------------------
             # REQ                                        ----------- REQUESTS -----------   -------------- DELAYED REQUESTS --------------
    PEER     TOTAL      -- CF LINKS --                        #    -SERVICE TIME(MIC)-          #     % OF   ------ AVG TIME(MIC) ------
    CF       AVG/SEC    TYPE       USE                       REQ     AVG    STD_DEV            REQ    REQ    /DEL    STD_DEV     /ALL

    CF1         430     ICP          2        SYNC     430     0.1        0.0    SYNC     0    0.0    0.0        0.0       0.0
                0.7
```

Figure 5-48 CF to CF report

XCF Activity Report (XCF Usage by Member)

This report shows the communication activity between XCF group members on different systems.

An XCF group is a set of related members that a multisystem application defines to XCF. A member in one system can communicate with other members of the same group in another system in the sysplex.

You can think of the SMSVSAM functions that use an RLS lock structure as members of an XCF group. To find out what Group Name is assigned to an exploiter of a lock structure, issue this command:

```
D XCF,STR,STRNAME(structurename)
```

Figure 5-30 on page 255 shows the command output, which you can use to find what group name (GRPNAME) is using it:

```
XCF GRPNAME    : IXCL0001
```

You can use this report to monitor the communication between systems in the sysplex regarding the utilization of an RLS lock structure. For example, the SMSVSAM running in SC63 sends a 2206 request to SMSVSAM running in SC70 (both in IXCLO001 group) as shown in Figure 5-49.

```
 z/OS V1R12               SYSTEM ID SC63             DATE 05/18/2011            INTERVAL 10.00.000
                                      RPT VERSION V1R12 RMF      TIME 18.40.00              CYCLE 1.000 SECONDS
-
                                                        XCF USAGE BY MEMBER
 -------------------------------------------------------------------------------------------------------------------------------------
0               MEMBERS COMMUNICATING WITH SC63                                                   MEMBERS ON SC63
 ------------------------------------------------------------------           --------------------------------------------------------
0                                                  REQ          REQ
                                                  FROM           TO                                                   REQ          REQ
 GROUP       MEMBER              SYSTEM           SC63         SC63           GROUP       MEMBER                      OUT           IN
0IXCL0001    M479                SC64                0            0           IXCL0001    M480                      2,066        3,385
             M481                SC65                0            0                                            ----------   ----------
             M482                SC70            2,066        3,385           TOTAL                             2,066        3,385
                                            ----------   ----------
 TOTAL                                           2,066        3,385
0IXCL0004    M444                SC64               24           34           IXCL0004    M445                         40           51
             M446                SC65                8            9                                            ----------   ----------
             M447                SC70                0            0           TOTAL                                    40           51
                                            ----------   ----------
 TOTAL                                              32           43
0IXCL0005    M583                SC64                0            0           IXCL0005    M584                          0            0
             M585                SC65                0            0                                            ----------   ----------
             M586                SC70                0            0           TOTAL                                     0            0
                                            ----------   ----------
 TOTAL                                               0            0
0IXCL0013    M217                SC65              103           79           IXCL0013    M216                        320          277
             M218                SC70               53           34                                            ----------   ----------
                                            ----------   ----------           TOTAL                                   320          277
 TOTAL                                             156          113
0                                                                             IXCL002F    M79                           0            0
                                                                                                               ----------   ----------
                                                                              TOTAL                                     0            0
```

Figure 5-49 XCF Activity report

5.9.3 SMF records covering VSAM RLS

There are two SMF records that cover VSAM RLS statistical performance data: SMF record type 42 and 64. This section just describes the fields that are associated with VSAM RLS.

SMF record 42

This record can be created on a timed interval or whenever the SMF timer ends. You can specify SMF_TIME in IGDSMSxx to synchronize SMF type 42 data with SMF and RMF data intervals. SMF record type 42 has many subtypes, some of which have RLS cache structure statistics. This data includes information for each system and a sysplex-wide summary:

- Subtype 15 collects RLS statistics by Storage Class
- Subtype 16 collects RLS statistics by Data Set
  - You must activate collection of subtype 16 statistics by issuing this command:

    `V SMS,MONDS(`*`datasetname or mask`*`),ON`
  - To be able to see the RMF III RLSDS reports, issue this command:

    `F RMF,F III,VSAMRLS(ADD(`*`dsname.`*`**))`

    Do not forget to update the corresponding ERBRMFxx parmlib member if you want to continuously monitor your VSAM RLS data sets.

- Subtype 17 collects data about coupling facility lock structure usage
- Subtype 18 collects data about coupling facility cache partition usage
- Subtype 19 collects BMF statistics

Only one system in the sysplex collects the SMF 42 records. You can discover what system is collecting the records in the output of command D SMS,SMSVSAM as shown in Figure 5-29 on page 253).

SMF record 64

SMF writes a record type 64 when a cluster is closed or processed by EOV. SMF writes one record for each component in the cluster. SMF record type 64 has no subtypes. New sections were added to give CF cache information, and some fields were modified to reflect RLS use of the cluster or component.

Here are some fields of type 64 records that you can use to monitor activity against your data sets:

SMF64DLR	Number of logical records
SMF64DDE	Number of delete requests
SMF64DIN	Number of insert requests
SMF64DUP	Number of update requests
SMF64DRE	Number of retrieve requests
SMF64BMH	Number of BMF hits in the local buffer pool
SMF64CFH	Number of CF hits in the RLS cache structure
SMF64RIO	Number of requests that are read from DASD
SMF64DEP	Total number of requests
SMF64NLR	Number of logical records at open

For more information about SMF records, see *z/OS MVS System Management Facilities (SMF)*, SA22-7630.

5.10 VSAM and high availability

Many large z/OS installations run critical applications that access information about VSAM data sets. Those applications require a high level of availability.

VSAM RLS is a requirement for applications that demand high levels of availability for VSAM data. It allows you to have several CICS regions and batch applications that access the same VSAM data sets for read/write, while maintaining data integrity.

This section addresses some improvements that you can implement to achieve the highest levels of availability for your applications that use VSAM data sets, along with the related guidelines.

5.10.1 Using multiple CFs

Use multiple CFs with global connectivity to ensure maximum availability, simplify systems management, and allow for non-disruptive lock transfer and cache rebuild in the event of a CF outage.

You must have at least one CF connected to all systems capable of VSAM RLS within the Parallel Sysplex. For lock structure duplexing, you must have at least two CFs. For multiple CFs, select one facility with global connectivity to contain the master lock structure. For maximum availability, use a second CF with global connectivity. If a CF becomes unavailable, the SMSVSAM address space can transfer its in-storage duplexed copy of the locks to the other available CF without causing any application disruption.

Figure 5-50 illustrates this scenario:

1. A lock structure was initially allocated on CF A, the preferred CF, as it was defined by the IXCMIAPU run.
2. A failure on CF A occurs.
3. The SMSVSAM address spaces try to rebuild the lock structure in CF B by using the information in the address spaces.
4. If rebuild is successful, recovery is complete. Applications continue to run without disruption. If SMSVSAM cannot rebuild the lock structure, the applications experience an outage. All locks for recoverable VSAM data sets are marked as “lost locks”.

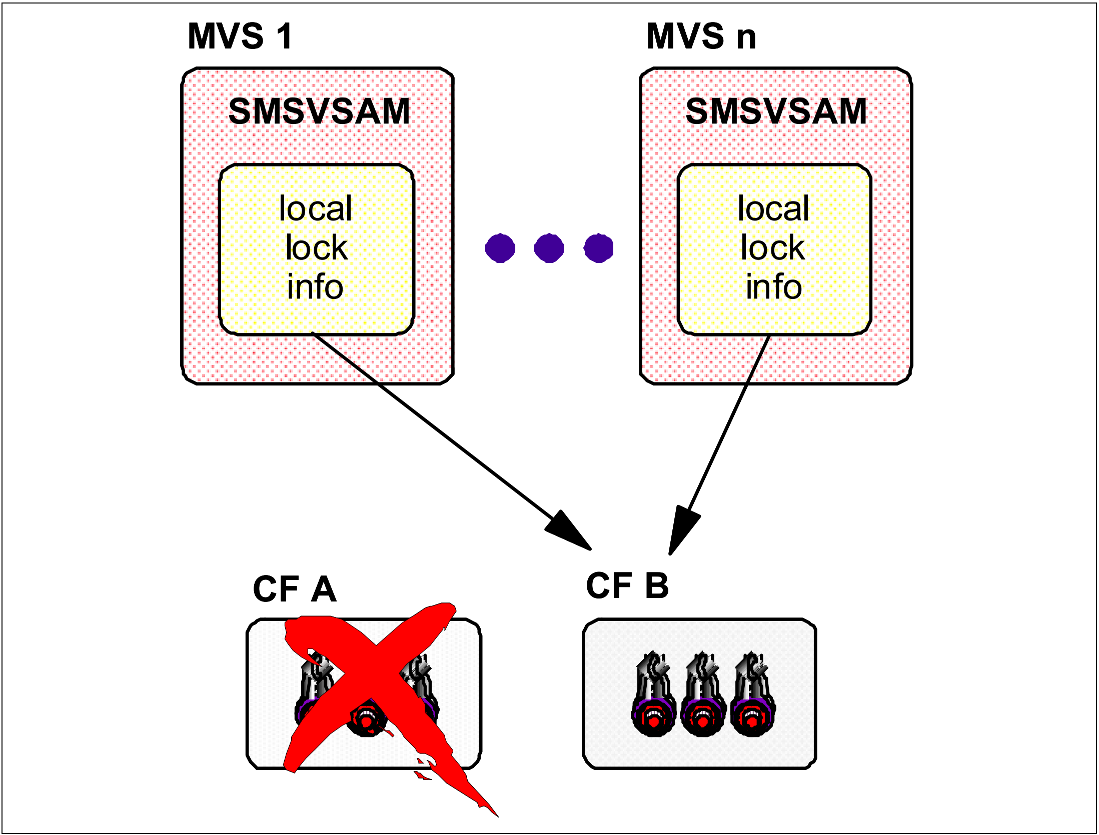


Figure 5-50 Lock structure rebuild

You can attach CFs that do not contain the master lock structure IGWLOCK00 to a subset of the systems. Plan to implement system managed duplex for your lock structures. For more information, see 5.8.10, “RLS lock structure duplexing” on page 283.

A CF cache structure that is referenced by a storage class cache set must have at least the same connectivity as the storage groups to which the storage class maps.

5.10.2 Defining multiple cache structures

Define at least two different RLS cache structures for each Cache Set Name that you define in your SMS configuration. Define each one in a different Coupling Facility. For more information about specifying the Cache Set Name, see “Defining the CF cache structures names to SMS” on page 227.

Figure 5-51 shows an example that defines two RLS CF cache structures named:

- CACHE 1
- CACHE 2

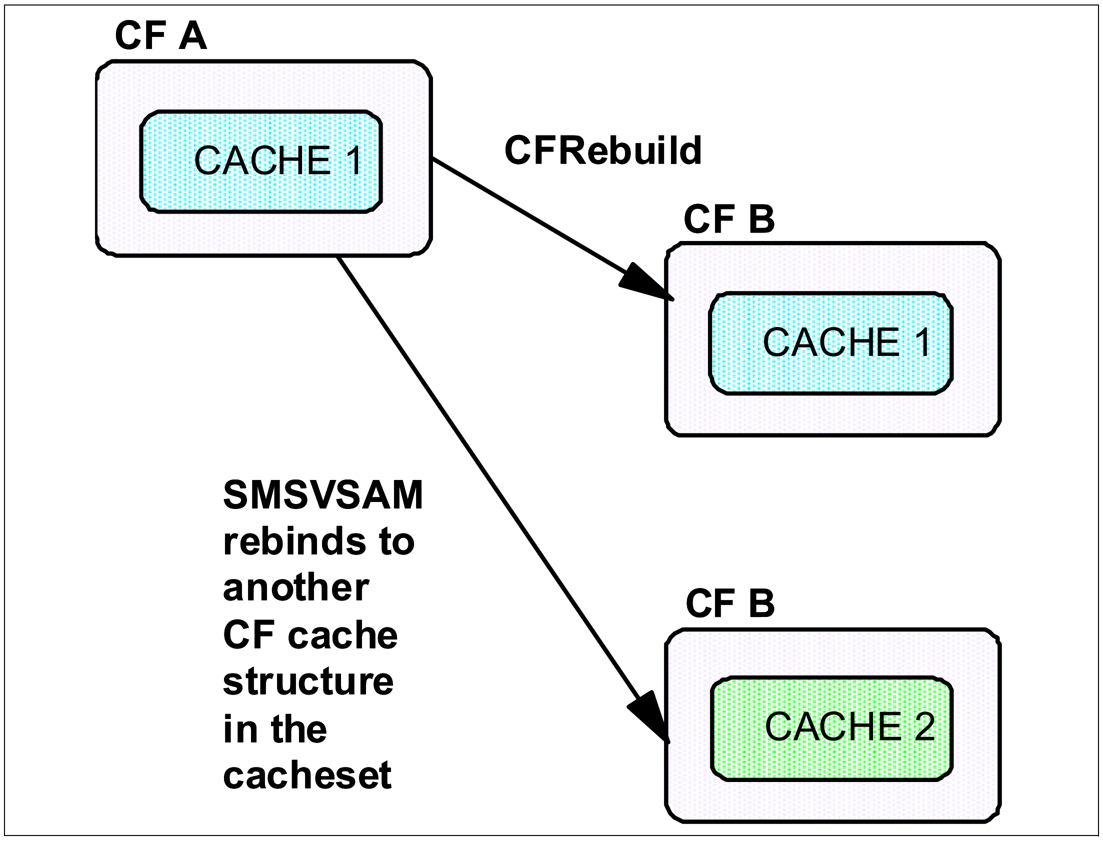

CF A
CACHE 1
CFRebuild
CF B
CACHE 1
SMSVSAM rebinds to another CF cache structure in the cacheset
CF B
CACHE 2

Figure 5-51 Multiple cache structures

Structure CACHE 2 was defined with CF A as the preferred CF in the preference list for the IXCMIAPU utility run. CF B was the second CF in that preference list. However, CACHE 2 has CF B specified as its preferred CF. Both CACHE 1 and CACHE 2 are associated with the same Cache Set Name that is defined in the SMS configuration. As a VSAM data set is opened in RLS mode, it can use either CACHE 1 or CACHE 2.

Suppose a failure in Coupling Facility CF A that causes the SMSVSAM instances to lose access to the CACHE 1 cache structure. The following events sequence takes place:

1. SMSVSAM instances lose access to CACHE 1 structure in CF A.
2. XCF protocols try to rebuild CACHE 1 in Coupling Facility CF B. Requests that access CACHE 1 are queued until rebuild is finished;
3. If the rebuild fails, the SMSVSAM instances bind the VSAM RLS data sets to CACHE 2 structure. Requests that access CACHE 1 are queued until the bind to CACHE 2 is complete.

The CI copies in local buffers are marked as invalid after the rebuild or bind is complete.

VSAM RLS uses a store-through cache method to write to data sets accessed in RLS mode. That is, SMSVSAM writes a copy of the updated CI to the CF cache structure, and also to disk. This means that any problems while accessing the cache structures do not cause data loss.

5.10.3 Placement of the SHCDSs

SHCDSs are designed to contain the information that is required for DFSMS to continue processing with a minimum of unavailable data and no data corruption when failures occur. The SCDS data sets can be either SMS or non-SMS managed.

For more information about SHCDS allocation, see 5.4.1, “Defining Sharing Control Data Sets (SHCDS)” on page 220.

For maximum availability, allocate each SHCDS (primary, alternate, and spares) in a different disk subsystem. If this configuration is not possible, spread these data sets on different volumes as shown in Figure 5-52. This rule applies to the SHCDS and also to all Sysplex Couple Data Sets.

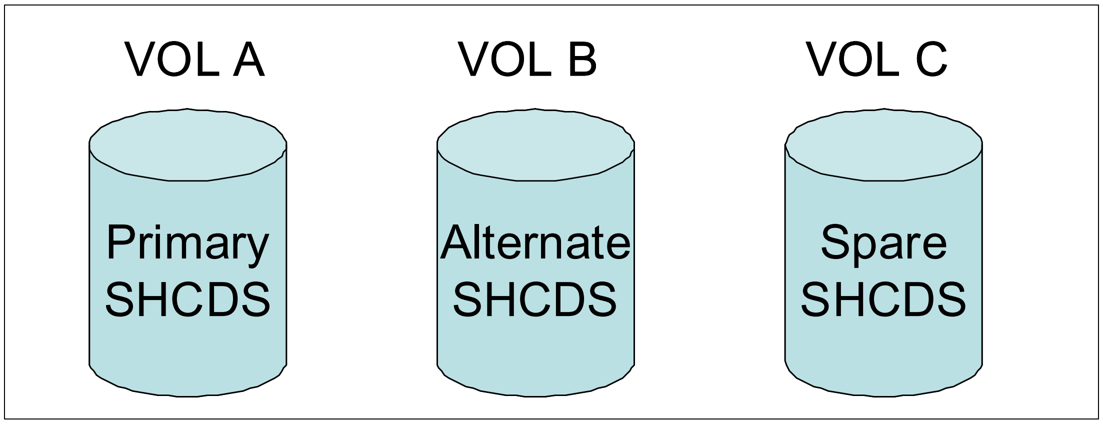


Figure 5-52 SHCDS placement

In a multiple site workload environment, running production applications in two sites, accessing the same data, and disk subsystems in both sites, use the following placement for the SHCDSs:

- Primary SHCDS on disk in one site (Site A)
- Alternate SHCDS on disk on the other site (Site B)
- Two spare SHCDS:
  - One in site A disk
  - One in site B disk

This configuration allows SMSVSAM t in one site to survive a failure in the other site.

5.10.4 Remote copying your data

Today, most organizations demand that their data must be available to their customers 24 hours a day, 7 days a week. Fewer outages are tolerated. Even small outages can cost lots of money, depend on the nature of the business. An event that causes an entire site to fail might cause the enterprise to go out of business, if it is not prepared for the event.

With this demand for high availability and fast recovery from disasters or site failures, mirroring your data is crucial. What technology you use to mirror the data depends on the particular business need, and the costs to implement a solution that is adequate for your business.

Remote copy operates with at least two disk subsystems: A primary subsystem at one location and a secondary or recovery subsystem at another location. You can locate both subsystems in the same building, or at remote locations.

IBM provides the following remote copy solutions:

- Metro Mirror, also known as Peer-to-Peer Remote Copy (PPRC)
- Global Copy, also known as asynchronous PPRC (PPRC-XD)
- z/OS Global Mirror, known as Extended Remote Copy (XRC)
- Global Mirror

The solution that you choose to remote copy your data must be based on studies like the Business Impact Analysis. These studies can estimate the cost of an outage or a disaster.

What to remote copy

The general rule is to remote copy (mirror) everything from the disks that the production systems access to the remote site. That means that every data set must be mirrored. For VSAM, copy every VSAM data set, catalog, alternate index, and DASD logstream that is used by CICS of DFSMStvs.

Specific considerations exist for each remote copy solution. IBM provides a services portfolio that covers these remote copy solutions. Contact IBM for planning and implementing these solutions.

HyperSwap

The IBM HyperSwap® technology has been around since 2002. It was initially shipped with GDPS/PPRC solution, and then with the GDPS/HyperSwap Manager (HM) solution in 2005. GDPS/PPRC and GDPS/HM provide both a continuous availability and a disaster recover solution. Together with Metro Mirror, they provide automation that runs these tasks:

1. Freezes writes to primary disks
2. Calls z/OS address spaces to run the swap of UCBs in systems in the sysplex
3. Resumes I/O operations on the secondary disks, which is transparent to the applications

Beginning with z/OS 1.9, IBM provides another type of the HyperSwap, called the Basic HyperSwap. Basic HyperSwap is a continuous availability solution.

HyperSwap eliminates disk volumes as single point of failures in a Sysplex, and helps keep your systems up and running in planned outages such as disk maintenance. Before HyperSwap, any disk failure would result in an application, system, or even sysplex outage. With HyperSwap, z/OS is now able to react to those failures by swapping the UCB pointers from the primary disks to the secondary disks.

Figure 5-53 on page 304 illustrates an HyperSwap operation. On the left, applications are running in z/OS accessing one primary volume through a control block in z/OS virtual memory called unit control block (UCB). Every device in z/OS is mapped to a unique UCB.

Each UCB in turn has a pointer to another control block called subchannel information block (SCHIB). SCHIB is used as to communicate with the I/O devices. The HyperSwap operation basically swaps the UCB pointer from the SCHIB that represents the primary device to the SCHIB that represents the secondary device.

On the right is the result of a HyperSwap operation. The UCB pointer to SCHIB now points to the SCHIB representing the secondary volume. Also, the Metro Mirror direction is reversed, although all the Metro Mirror volumes are in fact in Suspended state.

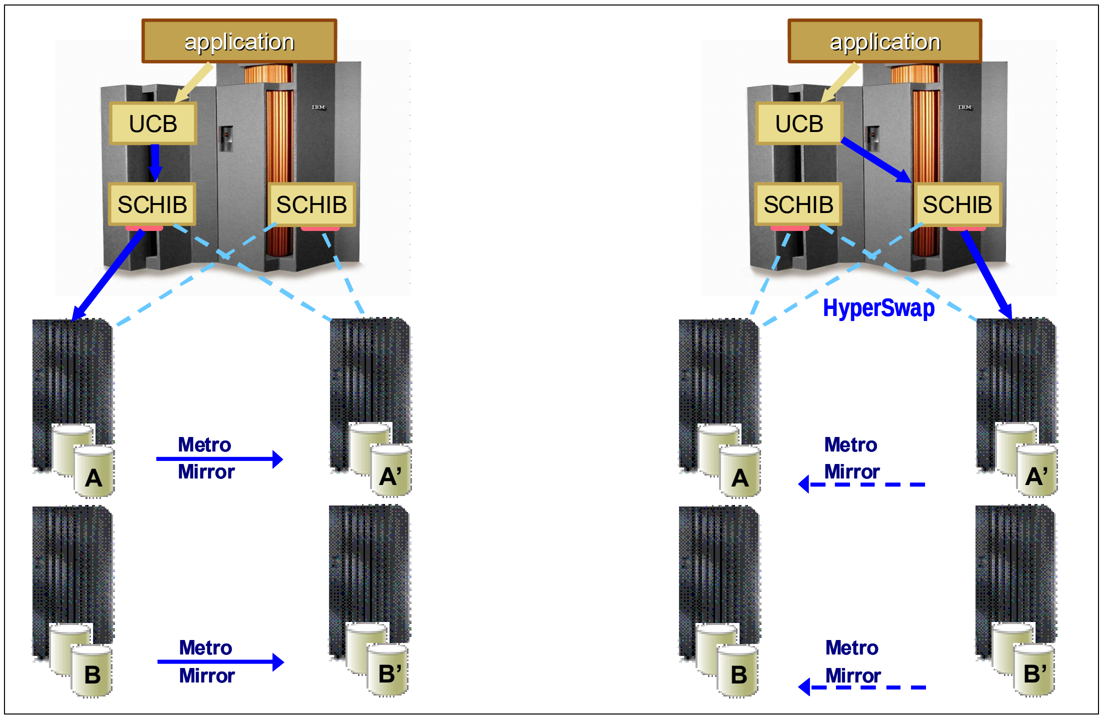


Figure 5-53 HyperSwap

VSAM in RLS and non-RLS mode can benefit from this technology. All VSAM data sets, catalogs, alternate indexes, and DASD logstreams must be included in the HyperSwap set of disks.

SHCDS maybe be excluded from HyperSwap disks. For HyperSwap environments, they can be allocated in the following configurations:

- A primary and a spare SHCDS in primary, not mirrored, disks
- An alternate and a spare SHCDS in secondary, not mirrored, disks

The scope of HyperSwap is sysplex wide. Take care that any data set or catalog that you must share with systems outside the sysplex are allocated on disks that are not defined to the HyperSwap set of disks.

5.10.5 Making backup copies

Using technology to mirror your volume disks does not eliminate the need for you to back up your data sets. Most of the data set damage and data loss that occur nowadays are because of other causes that are not hardware-related failures. These causes include user errors, application errors, improper serialization, and even software errors.

This section addresses some of the most common backup methods, focusing on the DFSMSdss backup and restore functions.

DFSMSdss

DFSMSdss is the primary tool for making backup copies and restoring these copies of any data set. The DUMP command of the DFSMSdss is the most used command to make backup copies of data sets, which are also called data set dumps. The command can also be used for full volume backups (full volume dumps).

- Data set dumps

  The DFSMSdss DUMP command can take either logical or physical backup copies of your VSAM data sets. During logical data set dump operations of SMS-managed VSAM data sets, DFSMSdss communicates with VSAM RLS to run quiesce processing of data sets being accessed by another job. This processing is done by using record-level sharing (RLS).

  You can control whether DFSMSdss uses timeout protection during RLS quiesce processing, and what the timeout value should be using the DSSTIMEOUT parameter of the IGDSMSxx PARMLIB member.

  You can also change the timeout value without IPLing the system by using the SETSMS DSSTIMEOUT(*nnnnn*) command.

- Volume dumps

  DFSMSdss can also take full volume dumps, either in logical or physical format.

  You must take care when you are restoring volumes by using the RESTORE command with the FULL or TRACKS keywords. If the target volume has data sets on it that have retained locks or data in the coupling facility that is associated with them, a full-volume or tracks restore can result in data integrity problems.

DFSMSdss supports backup-while-open serialization, which can run backup of data sets that are open for update for long periods of time. It can also run a logical data set dump of these data sets even if another application has them serialized.

When you dump data sets that are designated by CICS as eligible for backup-while-open processing, data integrity is maintained through serialization interactions. These interactions are between CICS (database control program), CICSVR (forward-recovery program), 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. These interactions are between DFSMStvs, VSAM record management, DFSMSdfp, and DFSMSdss.

Note that backup-while-open relies are only useful if you have a tool or software product, like CICS/VR, that does forward recovery by using a forward recover log. This log can be provided by CICS, DFSMStvs, or any application capable of producing such logs.

IDCAMS

You can use IDCAMS EXPORT/IMPORT or REPRO commands to make backup copies, move to another volume, and reorganize your VSAM data sets. You must quiesce these data sets from CICS, DFSMStvs, or any other application program before you run these IDCAMS functions.

IDCAMS REPRO is also a useful tool to repair damaged VSAM KSDS data sets.

5.10.6 Recovery protocols

Despite all the efforts that everybody makes to minimize error situations and their effect, errors occasionally occur. The effect of such errors depends on what sysplex components are affected by the error, and the tools and products that you implement to mitigate these error situations. This section describes some error situations.

CICS transaction failure

When an abend or another abnormal situation occur in a CICS transaction, the events that take place are shown in Figure 5-54. This example assumes that a transaction is initiated by a CICS user and the transaction made updates to a recoverable data set. However, the transaction abends before it reaches the end or the next sync point.

1. CICS releases any request for this transaction.
2. Using the CICS UNDO logs, CICS does the "undo" for all the transaction modifications to the recoverable data sets that it was updating. This process continues until the last sync point that the transaction established. See Figure 5-54.
3. CICS tells SMSVSAM to release all the locks that the transaction requested.

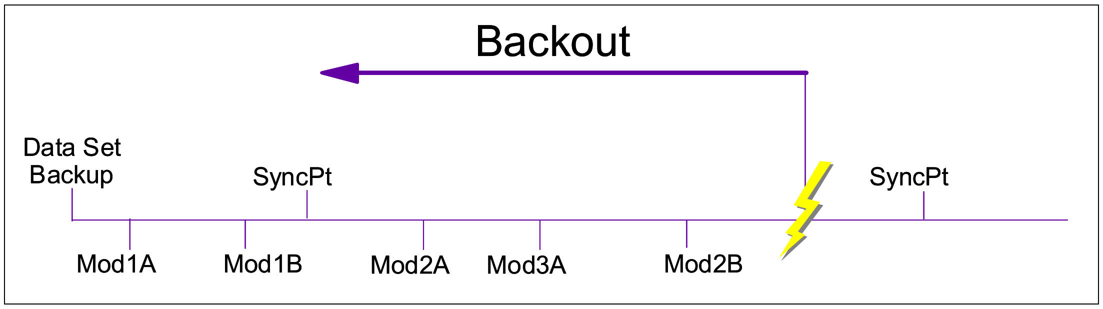


Figure 5-54 CICS transaction failure

Backout failure

After receiving a transaction error, CICS might decide to commit or, for recoverable data sets, to "undo" changes that are made by the transaction, by doing the transaction backout. The backout can fail in extreme situations, such as losing access to the recoverable data set. See Figure 5-55.

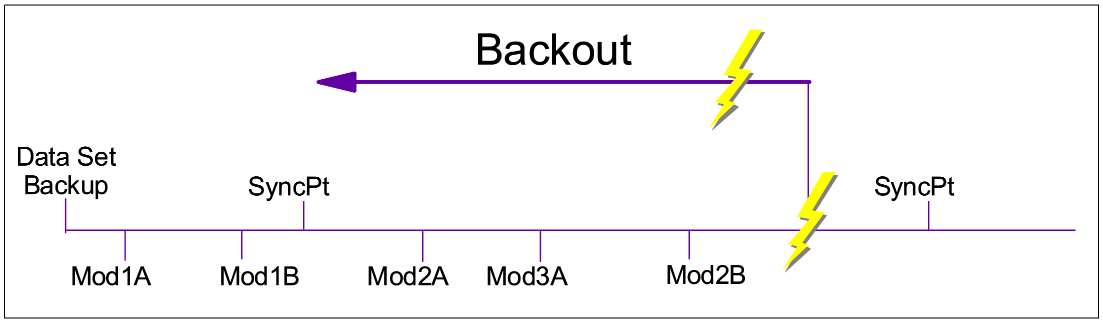


Figure 5-55 Backout failure

When a backout failure occurs, the following steps are run:

1. CICS requests SMSVSAM to mark for retention the exclusive locks (locks for update) for the VSAM data set for which backout failed.
2. CICS does the backout for other data sets updated by this transaction.
3. CICS requests that all locks marked for retention are converted to retained locks. It also requests that all other locks that are associated with this transaction are released.

The fix for this situation depends on what caused the failure. If it was caused by a data set corruption, for example, you might need to recover it from its backup copy.

Data set damage

The first sign of damage in the VSAM structure is usually a logical error, like a duplicate or missing record, detected by the application program. Under CICS you can see transaction errors, and, for recoverable data sets, probably transaction backout failures as well. These errors occur because the transaction could not complete its updates or undo the updates that it made since its last sync point.

If you suspect that your VSAM RLS data set is damaged, you perform the following actions to recover it. These actions rely on a forward recovery tool or product, like CICS/VR, to be done:

1. As soon as possible, close all applications that access that data set:
   a. From CICS, close the data set by using the appropriate QUIESCE or CLOSE command. The following command tells the CICS region named *cicsname* to quiesce data set *datasetname* from RLS access:

      ```
      F cicsname,CEMT SET DSN(datasetname),QUI
      ```

   b. Stop all applications that access the data set.
2. If the VSAM data set that you suspect is damaged is a KSDS, run IDCAMS EXAMINE DATATEST INDEXTEST against the damaged data set. This command will confirm that it is damaged. See Figure 5-56 for an EXAMINE example. Save the output for further investigation. For more information about the EXAMINE command, see *DFSMS Access Method Services for Catalogs*, SC26-7394.

```
//MHLRES3A JOB (999,POK),'MHLRES3',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=6M
//STEP1    EXEC    PGM=IDCAMS
//SYSPRINT DD      SYSOUT=*
//SYSIN    DD      *
   EXAMINE -
           NAME(MHLRES3.VSAM.KSDS) -
           INDEXTEST -
           DATATEST
/*
```

Figure 5-56 Examine example

3. Use the SHCDS FRSETRR command from IDCAMS to set the RECOVERY REQUIRED information in the catalog entry for the damaged data set. For example, the following command sets Recovery Required for VSAM data set MHLRES3.VSAM.KSDS:

   ```
   SHCDS SETRR(MHLRES3.VSAM.KSDS)
   ```

4. For recoverable data sets, use the IDCAMS SHCDS FRUNBIND command to unbind any retained lock information from the data set. The following command unbinds all retained locks for data set MHLRES3.VSAM.KSDS:

   ```
   SHCDS FRUNBIND(MHLRES3.VSAM.KSDS)
   ```

5. Define a new data set.
6. Restore a backup copy into the newly defined data set.
7. Use the IDCAMS SHCDS FRSETRR to set the RECOVERY REQUIRED information in the catalog entry for the new data set.
8. Run a forward recovery utility, like CICS/VR, against the new data set. This utility applies all updates that it finds in the forward logs to the existing backup copy. CICS/VR automatically runs the set and reset of the RECOVERY REQUIRED information, and also the bind and unbind of locks.

9. Delete or rename the original damaged data set.
10. Rename the new data set with the name of the original damaged data set.
11. Use the IDCAMS SHCDS FRBIND command to rebind the retained locks to this data set, as in the following example:

    ```
    SHCDS FRBIND(MHLRES3.VSAM.KSDS)
    ```

12. Use the IDCAMS SHCDS FRRESETRR command to reset the RECOVERY REQUIRED information in the catalog, as in the following example:

    ```
    SHCDS FRRESETRR(MHLRES3.VSAM.KSDS)
    ```

13. Unquiesce and allow CICS or DFSMStvs to run these recovery procedures:
    - Back out transaction updates
    - Commit transactions
    - Release retained locks

I/O errors

If an I/O error occurs, CICS or DFSMStvs cannot back out the updates that are made by any transaction:

- All of the CI copies of the affected data sets that exist in the local buffers are discarded.
- All affected transactions receive error return codes, and then attempt to recover.

You can minimize or even eliminate the effect of such errors by using technologies like HyperSwap. HyperSwap redirects the I/O operation to a secondary disk volume, transparently to CICS and all users.

CICS failure

In the case where a CICS region that has opened VSAM data sets in RLS mode fails, the following events take place:

1. SMSVSAM discards all the buffers that the failing CICS modified.
2. SMSVSAM converts all locks that are associated with recoverable VSAM data sets to retained locks. All other locks are released.
3. As soon as the failing CICS restarts, it requests the list of transactions with retained locks.
4. CICS performs the recovery of the transactions by committing, backing out, or just releasing the retained locks.

If an Abend or Cancel occurs when CA reclaim is in progress, VSAM and RLS try to complete the CA reclaim. The philosophy is to complete the interrupted process as soon as feasible to minimize any possible complications. The recovery will complete the interrupted CA reclaim even after the user issues the SETSMS command to disable CA reclaim.

The CA reclaim algorithm is such that when it is interrupted, the worst case is wasted DASD space, which is no worse than without CA reclaim. There are no data integrity problems that are caused by interrupted CA reclaim.

You can minimize the effect of CICS region failures by speeding up restart. Do so either by implementing the z/OS Automatic Restart Manager (ARM), or an automation product, like Tivoli System Automation for z/OS, that detects when an address space is gone and automates its restart.

SMSVSAM failure

The SMSVSAM address space normally restarts itself without any manual intervention. After the SMSVSAM server restarts, it uses z/OS Event Notification to notify all the CICS regions within its MVS image that the SMSVSAM server is available again.

CICS runs a dynamic equivalent of emergency restart for the RLS component, and drives backout of deferred transactions.

System failure

In the case where a z/OS image goes down, SMSVSAM and any CICS region under the system go down together. The other z/OS images and SMSVSAM instances in the sysplex convert the data set locks that are associated to recoverable VSAM data sets, to retained locks. All other locks are released.

As soon as the failing image, and its SMSVSAM address space and its CICS regions, are up again, the CICS regions run the required recovery.

If the system fails during a CA reclaim, a subsequent IDCAMS EXAMINE against the affected VSAM KSDS data sets might show some harmless and temporary messages that resulted from interrupted splits, especially the following message:

```
IDC11724I DATA COMPONENT CA NOT KNOWN TO SEQUENCE SET
```

This message reports a temporary situation. It will be cleaned up after the first application access to the data set, and does not cause any integrity issue.

If you want to get a clean EXAMINE execution, run the IDCAMS VERIFY RECOVER on the affected data sets before you run EXAMINE.

6

DFSMStvs

This chapter addresses DFSMStvs. For more information about DFSMStvs, see the following IBM Redbooks:

- *DFSMStvs Overview and Planning Guide*, SG24-5741;
- DFSMStvs Presentation Guide, SG24-6973;
- DFSMStvs Application Migration Guide, SG 24-6972.

This section does duplicate the material that is covered in those publications. Rather, it documents hands-on experiences with DFSMStvs.

This chapter includes the following sections:

6.1 Introducing DFSMStvs

This chapter introduces recent enhancements to VSAM and shows how DFSMStvs is a natural extension to the functions provided by VSAM Record Level Sharing.

The key functions of DFSMStvs are to allow VSAM to provide locking, two-phase commit, backout, and logging facilities. These facilities allow multiple batch update jobs to run concurrently with CICS access to the same data sets while still maintaining integrity and recoverability.

6.2 Why DFSMStvs?

In the past, it was often acceptable for a CICS system to be available during normal daytime business hours, perhaps for a total of 12 hours. This left plenty of time for the CICS system or application to be shut down and for supporting batch work to be run. There was no requirement to make business data available on the Internet.

CICS typically is active and then shut down in preparation for running batch work each day. As soon as CICS stops, backups are taken of key data sets as a point of recovery. Then, batch jobs can be scheduled to run. If you have several jobs that update the same data set, they run sequentially because they cannot share the data set for update. After the batch updates are complete, there might be a job to read the updated data and produce reports. Another backup is usually done at this stage. When this backup is complete, CICS is restarted and becomes active again. DFSMStvs allows you to extend the CICS window of availability.

6.2.1 Extending CICS availability

There is increasing pressure to extend the availability of CICS systems. This pressure is because of the need for better customer service. Better customer service in turn requires longer service hours. Modern customers expect 24 hours per day Internet access to core business applications. Meeting this expectation is desirable to improve competitiveness, extend the range of customers served, and reduce costs.

You can use DFSMStvs to meet these new or changing business objectives:

- Extending service hours for automated teller machines
- Providing kiosk access to your applications for your customers
- Linking your applications with your suppliers or customers
- Web-enabling applications for direct customer use
- Allowing batch reruns to be done during the online day

In each case, there will be new pressures on availability of your systems. One consequence of the need for greater availability is that you cannot afford the CICS downtime that is caused by your existing batch window. The actions that you must take depend on whether you can still tolerate a batch window, although it will be a shorter window.

6.2.2 Reducing the batch window

There are several techniques to reduce the batch window, but each has drawbacks.

You can use the External CICS Interface (EXCI), which allows a non-CICS application program to call a CICS program. Change application programs to use EXCI to pass data to a

CICS server transaction. This transaction then runs the updates with all the facilities available to CICS. However, this technique needs major application changes, and requires that you provide a server transaction, but it does completely remove work from a batch window.

You can remove a data set from CICS file control while CICS is still running. You use the CEMT transaction to close the data set and reopen it as read-only. You can then copy the file and make batch updates to the copy while still providing read access to the original. After the updates are completed, deallocate the old version and allocate the new, updated version to CICS. This technique does not remove work from the batch window, but does offer limited (read-only) access while updates are being done.

You can force VSAM to allow sharing by using SHAREOPTIONS (3) or SHAREOPTIONS(4). However, all integrity and recovery issues about locking and logging become an application responsibility. The application changes needed to give you complete integrity and recoverability are extremely complex. For example, you must provide logging, locking, and commit protocols such that a CICS transaction backout would not remove updates that were made correctly by a batch job.

Finally, you can change transactions to collect updates while a data set is being updated in batch and apply the changes later. A memo file is used to collect new records. Transactions are changed to switch between the main file and the memo file when the main file is not available for updates or additions.

Inquiries must also check the memo file first. When the main file update or reorganization is complete, the contents of the memo file are copied into the main file. The switch is then reset so that all accesses are now run solely against the main file. This configuration might have integrity problems, depending on the functions that you provide. Adding new records might be safe, but updating existing records might update a record that has been updated elsewhere.

These methods provide some relief at the expense of added complication or reduced integrity and recoverability. However, a more general solution to the problems of running batch updates for CICS VSAM data is needed.

6.3 DFSMStvs terms and concepts

If you are interested in DFSMStvs as a storage administrator, you might not be familiar with some of the terms and concepts involved. This section offers some brief definitions.

6.3.1 Backward recovery

If a transaction fails, it might leave data in an inconsistent state. For example, one update is requested, but the program failed before requesting a related update.

The process of removing changes that are possibly inconsistent is variously called *backward recovery*, *rollback*, and *backout*. It requires the use of a log that holds images of the records before they were updated. This log is called an *undo log*. In CICS terminology, the records are known as *before* images.

6.3.2 Forward recovery

If a data set is lost, a common way of getting the data back is to recover from a backup copy. You then apply all the changes that were made after the backup copy was taken.

This process is called *forward recovery*. It requires a log of the changes that were made to a data set together with a date and time stamp. This log is called a *redo* or *forward recovery* log.

6.3.3 Atomic updates

Atomic updates are an indivisible group of updates. Either all updates in the group are made or none are made. It is not possible for some updates to be made but for others to fail, and still maintain data integrity. This property is an absolute requirement for integrity. The need for atomic updates is shown in the simple example of a transfer of money between two bank accounts in Figure 6-1.

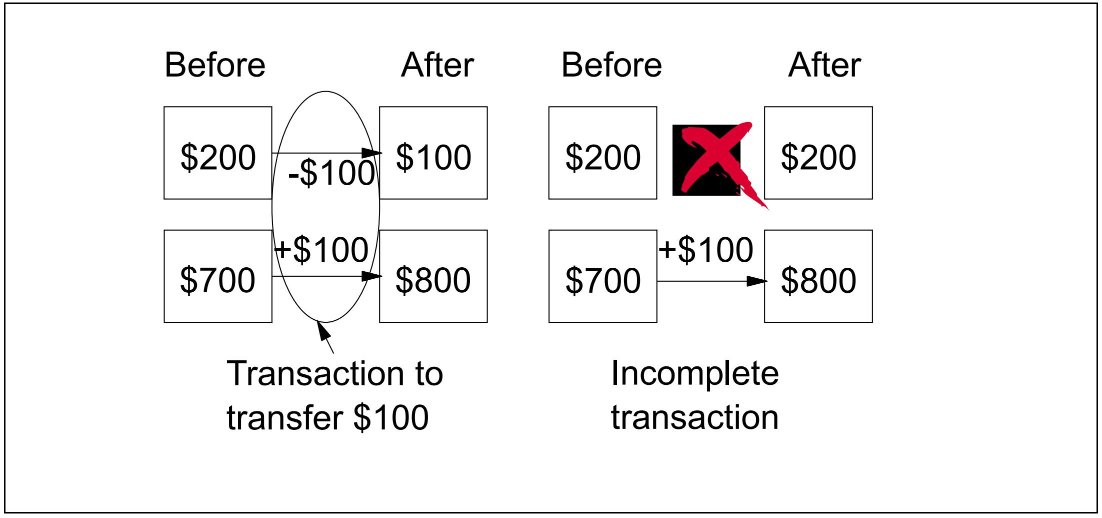


Figure 6-1 An atomic update example

Clearly, either *both* changes must be made, or *neither* can be made. If only the addition or credit is made, the bank has given the customer money. If only the subtraction or debit is made, the bank has taken money from the customer.

Making final changes to the data is called *committing*. Only the application itself knows when data should be committed. It requests that the data be committed at an appropriate point in the processing flow. For example, the application that is described in Figure 6-1 would request a commit when the updates for both accounts are complete. It would not make sense to commit if only one update is complete, as an integrity exposure would be created.

6.3.4 Unit of work and unit of recovery

A *unit of work* (UOW) is the term that is used in CICS publications for a set of updates that are treated as an atomic set of changes. The z/OS Resource Recovery Services (RRS) use *unit of recovery* to mean much the same thing. Therefore, a unit of recovery is the set of updates between synchronization points. There are implicit synchronization points at the start of a transaction. There should also be explicit synchronization points that are requested by an application within a transaction or batch job. It is preferable to use explicit synchronization for greater control of the number of updates in a unit of recovery.

Changes to data are durable after a synchronization point. That means that the changes survive any subsequent failure.

Figure 6-2 shows the units of recovery (noted as A, B, and C) in a job or transaction. Notice that the points of synchronization are shown, whether explicitly requested by a commit or implicitly at the start and end of the transaction.

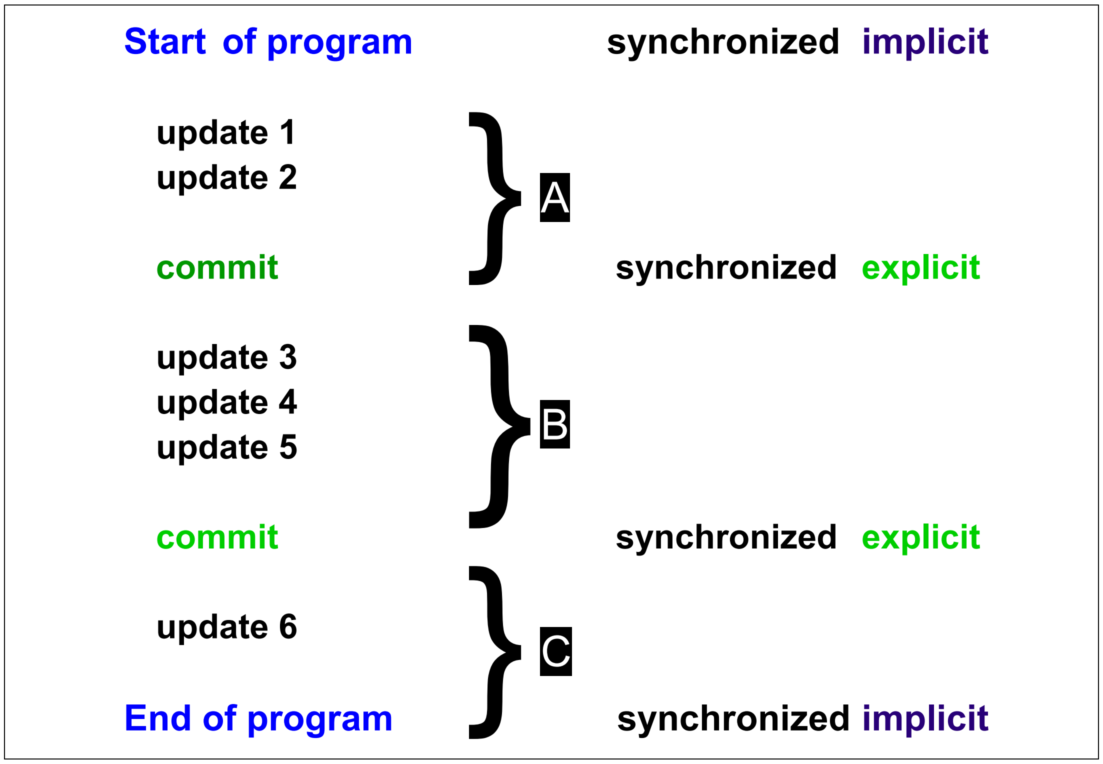


Figure 6-2 Unit of recovery examples

Figure 6-2 demonstrates what can happen, but it is not an ideal configuration. For reasons described later, generally you want an explicit commit done before a data set is closed.

The z/OS RRS and DFSMStvs documentation draw a distinction between unit of work and unit of recovery. In this case, unit of work has its MVS definition (a task that runs under its own TCB) and can refer to more than one unit of recovery. In Figure 6-2, the sequence of three units of recovery between the start and end of the transaction is a unit of work.

This definition of a unit of recovery in this book.

6.3.5 Two-phase commit

Two-phase commit is a technique that is widely used by database management systems to provide an atomic update capability. It is necessary when there are multiple resource managers that are involved and you have what is known as a distributed unit of work.

Two-phase commit requires that there is a coordinator that is called at synchronization points. In the first phase, the coordinator asks the individual resource managers (such as DFSMStvs) to prepare to commit the changes that they have made. The resource managers must reply to the coordinator to say that they are ready to commit. When they do so, the coordinator will, as the second phase, tell the updaters to make the changes permanent.

For DFSMStvs, z/OS RRS plays the coordinating role. Commit requests are generated implicitly at the successful end of a job step. They should also be done explicitly by inserting commit calls in the program logic.

6.3.6 In-flight and in-doubt

A unit of recovery is described as being *in-flight* if it has changed a record in a recoverable data set, but has not yet committed or backed out the changes.

A unit of recovery is described as being *in-doubt* in the brief time between DFSMStvs agreeing to an RRS request for a commit, and the signal from RRS that all the commit participants have agreed and that the commit should be done.

6.3.7 Repeatable read

With batch programs using RLS, you have a choice of two ways of handling read integrity. The default is *no read integrity* (NRI) where record locks are not obtained for a read. This configuration means that you can read a record and get uncommitted data that might be backed out. This is sometimes called a *dirty read*. The other option is to ask for a *consistent read*. In this case, RLS ensures that the read request waits until updates for that record are committed.

DFSMStvs adds a third option, *repeatable read* (also known as *consistent read explicit*). This option provides a way of ensuring that a record cannot be changed by someone else until the requester of repeatable read either commits or backs out. Only then does the system release the lock that prevents others from updating the record. It also gives your program isolation from other changes. It cannot see updates that are made by other programs that have not been committed, and so it is a form of consistent read.

The difference between a consistent read and a repeatable read is that a repeatable read keeps a lock on the record read until the unit of recovery that requested the read commits or backs out. Hence, a consistent read gives you a consistent view of data where all associated updates have been committed, but does not maintain that view. A repeatable read allows the requester to maintain that consistent view of the data so that future reads will get the same data until the requester itself decides to commit.

6.3.8 Recoverable data sets

RLS introduced the concept of recoverable and non-recoverable data sets to VSAM itself. Previously, CICS file control supported recoverable VSAM data sets, but only for access from CICS transactions.

A *recoverable data set* must be accessed by a system that can run two-phase commit and backout of failed updates. The VSAM cluster must be defined with either the LOG(UNDO) or LOG(ALL) attributes to be recoverable.

If LOG(UNDO) is specified, backout can be done, but not forward recovery. This configuration means that the image of a record before it was changed is logged. Therefore, the log entry can be used to undo a change in the event of failure.

If LOG(ALL) is specified, both backout and forward recovery can be done. Forward recovery requires that the image of a record after a change is stored so that the log entry can be used to redo a change.

A *non-recoverable data set* is defined with either the LOG(NONE) attribute or without the log attribute being specified. Neither a backout or a forward recovery capability exists for a non-recoverable data set. Because these capabilities do not exist, they cannot be compromised by batch updates, so batch updating is allowed for non-recoverable data sets.

6.4 CICS support for recoverable VSAM

You can define VSAM data sets to CICS as recoverable resources. This definition is in the CICS file resource definition or the ICF catalog. For RLS and DFSMStvs, the definition must be in the ICF catalog.

CICS can provide logging for both backward and forward recovery. Backward recovery supports transaction backout by writing a copy of a record before it was updated to a log, sometimes called an *undo* log. CICS runs the backwards recovery itself. Forward recovery adds the ability to take a backup copy of a VSAM data set and use logged updates to reapply changes to that data set. This is also referred to as a *redo* log. CICS does not do forward recovery. A program such as CICSVR is needed to use the redo logs to provide forward recovery.

If a transaction ends in error, CICS backs out the changes that were made by that transaction. CICS provides a commit mechanism that is used when a logical set of updates has completed. When changes are committed, they cannot be backed out.

VSAM data has been able to be shared between CICS systems by using function shipping between application-owning regions (AORs) and a file-owning region (FOR). This configuration is shown in Figure 6-3.

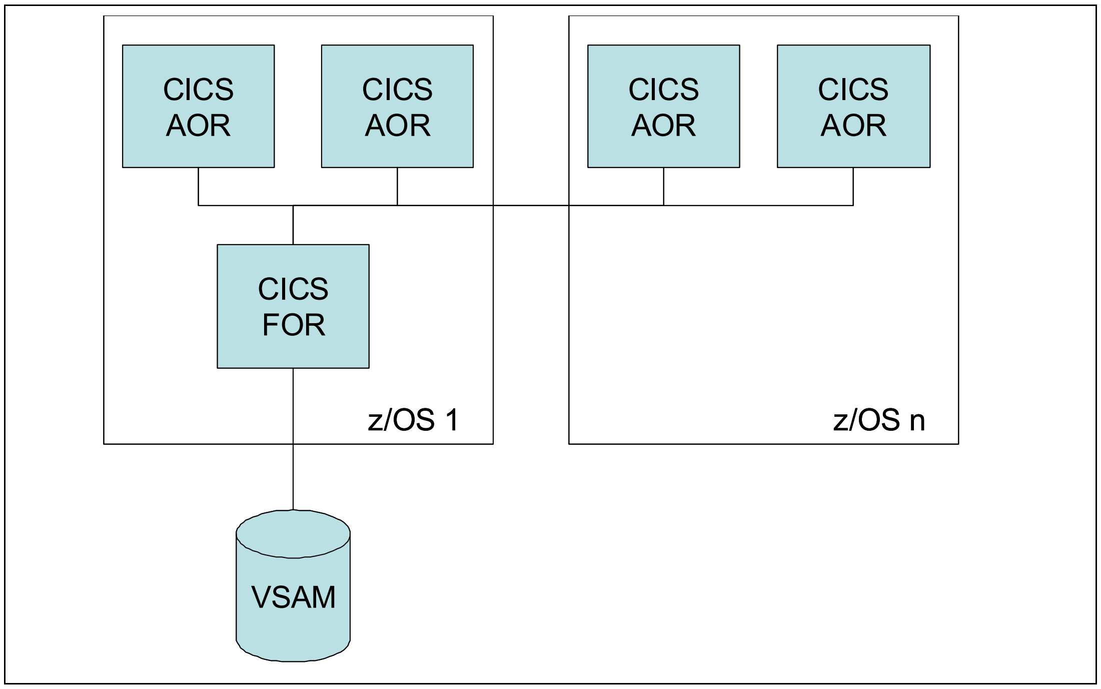


Figure 6-3 Sharing VSAM data through a CICS file-owning region

There are several drawbacks with this approach:

- The FOR is a single point of failure.
- If the workload of the FOR increases, it might become a performance bottleneck. You cannot use Parallel Sysplex horizontal growth to provide a growth path.
- Transactions are shipped to the system with the FOR from other systems by using XCF links or VTAM. These applications can impose performance costs.

6.5 The RLS connection

As seen in Chapter 5, “VSAM Record Level Sharing” on page 209, RLS provides a new mode of access to VSAM data. RLS uses the Coupling Facility to provide the lock structures and data caching abilities to manage and ensure data integrity when VSAM data is shared. Therefore, Coupling Facility is required for RLS even if you want to share VSAM data in a monoplex. DFSMStvs extends the sharing of data to CICS and batch level sharing, which also maintains data integrity for reads and updates. To do so, DFSMStvs must provide the same abilities that CICS provides, such as logging for forward and backward recovery, backout, and a two-phase commit process.

When a VSAM application issues a GET or a PUT (or the high-level language equivalent of these VSAM macros), VSAM RLS acquires locks on behalf of the application. A coupling facility is used to hold the locks so that they can be shared by all the SMSVSAM address spaces in a sysplex.

In general, VSAM RLS obtains shared locks for read requests, and exclusive locks for update or delete requests. It waits for a certain time if another task holds the required locks. A share lock can be obtained for a resource if there is no lock for that resource, or there are only share locks held for it. An exclusive lock can be obtained only if no other locks are held.

Tip: CICS publications refer to share locks as shared locks.

The DFSMStvs access mode uses VSAM RLS locking, and so it is compatible with CICS.

6.6 DFSMStvs logging

The z/OS system logger is used to provide the logging functions that are needed to ensure data consistency after failure. The redo log name is held as an attribute of a recoverable data set, whereas the undo log name is fixed. Forward recovery logging is done by both CICS and DFSMStvs to the same redo log. CICS logs changes that are made by CICS transactions, while DFSMStvs logs changes made by its callers as shown in Figure 6-4.

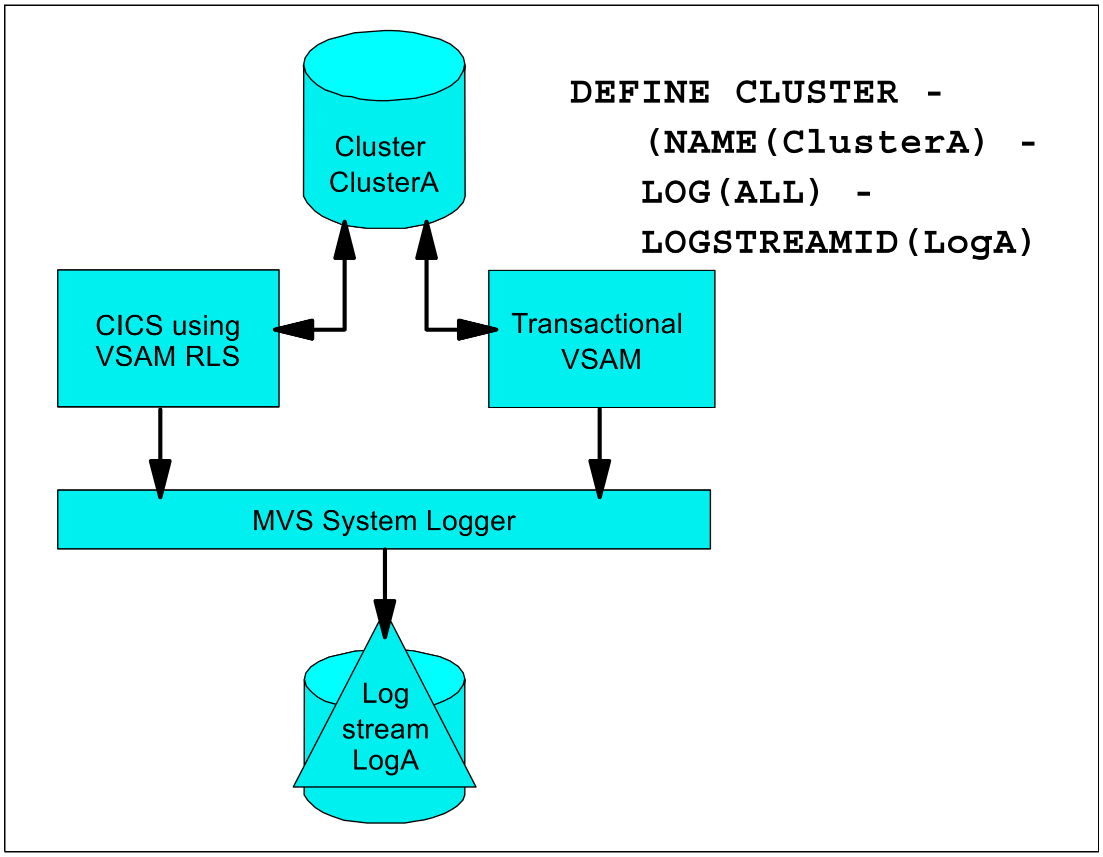


Figure 6-4 Merged forward recovery log

The system logger is used because it can merge log entries from many z/OS images to a single logstream. A logstream is a set of log entries. A single logstream across a sysplex eases management and protects data integrity.

This ability to merge log entries is used for the forward recovery logs to ease recovery. The system logger writes log entries to the coupling facility, disk, or both. However, each instance of DFSMStvs has a private undo log. The undo logs are not shared. An *instance* of DFSMStvs is the single version of DFSMStvs running in one z/OS image and defined by the IGDSMSxx PARMLIB member for that z/OS image.

DFSMStvs uses the following logstreams:

- *Primary*, also called *backout* or *undo logstream*

  There is one backout logstream for each instance of DFSMStvs. It is not shared between SMSVSAMs. The name of the backout logstream is constructed from the unique name of an instance of DFSMStvs that you define in the TVSNAME keyword of IGDSMSxx parmlib member.

The name format is *tvsname*.IGWLOG.SYSLOG, where *tvsname* is of the form IGWTV*nnn*.

- *Secondary*, also called *shunt logstream*

  There is one shunt logstream for each instance of DFSMStvs. It is not shared between SMSVSAMs. The name of the shunt logstream is constructed by using the unique name for an instance of DFSMStvs that you define in the TVSNAME keyword of IGDSMSxx parmlib member.

  The name format is *tvsname*.IGWSHUNT.SHUNTLOG, where *tvsname* is of the form IGWTV*nnn*.

  The shunt log is used when backout requests fail and for long running units of recovery. In these cases, log records are moved so that DFSMStvs can better manage space in the primary log.

- *Forward recovery logstreams*

  Forward recovery logs are used by data sets for which LOG(ALL) is specified. When LOG(ALL) is specified, you must also specify LOGSTREAMID to provide the name of the forward recovery logstream. This logstream is used by all instances of DFSMStvs and CICS to provide forward recovery capability.

- *Log of logs*

  This is an optional logstream that is specified in SYS1.PARMLIB(IGDSMSxx). If it is defined, it contains copies of log records that are used to automate forward recovery.

  Although the log of logs is optional, use it if you want to use forward recovery at all. However, when you define a log of logs, it must be present otherwise DFSMStvs fails.

Figure 6-5 shows a simple example with undo and redo logging. The vertical arrow shows the commit point at which time the undo log records are effectively forgotten. The undo log records show how the records existed before the transaction, whereas the redo log records shows how the records exist after they are updated. The log records are prefixed with header information used for recovery or backout.

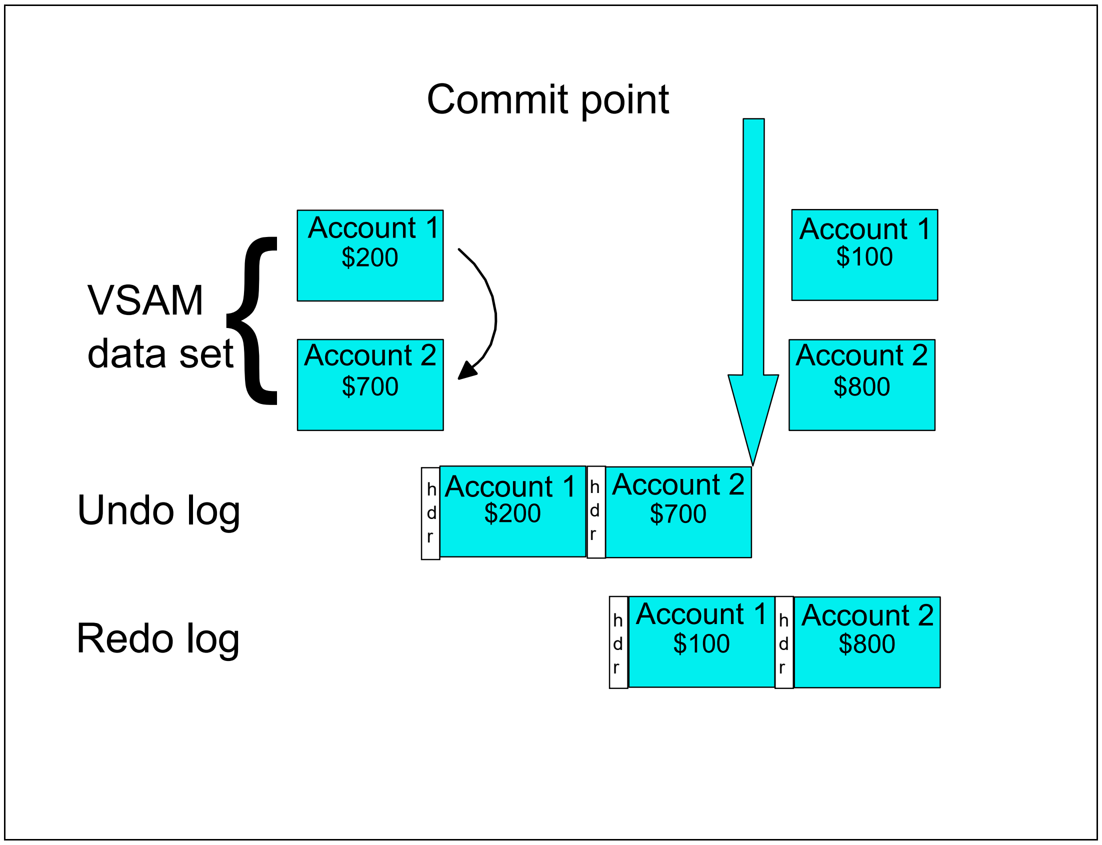


Figure 6-5 Undoing and redoing logging

6.7 Recovery coordination

For DFSMStvs, the resources that are being protected are the records within recoverable VSAM data sets. There are three types of participants in resource recovery:

1. Application: The application program requests the use of resources, in this case, records. It is responsible for requesting that changes be committed to make them permanent or that they be backed out to remove them. These processes return the records to their previous state.
2. Resource Manager: DFSMStvs uses the VSAM provided interfaces for two-phase commit and backout. These interfaces allow for these processes:
   - Reading records
   - Updating records
   - Insertion of records
   - Erasure of records

3. Syncpoint manager: RRS is the syncpoint manager. It is responsible for providing a single point of coordination for the commit processing of resources that are owned by different resource managers. It provides the interfaces that an application uses to perform these changes:
   - Commit changes
   - Back out changes

   z/OS Recoverable Resource Services are used to coordinate commit and backout requests from applications, and any other resource managers that might be needed. Apart from DFSMStvs, other resource managers that use z/OS RRS are DB2 UDB, IMS, and IBM MQSeries®. An application can use a combination of VSAM, IMS, DB2 UDB, and IBM WebSphere® MQ services, and have commit and backout that is performed atomically for all these resources. Figure 6-6 shows RRS as the syncpoint manager.

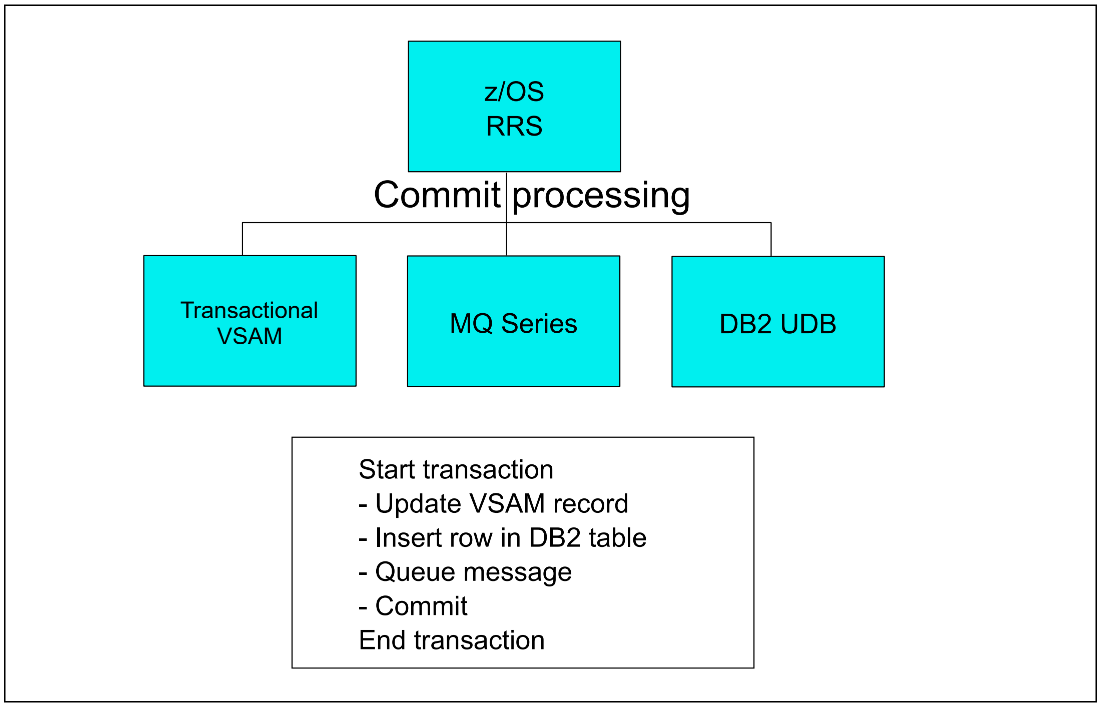


Figure 6-6 RRS as the syncpoint manager

The DFSMStvs code calls RRS to register its interest in a unit of recovery. A unit of recovery represents the set of changes made by an application to a resource since the last commit (implicit or explicit) or backout. Each unit of recovery is associated with a context.

DFSMStvs writes a copy of an unmodified record (the record as it existed before an update) to the undo log. When the data set’s log attribute is set to ALL, it also writes a copy of the modified record (as it exists after an update) to the forward recovery log.

When an application requests either a commit or a backout, z/OS RRS calls DFSMStvs to do the commit or backout on behalf of the application. The processing flow shown in Figure 6-7 on page 323 is in a simplified form for a successful commit.

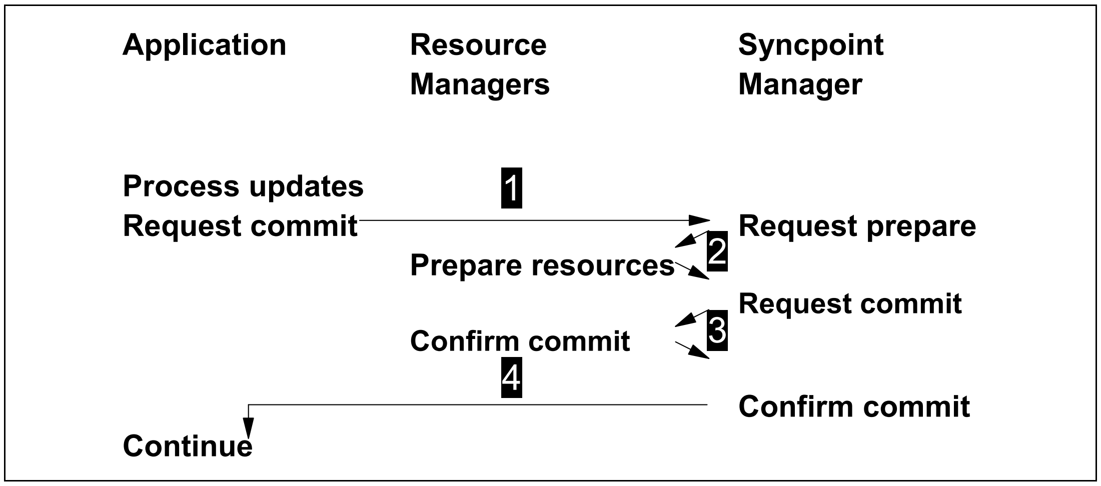


Figure 6-7 Commit processing participants

1. The application decides that it is ready to commit changes. It requests a commit and that request is passed to the syncpoint manager.
2. The syncpoint manager sees what resources are involved. It asks each resource manager that has expressed interest in this unit of recovery to prepare to commit and notify it of readiness.
3. If all the resource managers reply that they are ready to commit, the syncpoint manager tells them to complete the process.
4. When that is done, the syncpoint manager can then confirm that the commit has completed to the application

6.8 Ordering and installing DFSMStvs

First, verify whether you already have DFSMStvs installed or not. DFSMStvs is a separately priced feature that works on top of VSAM RLS/SMSVSAM. It allows non-CICS applications to function at the transaction level and takes advantage of the features available using VSAM RLS data sets.

The easiest way to verify the current DFSMStvs installation status on your system is to watch for the Health Checker messages IGWRH301E or IGWRH302E. To enable the health check, verify that either your POLICY statement or HZSPRMxx parmlib member includes:

```
UPDATE
CHECK(IBMVSAMRLS, VSAMRLS_TVS_ENABLED)
SEVERITY(LOW) INTERVAL(04:00) DATE(20061206)
REASON('Your reason for making the update.')
```

If you get the IGWRH301E message, DFSMStvs is installed but not started. If you get the IGWRH302E message, DFSMStvs is not properly installed.

Perform these steps to install DFSMStvs:

1. Purchase the priced feature: DFSMStvs is a separately priced feature that must be purchased. You can find the product through normal ordering channels with feature number 6330.

2. SMP/E install: You must install DFSMStvs through SMP/E. The FMID to be included is named IZOTV*rrr*, where *rrr* is the release of your DFSMS MVS product such as 190, A10, or B10.
3. Enable the product in your IFAPRDxx member: You need your IFAPRDxx parmlib member to include the following entry:

```
PRODUCT OWNER('IBM CORP')
        NAME('Z/OS')
        ID(5694-A01)
        VERSION(*) RELEASE(*) MOD(*)
        FEATURENAME(DFSMSTVS)
        STATE(ENABLED)
```

For more information, see IBM APAR II14627.

6.9 Implementing DFSMStvs

You must have VSAM RLS already set up before you can implement DFSMStvs.

After RLS is set up, you already have defined the necessary lock and cache structures in the coupling facility. Perform the following steps to implement DFSMStvs:

1. Defining list structures in the CFRM policy
2. Defining the log structures and logstreams in LOGR policy
3. DASD staging data sets
4. Setting up RRS
5. Setting up security definitions
6. Modifying IGDSMSxx parmlib member

6.9.1 Defining list structures in the CFRM policy

You must define the list structures in the coupling facility resource management (CFRM) policy. The list structures contain the various logstreams that are used by DFSMStvs. For more information, see 6.9.2, “Defining the log structures and logstreams in LOGR policy” on page 325.

As you define your list structures, consider adopting a naming convention for your list structures that helps to identify the purpose of the structure. For example, you can use a format such as LOG_purpose_nnn:

purpose	Identifies how you use the structure.
nnn	A sequence number to allow for more than one structure for each purpose.

For example, you might choose the following names:

- LOG_IGWLOG_001

  For the DFSMStvs primary system log. Make the structure large to avoid the need to write data to DASD.
- LOG_IGWSHUNT_001

  For the DFSMStvs secondary system log. Make this structure s small. However, it requires a large buffer size. A structure of 100 KB per logstream is usually sufficient.
- LOG_FORWARD_001

  For forward recovery logs in which block writes are forced periodically.

- LOG_IGWLGLGS_001

  For the log of logs.

Figure 6-8 provides sample JCL to define the list structures. Four list structures are defined by this job.

```
//LstStruc JOB (999,POK),'CFRM',CLASS=A,REGION=4096K,
//             MSGCLASS=X,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
  DATA TYPE(CFRM) REPORT(YES)
  DEFINE POLICY NAME(CFRM21) REPLACE(YES)

    CF NAME(CF01) DUMPSPACE(2048) PARTITION(E) CPCID(00)
       TYPE(002064) MFG(IBM) PLANT(02) SEQUENCE(000000010ECB)

    CF NAME(CF02) DUMPSPACE(2048) PARTITION(D) CPCID(00)
       TYPE(002064) MFG(IBM) PLANT(02) SEQUENCE(000000010ECB)
         STRUCTURE NAME(LOG_IGWLOG_001)
             INITSIZE(8192)
             SIZE(16384)
             PREFLIST(CF02,CF01)
             ALLOWAUTOALT(YES)
             DUPLEX(ALLOWED)

         STRUCTURE NAME(LOG_IGWSHUNT_001)
             INITSIZE(8192)
             SIZE(16384)
             PREFLIST(CF01,CF02)
             ALLOWAUTOALT(YES)
             DUPLEX(ALLOWED)

         STRUCTURE NAME(LOG_FORWARD_001)
             INITSIZE(8192)
             SIZE(16384)
             PREFLIST(CF01,CF02)
             ALLOWAUTOALT(YES)
             DUPLEX(ALLOWED)

         STRUCTURE NAME(LOG_IGWLGLGS_001)
             INITSIZE(8192)
             SIZE(16384)
             PREFLIST(CF02,CF01)
             ALLOWAUTOALT(YES)
             DUPLEX(ALLOWED)
```

Figure 6-8 Defining list structures in the CFRM policy for DFSMStvs

6.9.2 Defining the log structures and logstreams in LOGR policy

The definitions for log structures and logstreams are done in the system LOGR policy. Their definitions refer to the list structures you defined in the previous step.

Figure 6-9 shows a sample JCL. In this job, the log structures are defined first. Then, the logstreams for SYSLOG and SHUNTLOG for three z/OS systems in our sysplex are defined in these log structures. Finally, the logstreams for FR.LOG and LOG.OF.LOGS are defined.

```
//TVSLOG JOB (999,POK),'LOGR RRS',CLASS=A,REGION=4M,
//             MSGCLASS=T,TIME=10,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//STEP1    EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
     DATA TYPE(LOGR) REPORT(YES)

  DEFINE STRUCTURE NAME(LOG_IGWLOG_001) LOGSNUM(5)
         MAXBUFSIZE(64000) AVGBUFSIZE(2048)

  DEFINE STRUCTURE NAME(LOG_IGWSHUNT_001) LOGSNUM(5)
         MAXBUFSIZE(64000) AVGBUFSIZE(2048)

  DEFINE STRUCTURE NAME(LOG_FORWARD_001) LOGSNUM(10)
         MAXBUFSIZE(64000) AVGBUFSIZE(2048)

  DEFINE STRUCTURE NAME(LOG_IGWLGLGS_001) LOGSNUM(1)
         MAXBUFSIZE(64000) AVGBUFSIZE(2048)

  DEFINE LOGSTREAM
         NAME(IGWTV063.IGWLOG.SYSLOG)
         STRUCTNAME(LOG_IGWLOG_001)
         LS_DATACLAS(SHARE33)
         HLQ(LOGR) MODEL(NO) LS_SIZE(1180)
         LOWOFFLOAD(15) HIGHOFFLOAD(95)
         STG_DUPLEX(YES) DUPLEXMODE(COND)
         STG_DATACLAS(SHARE33)
         DIAG(YES)

  DEFINE LOGSTREAM
         NAME(IGWTV064.IGWLOG.SYSLOG)
         STRUCTNAME(LOG_IGWLOG_001)
         LS_DATACLAS(SHARE33)
         HLQ(LOGR) MODEL(NO) LS_SIZE(1180)
         LOWOFFLOAD(15) HIGHOFFLOAD(95)
         STG_DUPLEX(YES) DUPLEXMODE(COND)
         STG_DATACLAS(SHARE33)
         DIAG(YES)

  DEFINE LOGSTREAM
         NAME(IGWTV065.IGWLOG.SYSLOG)
         STRUCTNAME(LOG_IGWLOG_001)
         LS_DATACLAS(SHARE33)
         HLQ(LOGR) MODEL(NO) LS_SIZE(1180)
         LOWOFFLOAD(15) HIGHOFFLOAD(95)
         STG_DUPLEX(YES) DUPLEXMODE(COND)
         STG_DATACLAS(SHARE33)
         DIAG(YES)
```

Figure 6-9 Defining the log structures and logstreams for DFSMStvs

The sample JCL is continued in Figure 6-10.

```
DEFINE LOGSTREAM
       NAME(IGWTV063.IGWSHUNT.SHUNTLOG)
       STRUCTNAME(LOG_IGWSHUNT_001)
       LS_DATACLAS(SHARE33)
       HLQ(LOGR) MODEL(NO) LS_SIZE(100)
       LOWOFFLOAD(15) HIGHOFFLOAD(95)
       STG_DUPLEX(YES) DUPLEXMODE(COND)
       STG_DATACLAS(SHARE33)
       DIAG(YES)

DEFINE LOGSTREAM
       NAME(IGWTV064.IGWSHUNT.SHUNTLOG)
       STRUCTNAME(LOG_IGWSHUNT_001)
       LS_DATACLAS(SHARE33)
       HLQ(LOGR) MODEL(NO) LS_SIZE(100)
       LOWOFFLOAD(15) HIGHOFFLOAD(95)
       STG_DUPLEX(YES) DUPLEXMODE(COND)
       STG_DATACLAS(SHARE33)
       DIAG(YES)

DEFINE LOGSTREAM
       NAME(IGWTV065.IGWSHUNT.SHUNTLOG)
       STRUCTNAME(LOG_IGWSHUNT_001)
       LS_DATACLAS(SHARE33)
       HLQ(LOGR) MODEL(NO) LS_SIZE(100)
       LOWOFFLOAD(15) HIGHOFFLOAD(95)
       STG_DUPLEX(YES) DUPLEXMODE(COND)
       STG_DATACLAS(SHARE33)
       DIAG(YES)

DEFINE LOGSTREAM
       NAME(IGWTVS.FR.LOG001)
       STRUCTNAME(LOG_FORWARD_001)
       LS_DATACLAS(SHARE33)
       HLQ(LOGR) MODEL(NO) LS_SIZE(100)
       LOWOFFLOAD(0) HIGHOFFLOAD(80)
       STG_DUPLEX(NO)

DEFINE LOGSTREAM
       NAME(IGWTVS.LOG.OF.LOGS)
       STRUCTNAME(LOG_IGWLGLGS_001)
       LS_DATACLAS(SHARE33)
       HLQ(LOGR) MODEL(NO) LS_SIZE(1180)
       LOWOFFLOAD(0) HIGHOFFLOAD(80)
       STG_DUPLEX(NO)
```

Figure 6-10 Defining the log structures and logstreams for DFSMStvs (continued)

Guideline: Generally, specify DIAG(YES) for any DFSMStvs undo and shunt logs. This option causes the system logger to create a memory dump in the event of errors such as a lost block.

6.9.3 DASD staging data sets

In the system logger environment, log entries are written to both the coupling facility and the IXGLOGR data space before offloading. So, if a coupling facility or MVS failure occurs, log data is not lost. However, if both the coupling facility and MVS fail at the same time, log data is lost because the coupling facility and the data space are not permanent media.

If you want to protect against this double failure, you can keep the log data in DASD staging data sets. You can also protect the log data from site failures by staging the log data in disk volumes, and using a remote copy solution to mirror your volumes to an alternate site.

If you decide to use DASD staging data sets, you need one per logstream. A DASD staging data set is a VSAM linear data set, and should be managed by SMS. The following recommendations apply to defining logstreams:

- Define STG_DUPLEX(YES) and DUPLEXMODE(UNCOND) for logstreams that are associated with the system log and the forward recovery logs. This option ensures that the system logger duplexes to DASD staging data sets.
- Define each DASD staging data set to be at least the same size as the associated logstream's share of the coupling facility. However, round the average block size up to 4 K.

For more information about System Logger and staging data sets, see z/OS MVS *Setting Up a Sysplex*, SA22-7615.

6.9.4 Setting up RRS

RRS provides a global syncpoint manager that any resource manager on z/OS can use. RRS is increasingly becoming a prerequisite for new resource managers, and for new capabilities in existing resource managers. Rather than having to implement their own two-phase commit protocol, these products can use the support that is provided by RRS.

DFSMStvs uses the same logging mechanism as VSAM RLS, and this support is managed from RRS. To update a data set, DFSMStvs uses functions that are managed from RRS.

Use the following steps to implement RRS on z/OS. Some of them are optional, but should be done for recovery actions:

1. Define the RRS logstreams
2. Establish the priority for the RRS address space
3. Define RRS as a subsystem
4. Create procedures to start RRS
5. Set up automation to restart RRS
6. Set up RRS security definitions
7. Enable RRS ISPF panels (optional)
8. Set up RRS component traces.

For more information about the setup of RRS, see Chapter 3 in *ABCs of z/OS System Programming: Volume 5*, SG24-6985.

6.9.5 Setting up security definitions

You can define a RACF generic profile to protect all the logstreams that are referenced by a DFSMStvs instance. For example, issue the RACF command:

```
RDEFINE LOGSTRM tvsname.** UACC(NONE).
```

The *tvsname* is in the form `IGWTVnnn`.

Figure 6-11 shows an example of how to give access to three categories of users. In these examples, `smsvsam_userid` is the user ID of the VSAM RLS address space in which DFSMStvs makes its calls to the system logger.

```
PERMIT tvsname.** CLASS(LOGSTRM) ACCESS(UPDATE)
       ID(smsvsam_userid)
PERMIT tvsname.** CLASS(LOGSTRM) ACCESS(READ)
       ID(authorized_browsers)
PERMIT tvsname.** CLASS(LOGSTRM) ACCESS(UDPATE)
       ID(archive_userid)
```

Figure 6-11 RACF permit to logstream resource

DFSMStvs also writes to forward recovery logstreams and a log of logs that is used to optimize forward recovery. To protect these logstreams, code the appropriate RDEFINE commands and PERMIT commands for each of them.

For all forward recoverable data sets that DFSMStvs accesses, grant DFSMStvs access to the following logs:

- The log of logs
- The forward recovery logstreams

Each of these logstreams requires update authority for the SMSVSAM user ID defined in RACF. Figure 6-12 shows an example.

```
RDEFINE LOGSTRM FORWARD.RECOVERY.** UACC(NONE)
RDEFINE LOGSTRM FR.LOG.** UACC(NONE)
RDEFINE LOGSTRM LOG.OF.LOGS UACC(NONE)
PERMIT FOWARD.RECOVERY.** CLASS(LOGSTRM) ACCESS(UPDATE)
       ID(smsvsam_userid)
PERMIT FR.LOG.** CLASS(LOGSTRM) ACCESS(UPDATE)
       ID(smsvsam_userid)
PERMIT LOG.OF.LOGS CLASS(LOGSTRM) ACCESS(UPDATE)
       ID(smsvsam_userid)
```

Figure 6-12 Protecting forward recovery and log of logs

6.9.6 Modifying IGDSMSxx parmlib member

These are the IGDSMSxx parameters that are specifically related to DFSMStvs:

- SYSNAME(*sysname*[,*sysname*]...)
  - Identifies the systems on which you want to run DFSMStvs.
  - Specifies the names of the systems to which the DFSMStvs instance names of the TVSNAME parameter apply.

    SMS examines the specified system names and compares them to the system names in the CVT. When it finds a match, SMS stores the value of the TVSNAME parameter in the matching position as the DFSMStvs instance name for the system.

- TVSNAME(*nnn*[,*nnn*]...)
  - Specifies the identifiers of the DFSMStvs instances on the systems in the same order you specified in the SYSNAME parameter. These identifiers are appended to IGWVT to form the DFSMStvs instance name, such as:

    ```
    SYSNAME(SC63,SC64,SC65,SC70)
    TVSNAME(63,64,65,70)
    ```

    This combination of SYSNAME and TVSNAME creates the following instances of DFSMStvs:

    - IGWVT063 on system SC63
    - IGWVT064 on system SC64
    - IGWVT065 on system SC65
    - IGWVT070 on system SC70
  - The number of sysnames and the number of tvsnames must be the same.
- TV_START_TYPE({WARM|COLD}[,{WARM|COLD}]...)

  Specifies the start type for the single DFSMStvs instances in the same order as they are listed in the TVSNAME parameter.
- AKP(*nnn*[,*nnn*]...)
  - Specifies the activity keypoint trigger values for the DFSMStvs instances in the same order as they are listed in the TVSNAME parameter.

    Use the AKP parameter to specify the key pointing frequency. Review the activity keypoint (AKP) frequency that is defined for each DFSMStvs instance. The larger the value, the more coupling facility space you need for the system logs. But do not set AKP so low that transactions last longer than an activity keypoint interval. DFSMStvs manages the size of the system log by deleting old, completed units of work.
- LOG_OF_LOGS(*logstream*)

  Specifies the name of the logstream that is used as the log of logs.
- QTIMEOUT({*nnn*|300})

  Specifies the amount of time the DFSMStvs quiesce exits allowed to elapse before deciding that a quiesce event cannot be completed successfully.
- MAXLOCKS({*max*|0},{*incr*|0})
  - *max* is the maximum number of unique lock requests that a single unit of recovery can make. When this value is reached, the system issues a IGW859I warning message.

    This message can help you identify a job that is not issuing commits frequently enough or that has a logic problem. A job doing so often causes performance problems for other users of DFSMStvs.
  - *incr* is an increment value. After max is reached, each time the number of locks increases by the incr value, another warning message is issued.

6.10 Application requirements for DFSMStvs

In order for an application to participate in transactional recovery, it must first understand the concept of a transaction. It is not a good idea to modify an existing batch job to use DFSMStvs with no further changes. Doing so causes the entire job to be seen as a single transaction. As a result, locks would be held and log records would need to exist for the entire life of the job. This configuration can cause a tremendous amount of contention for the locked resources. It might also cause performance degradation as the undo log becomes excessively large.

To use the DFSMStvs capabilities, break the application processing down into a series of transactions. The application should issue frequent sync points by running RRS commit and backout processing (MVS callable services SRRCMIT and SRRBACK).

Batch RLS applications that open recoverable data sets on z/OS with the DFSMStvs feature installed should be modified to add SRRCMIT and SRRBACK interfaces. SRRCMIT and SRRBACK either commit or backout the unit of recovery (UR) provided by SMSVSAM on behalf of the VSAM RLS application.

Explicitly committing or backing out the UR releases record level locks in a timely fashion. Failure to do so can affect other sharers of the data set. SMSVSAM implicitly issues a commit or backout at EOT, if the VSAM application fails to do so.

Figure 6-13 shows what a sample application explicitly committing looks like.

```
//ddname  DD    DSN=Recoverabledatasetname,DISP=SHR
//step1   EXEC  PGM=vsamrlspgm
Begin JOB Step   ----------------------- No locks held
OPEN  ACB MACRF=(RLS,OUT)
(UR1)
GET UPD record 1------------------------ Obtain an exclusive lock on record 1
PUT UPD  record 1 ---------------------- Lock on record 1 remains held
GET repeatable read record n------------ Obtain a shared lock on record n
PUT ADD record n+1---------------------- Obtain an exclusive lock on record n+1
GET UPD record 2 ----------------------- Obtain an exclusive lock on record 2
PUT UPD record 2 ----------------------- Lock on record 2 remains held
Call SRRCMIT --------------------------- Commit changes, all locks released .
CLOSE
End of JOB Step
```

Figure 6-13 Explicit commit

Figure 6-14 shows an example of explicit backout.

```
//ddname  DD    DSN=Recoverabledatasetname,DISP=SHR
//step1   EXEC  PGM=vsamrlspgm
Begin JOB Step   ----------------------- No locks held
OPEN  ACB MACRF=(RLS,OUT)
(UR1)
GET UPD record 1------------------------ Obtain an exclusive lock on record 1
PUT UPD  record 1 ---------------------- Lock on record 1 remains held
GET repeatable read record n------------ Obtain a shared lock on record n
PUT ADD record n+1---------------------- Obtain an exclusive lock on record n+1
GET UPD record 2 ----------------------- Obtain an exclusive lock on record 2
PUT UPD record 2 ----------------------- Lock on record 2 remains held
Call SRRBACK --------------------------- Undo changes, all locks released .
CLOSE
End of JOB Step
```

Figure 6-14 Explicit backout

For more information about how to create new application programs and migrate the existing ones to use the commit and backout protocols of RRS and be eligible to use DFSMStvs, see *Transactional VSAM Application Migration Guide*, SG24-6972.

For more information about the Application_Commit_UR (SRRCMIT) and Application_Backput_UR (SRRBACK) services, see *z/OS MVS Programming: Callable Services for High-Level Languages*, SA22-7613.

6.11 Operational considerations

This section addresses some commands and shows some considerations about using DFSMStvs.

6.11.1 Quiescing a data set

VSAM RLS provides the capability for DFSMS and application programs to initiate the quiesce of RLS and DFSMStvs activity against a sphere across the sysplex. Quiesce of a sphere is initiated by using the CICS CEMT SET DSN or VARY SMS operator command.

DFSMStvs supports quiesce by providing a quiesce exit. It also provides a mechanism for quiescing data sets by using the VARY SMS command. However, because DFSMStvs does not control opens and closes of data sets accessed by using it, it can support only a limited form of quiesce for close. The quiesce completes successfully only if the data set is not open for DFSMStvs access. If the data set is open for DFSMStvs access, the quiesce is rejected.

Perform these steps to quiesce a VSAM data set from TVS:

1. Display the jobs that are using the data set by using one of the following commands:
   ```
   DISPLAY SMS,DSNAME
   IDCAMS SHCDS LISTDS(dsname),JOBS
   ```
2. After you identify the jobs, allow them to either complete or cancel them.
3. Use the following command, or its CICS equivalent, to quiesce the data set:
   ```
   VARY SMS,SMSVSAM,SPHERE(datasetname),QUIESCE
   ```
4. Perform the operations which required the data set to be quiesced.
5. Enable the data set back for RLS and DFSMStvs use by using the following command, or its CICS equivalent command:
   ```
   VARY SMS,SMSVSAM,SPHERE(datasetname),ENABLE
   ```

6.11.2 Closing/deleting/renaming a data set with inflight UR

Closing the data set allows any of the following tasks to be done:

- The data set can be deleted
- The data set can be renamed
- A PERMITNONRLSUPDATE can be done
- The data set can be quiesced
- The locks that are associated with the data set can become retained

Do not delete or rename data sets with an outstanding inflight UR. If backout is required, UR is shunted.

Do not delete or rename data sets with shunted log records and retained locks. Doing so can cause loss of association between the data set and its log records and locks.

A new data set can be allocated with the old name.

6.11.3 Displaying DFSMStvs information

This section addresses some display commands to support DFSMStvs.

DISPLAY SMS,TRANVSAM,ALL

This command displays information about the instance of DFSMStvs on this system. It displays information about all systems in the sysplex when the ALL keyword is specified. The output includes the following information:

- The AKP trigger, which is the number of logging operations between the taking of keypoints.
- The status of this instance of DFSMStvs (initializing, active, quiescing, quiesced, disabling, disabled).
- How DFSMStvs started:
  - Cold start: The log data was not read, and any old data was discarded.
  - Warm start: The log data was read and processed.
- DFSMStvs status concerning RRS.
- The quiesce timeout value.
- All logs known to this instance of DFSMStvs, including the log of logs if one is in use.
- The number of active units of recovery.

Figure 6-15 is the output of the command from the test system.

```
IGW800I 16.31.01 DISPLAY SMS,TRANSACTIONAL VSAM,ALL

DISPLAY SMS,TRANSACTIONAL VSAM,ALL - SERVER STATUS
 System   TVSNAME  State   Rrs    #Urs     Start     AKP    QtimeOut
 -------- -------- ------ ----- -------- --------- -------- --------
 SC63     IGWTV063 DISED  UNREG        0 WARM/WARM     1000      300
 SC64     IGWTV064 ACTIVE REG          0 WARM/WARM     1000      300
 SC65     IGWTV065 ACTIVE REG          0 WARM/WARM     1000      300
 SC70     IGWTV070 DISED  UNREG        0 WARM/WARM     1000      300

DISPLAY SMS,TRANSACTIONAL VSAM,ALL LOGSTREAM STATUS
LogStreamName:  IGWTV063.IGWLOG.SYSLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC63      IGWTV063         Disabled   UnDoLog    DisConnected
LogStreamName:  IGWTV063.IGWSHUNT.SHUNTLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC63      IGWTV063         Disabled   ShuntLog   DisConnected
LogStreamName:  IGWTV064.IGWLOG.SYSLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC64      IGWTV064         Enabled    UnDoLog    Connected
LogStreamName:  IGWTV064.IGWSHUNT.SHUNTLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC64      IGWTV064         Enabled    ShuntLog   Connected
LogStreamName:  IGWTV065.IGWLOG.SYSLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC65      IGWTV065         Enabled    UnDoLog    Connected
LogStreamName:  IGWTV065.IGWSHUNT.SHUNTLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC65      IGWTV065         Enabled    ShuntLog   Connected
LogStreamName:  IGWTV070.IGWLOG.SYSLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC70      IGWTV070         Disabled   UnDoLog    DisConnected
LogStreamName:  IGWTV070.IGWSHUNT.SHUNTLOG
 System     TVSNAME         State      Type       Connect Status
 --------   --------      ------------ ---------- --------------
 SC70      IGWTV070         Disabled   ShuntLog   DisConnected
```

Figure 6-15 Output of D SMS,TRANVSAM, ALL

DISPLAY SMS,SHUNTED, {SPHERE(*sphere*)|UR({*urid*|ALL})}

This command displays the entries that are currently contained in the shunt logs of the systems in the sysplex. Entries are moved to the shunt log when DFSMStvs is unable to finish processing a sync point, for example, because of an I/O error. When a shunted entry exists, the locks that are associated with that entry are retained.

These types of information can be displayed in response to this command:

- When neither SPHERE or URID is specified, a list of systems in the sysplex and the number of units of recovery that that system has shunted are displayed.
- When the SPHERE keyword is specified, a list of shunted work for the sphere that is specified for all of the systems in the sysplex is displayed.
- When the URID keyword is specified, a list of shunted work for the unit of recovery that is specified for all of the systems in the sysplex is displayed.
- When ALL is specified, a list of shunted work for all shunted units of recovery for all the systems in the sysplex is displayed. To avoid flooding the console, DFSMStvs writes out 255 lines and then issues a WTOR to determine whether to continue.

If the error is correctable, the installation might choose to fix the problem and then request that DFSMStvs again. It attempts processing of the entry by issuing the SHCDS RETRY command. If the data set cannot be restored to a point where it is consistent with the log entries, it does not make sense to attempt processing of the log entry again. In this case, the installation might choose to discard the log entry by issuing the SHCDS PURGE command.

Figure 6-16 is the output of the command that shows that there are no shunted units of work currently in the DFSMStvs shunt log.

```
 IEE932I 452
  IGW803I 17.19.03 DISPLAY SMS,SHUNTED (Summary Data)
SysName   # Urid(s) SysName   # Urid(s) SysName   # Urid(s)
  --------  --------  --------  --------  --------  --------
  SC63             0  SC64             0  SC65             0
```

Figure 6-16 Output of the D SMS,SHUNTED command

DISPLAY SMS,JOB(jobname)

This command displays information about a particular job that is using DFSMStvs services on one of the systems in the sysplex. The output includes this information:

- The name of the current step within the job
- The current URID for the job
- The status of the unit of recovery (in-reset, in-flight, in-prepare, in-commit, in-backout, indoubt)

DISPLAY SMS,URID(*urid* |ALL)

This command displays information about a particular unit of recovery currently active within the sysplex. It can also display information about all units of recovery currently active on the system on which the command was issued on whose behalf DFSMStvs has performed any work. This parameter does not include information about work that has been shunted. You can use the DISPLAY SMS,SHUNTED command to display that information. This parameter also does not include information about units of recovery that might be in restart processing as a result of an earlier failure. This work is not considered to be currently active because it is not associated with any batch job. In addition, the units of recovery that are associated with the work end as soon as commit or backout processing for them can be completed. The output includes this information:

- The age of the unit of recovery
- The name of the job with which the unit of recovery is associated
- The name of the current step within the job

- The status of the unit of recovery (in-reset, in-flight, in-prepare, in-commit, in-backout, indoubt)
- The user ID associated with the job

Figure 6-17 shows the output from the command in the test system. It shows that there are no shunted units of recovery.

```
-D SMS,SHUNTED,urid(all)
 IEE932I 717
 IGW803I DFSMS REQUEST TO DISPLAY ALL TRANSACTIONAL VSAM
 SHUNTED URs
 WAS REJECTED,  THERE ARE NO SHUNTED URs ASSIGNED
 TO ANY TRANSACTIONAL VSAM INSTANCE IN THE SYSPLEX.
```

Figure 6-17 Output of the D SMS,URID(ALL) command

DISPLAY SMS,LOG(*logstreamid*|ALL)

This command displays information about a logstream that DFSMStvs is using on one of the systems in the sysplex. If you specify ALL, it displays information about all of the logs in use on the entire sysplex. The output includes this information:

- The status of the logstream (failed or available)
- Type of log (undo, shunt, forward recovery, or log of logs)
- The job name and URID of the oldest unit of recovery using the log
- A list of all DFSMStvs instances that are using the log

If information about a specific logstream is requested and the logstream is either a system log or a forward recovery log, the output includes the names of the jobs that use the logstream.

This command might be issued to determine why a logstream is increasing in size. If a unit of recovery is long running, DFSMStvs would be unable to delete any log blocks that contain data that are associated with the unit of recovery. This process in turn would make truncation of the logstream impossible.

Figure 6-18 shows the output of this command using the logstream name IGWTV064.IGWLOG.SYSLOG.

```
RESPONSE=SC63
 IEE932I 725
 IGW804I 18.16.28 DISPLAY SMS,LOG

 DISPLAY SMS,LOG      - LOG STREAM STATUS
   Name: IGWTV064.IGWLOG.SYSLOG     State: Enabled      Type: UnDo
   System   TVSNAME  JobName  Urid of Oldest Log Block
   -------- -------- -------- --------------------------------
   SC64     IGWTV064 **NONE** --------- NO ACTIVE UR ---------*

 DISPLAY SMS,LOG      - LOG STREAM USAGE
   LogStreamName: IGWTV064.IGWLOG.SYSLOG
   System   TVSNAME  JobName  JobName  JobName  JobName  JobName
   -------- -------- -------- -------- -------- -------- --------
   SC64     IGWTV064 **NONE**
 *OLDEST URID ACROSS ALL SYSTEMS IN THE SYSPLEX
```

Figure 6-18 Output of the D SMS,LOG(IGWTV064.IGWLOG.SYSLOG command

DISPLAY SMS,DSNAME(dsn)

For a given fully qualified data set name, this command displays the jobs currently accessing the data set by using DFSMStvs on the systems within the sysplex. Figure 6-19 shows that the specified data set is not open for DFSMStvs access.

```
D SMS,DSNAME(MHLRES3.VSAM.KSDS)
IEE932I 678
IGW805I DFSMS REQUEST TO DISPLAY TRANSACTIONAL VSAM USAGE OF
DATASET: MHLRES3.VSAM.KSDS WAS REJECTED.
DATASET NOT KNOWN TO TRANSACTIONAL VSAM.
```

Figure 6-19 Output of the D SMS, DSNAME(dsn) command

DISPLAY SMS,OPTIONS

This command displays all of the SMS parameters and their status at the time this command is issued. The display indicates this information:

- Whether each option is on or off
- What data sets are being used
- The size of regions
- The time interval for recording data
- All other parameter specifics

When DFSMStvs is running on the system, the output of this command includes DFSMStvs-related information. The output in Figure 6-20 shows only the DFSMStvs-related information

```
IGD002I 19:06:31 DISPLAY SMS 772
 /* This output has been edited to remove all non DFSMStvs related data*/
 LOCAL_DEADLOCK = 15     GLOBAL_DEADLOCK = 4
 REVERIFY = NO           DSNTYPE = PDS
 ACSDEFAULTS = NO        PDSESHARING = EXTENDED
 OVRD_EXPDT = NO         SYSTEMS = 8
 PDSE_HSP_SIZE = 0MB   PDSE1_HSP_SIZE = 256MB
 USE_RESOWNER = YES      RLS_MAX_POOL_SIZE = 100MB
 RLSINIT = YES           RLSTMOUT = 0
 COMPRESS = GENERIC      LOG_OF_LOGS = IGWTVS.LOG.OF.LOGS
 QTIMEOUT = 300          TVSNAME = 063
 AKP = 1000              TV_START_TYPE = WARM
 MAXLOCKS = (0,0)
```

Figure 6-20 Modified output from the D SMS,OPTIONS command

6.11.4 Controlling DFSMStvs processing

You can use the VARY SMS command to change the status for DFSMStvs in these ways:

- Change the state of a DFSMStvs instance or of all DFSMStvs instances in the sysplex
- Change the state of a logstream 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

Changing the state of DFSMStvs

You can use the V SMS,TRANVSAM command to change the state of a single DFSMStvs instance, or all DFSMStvs instances.

The syntax of the V SMS,TRANVSAM command is as follows:

```
V SMS,{TRANVSAM({tvsname|ALL}){,{QUIESCE|Q}} }
      {                       {,{ENABLE|E }} }
      {                       {,{DISABLE|D}} }
```

- QUIESCE

  DFSMStvs completes processing of any units of recovery that are in progress, but does not accept any new ones. DFSMStvs is unavailable until a VARY SMS,TRANVSAM(*tvsname*),ENABLE command is issued.

- ENABLE

  DFSMStvs begins accepting new units of recovery for processing.

- DISABLE

  DFSMStvs immediately stops processing new work requests, including units of recovery that are currently in progress. When the last data set that is open for DFSMStvs access is closed, DFSMStvs retains locks and unregisters with RRS. It is unavailable until a VARY SMS,TRANVSAM,ENABLE command is issued.

The *tvsname* that you use in this command is the suffix of the IGWTVxxx instance that you want to change the state. For example, if you want to quiesce an instance that is called IGWTV063, issue the following command:

```
V SMS,TRANVSAM(063),Q
```

Changing the state of a logstream

You can use the V SMS,LOG(logstreamid) command to quiesce, enable, or disable DFSMStvs access to the specified logstream.

The syntax of this command is:

```
V SMS,{LOG(logstreamid){,{QUIESCE|Q}} }
      {                {,{DISABLE|E}} }
      {                {,{ENABLE|E}} }
```

- QUIESCE

  DFSMStvs completes the processing of any in-progress units of recovery using the log stream. However, it does not accept any new ones that would require the log stream, except for the log of logs. If the log is a DFSMStvs system log (undo or shunt log), it becomes quiesced when all units of recovery that are using DFSMStvs complete and any open data sets are closed. If the log is a forward recovery log, it becomes quiesced when the last data set that is open for output in DFSMStvs is closed. If the log is a log of logs, it becomes quiesced when the last forward-recoverable data set that is open for output in DFSMStvs mode, for which a tie-up record was written to the log of logs, is closed.

- DISABLE

  DFSMStvs immediately stops using the log stream. This process can prevent completion of commit or backout for units of recovery. Those units of recovery are shunted, if shunting them does not require reading or writing the now disabled log.

- ENABLE

  DFSMStvs begins accepting new units of recovery that use the log stream for processing. If DFSMStvs work was left incomplete when DFSMStvs processing was stopped, DFSMStvs completes that work as part of its restart processing.

For example, if you want to quiesce the undo log for the DFSMStvs IGWTV064 instance, issue this command:

```
V SMS,LOG(IGWTV064.IGWLOG.SYSLOG),Q
```

You can find the name of any undo log in the output of the D SMS,TRANVSAM,ALL command as shown in Figure 6-15 on page 334. After you issue the command to quiesce the logstream, you would see its state in the output of D SMS,TRANVSAM command as shown in Figure 6-21.

```
 LogStreamName:  IGWTV064.IGWSHUNT.SHUNTLOG
   System     TVSNAME         State      Type       Connect Status
   --------   --------      ------------ ---------- --------------
   SC64      IGWTV064         Quiesced   ShuntLog   DisConnected
```

Figure 6-21 Quiesced logstream

Attention: Do not use this DISABLE option without first quiescing the logstream unless the logstream is damaged or has errors that cannot be corrected.

Quiescing a VSAM data set

Use the V SMS,SMSVSAM,SPHERE(*datasetname*) command to quiesce the specified data set from DFSMStvs and RLS mode, or to enable them back to RLS and DFSMStvs processing.

The syntax of this command is:

```
V SMS,{SMSVSAM,SPHERE(datasetname){,{QUIESCE|Q}} }
      {                           {,{ENABLE|E }} }
```

For more information about this command and the Quiesce interface, see 6.11.1, “Quiescing a data set” on page 332.

PEERRECOVERY

Use the V SMS,TRANVSAM(tvsname),PEERRECOVEY to start or stop peer recovery processing for a failed instance of DFSMStvs. This command applies only to the system on which it is issued. That system is responsible for running all peer recovery processing for the failed DFSMStvs instance.

The syntax of this command is:

```
V SMS,{TRANVSAM({tvsname),PEERRECOVERY{,{ACTIVE|A }}}
      {                               {,ACTIVEFORCE }}
      {                               {,{INACTIVE|I}}}
```

- ACTIVE

  This system begins peer recovery processing on behalf of the specified failed instance of DFSMStvs. If the failed instance of DFSMStvs was not disabling or disabled because of an operator command, the system runs the necessary initialization. It then starts tasks to perform any work that was left incomplete by a system failure.

- ACTIVEFORCE

  The system begins peer recovery processing on behalf of the specified failed instance of DFSMStvs, regardless of the failed instance's status.
- INACTIVE

  This system stops processing peer recovery work on behalf of the specified instance of DFSMStvs.

Peer recovery is the process of completing the work that was left in an incomplete state by the failure of an instance of DFSMStvs by another instance of DFSMStvs. The only function of peer recovery is to complete that work and then end. A peer recovery instance of DFSMStvs does not accept any new work.

Peer recovery occurs only in cases of system failure, not merely when DFSMStvs fails. The following rules apply to peer recovery:

- Peer recovery occurs only when both DFSMStvs and the system on which it was running fail. Peer recovery does not occur when the DFSMStvs instance failed, but the system continued to run. In this case, either DFSMStvs automatically restarts, or the DFSMStvs instance was stopped and was not meant to be restarted.
- All resource managers that shared interest in units of recovery restart on the same system. For this reason, DFSMStvs uses the automatic restart manager (ARM) to manage the grouping.

  If the installation is not using ARM, or if ARM is unavailable, you can manually initiate peer recovery by issuing the following command:

  ```
  V SMS,TRANVSAM(tvsname),PEERRECOVERY,ACTIVE
  ```

  For example, the following command starts PEERRECOVERY on behalf of DFSMStvs instance IGWTV**064** if the system where it runs is down, or does not have ARM implemented:

  ```
  VARY SMS,TRANVSAM(064),PEERRECOVERY,ACTIVE
  ```

  Issue the VARY SMS command on another system. That system then runs peer recovery for the instance that is specified on the command, and any instances for which that instance was running peer recovery.

  Normally, if a DFSMStvs was disabling or disabled because of an operator command, peer recovery does not run. This limitation is because that instance of DFSMStvs was told to come down and not restart. In this case, if you want peer recovery to occur, issue the following command:

  ```
  VARY SMS,TRANVSAM(tvsname),PEERRECOVERY,ACTIVEFORCE
  ```
- A peer recovery DFSMStvs instance is only started if there is a DFSMStvs instance that is running on the system. If DFSMStvs is not started on the system, peer recovery does not run.
- Peer recovery starts asynchronous tasks to process outstanding units of recovery in parallel. As the tasks complete, more tasks are started until all the outstanding units of recovery are processed or you issue the following command:

  ```
  VARY SMS,TRANVSAM(tvsname),PEERECOVERY,INACTIVE
  ```

  If you issue that command, tasks that are already running are allowed to complete, and then peer recovery processing ends.
- Peer recovery is allowed to run if the state of the failed DFSMStvs instance was quiescing, enabling, or enabled. It is allowed because peer recovery would complete only the quiesce process. It is not allowed if the state was quiesced because, in this case, there should be

no work to do. It is also not allowed if the state was disabled. This state implies that the installation did not want the DFSMStvs instance to do any work. If you want peer recovery to be run on behalf of a DFSMStvs instance that was disabling or disabled, you must use the VARY SMS command with the ACTIVEFORCE keyword.

6.12 Changes to support DFSMStvs

This section addresses modifications that you must be aware of to implement DFSMStvs.

6.12.1 Changes to job control language (JCL)

The following are JCL changes to support DFSMStvs:

- RLSTMOUT

  This subparameter of the EXEC statement specifies a timeout value for DFSMStvs requests for required locks.

  - CRE

    This subparameter of the RLS statement obtains a shared lock for VSAM record-level sharing (RLS).

For more detailed information about these changes, see *z/OS DFSMStvs Planning and Operating Guide*, SC26-7348.

6.12.2 Changes to IDCAMS

There are numerous changes and additions to IDCAMS commands. Here are some of the modifications:

- ALLOCATE

  These changes were made to the ALLOCATE command:

  - 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.

  - DATACLASS CISIZE

    For DFSMStvs, specification of n*2K avoids wasting space in the coupling facility cache structure.

  - SHAREOPTIONS

    When you use DFSMStvs access, DFSMS assumes that the value of SHAREOPTIONS is (3,3).

- ALTER

  The following changes were made to the ALTER command.

  - BWO(TYPECICS)

    The TYPECICS option of the BWO parameter specifies BWO in a DFSMStvs environment. For RLS processing, this option activates BWO processing for DFSMStvs.

- LOG(UNDO)

  This option 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)

  This option specifies that changes to the sphere accessed in DFSMStvs mode can be backed out, and is also forward recoverable by using external logs. DFSMStvs considers the sphere recoverable. LOGSTREAMID must also be defined.

- LOGSTREAMID

  This option changes or adds the name of the DFSMStvs forward-recovery logstream, for all components in the VSAM sphere.

► DEFINE ALTERNATEINDEX

The following changes were made to the DEFINE ALTERNATEINDEX command:

- BUFFERSPACE

  When you use DFSMStvs access, DFSMS ignores the BUFFERSPACE parameter.

- CONTROLINTERVALSIZE

  For DFSMStvs, specification of n*2K avoids wasting space in the coupling facility cache structure.

- KEYRANGES

  You cannot open the keyrange data sets for RLS or DFSMStvs processing because DFSMS no longer supports this parameter (as of V1R3).

- SHAREOPTIONS

  When you use DFSMStvs access, DFSMS assumes that the value of SHAREOPTIONS is (3,3).

- WRITECHECK

  When you use DFSMStvs access, DFSMS ignores the WRITECHECK parameter.

► DEFINE CLUSTER

The following changes were made to the DEFINE CLUSTER command:

- BUFFERSPACE

  When you use DFSMStvs access, DFSMS ignores the BUFFERSPACE parameter.

- BWO(TYPECICS)

  The TYPECICS option of the BWO parameter specifies BWO in a DFSMStvs environment. For RLS processing, this parameter activates BWO processing for DFSMStvs.

- CONTROLINTERVALSIZE

  For DFSMStvs, specification of n*2K avoids wasting space in the coupling facility cache structure.

- KEYRANGES

  You cannot open the keyrange data sets for RLS or DFSMStvs processing because DFSMS no longer supports this parameter (as of V1R3).

- LOG(UNDO)

  This parameter 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)

    This option specifies that changes to the sphere accessed in DFSMStvs mode can be backed out and also forward recoverable by using external logs. DFSMStvs considers the sphere recoverable. LOGSTREAMID must also be defined. If you use LOG(NONE), DFSMStvs considers the sphere to be unrecoverable.
  - LOGSTREAMID

    Changes or adds the name of the DFSMStvs forward-recovery logstream for all components in the VSAM sphere.
  - SHAREOPTIONS

    When you use DFSMStvs access, DFSMS assumes that the value of SHAREOPTIONS is (3,3).
  - WRITECHECK

    When you use RLS or DFSMStvs access, DFSMS ignores the WRITECHECK parameter.
- DEFINE PATH

  The following changes were made to the DEFINE PATH command:
  - NOUPDATE

    This parameter has the same meaning for DFSMStvs as it does for RLS.
- SHCDS

  The following are the changes that were made to the SHCDS command.
  - LISTDS

    A new, optional JOBS keyword that returns a list of the jobs that currently access the data set in DFSMStvs mode.
  - LISTSHUNTED

    This option lists information about work that was shunted because of an inability to complete a sync point (commit or backout) for a data set, a unit of recovery, or all shunted units of recovery.
  - PURGE

    This parameter discards log entries and releases the associated locks. It is for use when a data set is damaged and you cannot restore it to a state that is consistent with the log entries.
  - RETRY

    This option retries the sync point. It is for when you can restore a data set to a state that is consistent with the log entries.

6.12.3 Macros that have been changed to support DFSMStvs

Several macros have been changed to add new subparameters in support of DFSMStvs. The following macros that have been changed or that you need to understand if your application is going to be DFSMStvs enabled:

- The ACB macro
- The GENCB macro

For more information, see *z/OS DFSMStvs Planning and Operating Guide*, SC26-7348.

7

VSAM problem determination and recovery

This chapter provides some general tips on VSAM problem determination. It addresses a number of common VSAM problems and explains how to recover from them. It also includes suggestions on how to avoid problems and links to useful documentation that assists you with your problem determination and recovery.

This chapter includes the following sections:

7.1 VSAM problem determination hints and tips

In the complex environment of computing today, users have an array of tools and disciplines at their disposal to aid problem determination and recovery of VSAM spheres and data sets. z/OS writes numerous records like SMF, LOGREC, and syslog that can be used to debug a problem and recover data.

Your data is one of the most valuable assets of your business. The challenge to save your data is to use all the error information available to answer these three questions:

- What caused this problem?
- How can you recover your data?
- What you can do to avoid these problems in the future?

This section provides you some general tips on what to look for and how to determine what caused the problem and how to recovery from a problem.

7.1.1 Checking your VSAM data set

A number of IDCAMS commands are available to check your VSAM data sets:

- EXAMINE
- VERIFY
- DIAGNOSE
- DFSMSdss PRINT command
- LISTCAT

EXAMINE

You can use the EXAMINE IDCAMS command to analyze and report on the structural integrity of the index and data components of these objects:

- A key-sequenced data set cluster (KSDS)
- A variable-length relative record data set cluster (VRRDS)

Any problems with the VSAM data set are reported by one of the IDCxxxxx messages. For more information about these messages, see MVS System Messages, Volume 3 (GDE - IEB).

For more information about the EXAMINE command, see 7.4.1, “EXAMINE command” on page 363.

VERIFY

The VERIFY command causes a catalog to correctly reflect the end of a VSAM data set after an error occurs while closing a VSAM data set. The error might cause the catalog to be incorrect. For more information about this command, see 7.4.3, “VERIFY command” on page 364.

DIAGNOSE

The DIAGNOSE command can be used to scan a basic catalog structure (BCS) or a VSAM volume data set (VVDS) to validate the data structures and detect structure errors. For more information, see 7.4.2, “DIAGNOSE command” on page 364 for more details.

DFSMSdss PRINT command

With the PRINT command, you can print these data:

- A single volume non-VSAM data set, as specified by a fully qualified name. You must specify the volume where the data set is, but you do not have to specify the range of tracks it occupies.
- A single-volume VSAM data set component (not cluster). The component name that is specified must be the name in the VTOC, not the name in the catalog.
- Ranges of tracks.
- All or part of the VTOC. The VTOC location does not need to be known.

LISTCAT

You can list the catalog entries by using the LISTCAT command. For more information, see 7.6, “SMF record types that are related to VSAM data sets” on page 372.

7.1.2 z/OS system messages

z/OS components issue messages with the IEA prefix, associated with data set allocation, master scheduler functions, and RTM. The IOS prefix is for IOS functions. Usually the following messages follow abends:

- IEC070I — RC32, RC202, RC104, or RC203
- IOS000I — Command reject (IOS errors in general)

7.1.3 System LOGREC messages

Often the system recovers from a problem without your knowledge, but this process generates error records in the LOGREC. If you suspect a problem, LOGREC can provide valuable information.

7.1.4 GTF CCW traces

You can collect GTF CCW traces to get detailed information about I/Os to VSAM data sets. These traces can be formatted by using IPCS. For a sample JCL to collect a GTF trace, see Example 7-3 on page 363.

7.1.5 What can you get from the SMF records?

You can get information from SMF record types 60 to 69 to analyze VSAM problems. SMF record type 42 contains information about VSAM RLS statistics. For more information about these SMF records, see 7.6, “SMF record types that are related to VSAM data sets” on page 372. You can use the SMF 64 sample program that is described in that section.

7.2 Common VSAM problems

The section addresses common problems that might affect the processing and the existence of your VSAM data sets. To simplify your search for a solution to each problem, each section includes the three “Whats”:

- What happened to cause this problem?
- What must you do to recover your data?
- What must you do to avoid this problem in the future?

Broken data sets can be caused by many different circumstances, including user errors. When you are diagnosing these types of problems, first identify what is actually wrong with the data set. The first sign of a problem is the VSAM or the z/OS system messages. A single error can often generate numerous messages. Focus your attention on the return code that is presented and the companion explanation. This return code is the one passed by the system component that first encountered the error. In most cases, you need more documentation. For more information, see 7.3, "What documentation to collect" on page 362.

This section groups the errors by categories. However, some of the categories overlap and even interact with others. For example, a bad channel program can be caused by an improper sharing, which later caused structural damage.

7.2.1 Lack of virtual storage

The following messages might indicate a lack of virtual storage:

- `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code`
  - 136 (Close): Not enough virtual storage was available in the address space of the program for a work area for Close.
  - 132 (Open): One of the following errors occurred:
    - Not enough storage is available for work areas.
    - The format-1 DSCB or the catalog cluster record is incorrect.
  - 136 (Open): Not enough virtual storage space is available in the address space of the program for work areas, control blocks, or buffers.
  - 40 (I/O): There is insufficient virtual storage in the user's address space to complete the request.
- `IEC161I 001 [(087)]-ccc,jjj, sss,ddname,dev,ser,xxx, dsname,cat`

What happened?

An abend occurred because of a lack of virtual storage. In this case, a symptom memory dump might be included.

What to do for recovery?

Because the data set processing was interrupted (abended) in apparently unknown circumstances, there are two cases:

- If VSAM data set is being accessed by a subsystem as CICS, CICS was doing the sync point journaling. Therefore, use CICS to recover your data by rolling it back.
- If the VSAM data set is being accessed by your program, correct the virtual storage problem and rerun the program (if possible), or restore the backup and rerun the job.

Sometimes, the message is not the result of an abend. It can be an alert as with IEC161I, where the BLDVRP macro indicates that there was not enough virtual storage to satisfy the request that was done by System-Managed Buffering (SMB). SMB gets the available storage, and processing goes on.

What to do to avoid future problems?

For more information, see 4.4.12, "Region size" on page 146. If possible, increase your region below or above, or decrease the common area below 16 MB, or force your software to be in R31 mode.

Determine whether the VSAM buffers and their control blocks are below or above the 16-MB line. If below, move them above the line with integrity. For more information, see “Locating VSAM buffers above 16 MB” on page 163.

7.2.2 Initial loading problems

The following messages can indicate initial load problems:

- `IDC3308I ** DUPLICATE RECORD xxx`

  The output data set of a REPRO command already contains a record with the same key or record number.
- `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code`
  - 8 (I/O): 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 (I/O): This message can indicate the following error conditions:
    - You attempted to store a record out of ascending key sequence in skip-sequential mode
    - The record had a duplicate key
    - For skip-sequential processing, your GET, PUT, and POINT requests are not referencing records in ascending sequence
    - For skip-sequential retrieval, the key requested is lower than the previous key requested
    - For shared resources, the buffer pool is full
  - 116 (I/O): During initial data set loading, GET, POINT, ERASE, direct PUT, and skip-sequential PUT with OPTCD=UPD are not allowed. Initial data set loading is when records are being stored in the data set the first time it is opened. During initial data set loading, for initial loading of a relative record data set, the request was other than a PUT insert.

What happened?

These messages point to problems with initial load or mass insertion (also called skip sequential) of a VSAM cluster.

Initial load of your data set can be done by IDCAMS REPRO or by your program. For more information, see 4.4.11, “Initial load option” on page 145.

When you are loading a VSAM KSDS data set, the logical records must be sorted in a key sequenced order. No out-of-sequence or duplicated keys are allowed. Refer to the messages for a more detailed explanation about which of these requirements were not fulfilled.

If the *duplicated keys* message applies to the initial load of an alternate index (AIX) cluster, remember that AIX can have duplicated keys. In this case, do not use the UNIQUE parameter, which would definitely cause this error.

Mass insertion resembles initial load in the sense that all the added logical records also must be sorted by a key field. The difference is that in mass insertion, you are sequentially loading logical records to a data set that already contains data.

What to do for recovery?

There no need for recovery. Your data is saved in the input file. You must sort and then rerun the program.

What to do to avoid future problems?

After you correct the error, introduce a procedure in production to avoid having the same error again.

7.2.3 Mismatch between catalog and data set

The following error messages can indicate a mismatch between catalog and data set information:

- `IDC3351I ** VSAM OPEN RETURN CODE IS 108`

  108: Attention message: the time stamps of a data component and an index component do not match. This message indicates that either the data or the index was updated separately from the other. Check for possible duplicate VVRs.

- `IDC3351I data set IS ALREADY OPEN FOR OUTPUT OR WAS NOT CLOSED CORRECTLY`
- The data set is already OPEN for output by a user on another system, or was not previously closed.
- `IDC11709I DATA HIGH-USED RBA IS GREATER THAN HIGH-ALLOCATED RBA`

  The data component high-used relative byte address is greater than the high-allocated relative byte address. Supportive messages display pertinent data, and processing continues.

- `IDC11712I DATA HIGH-ALLOCATED RBA IS NOT A MULTIPLE OF CI SIZE`

  The high-allocated relative byte address is not an integral multiple of the control interval size.

- `IDC11727I INDEX HIGH-USED RBA IS GREATER THAN HIGH-ALLOCATED RBA`

  The index component high-used relative byte address is greater than the high-allocated relative byte address.

- `IDC3350I synad[SYNAD] NO RECORD FOUND from VSAM`

What happened?

The data set might be intact, but the catalog information that describes the data set has mismatch problems. This problem sometimes results in an open failure.

The following are the most common discrepancies between the catalog and cluster:

- There were different time stamps between index and data components.
- HURBA and HARBA were not correctly updated, usually caused by an abend, without a normal close.
- The open-for-output bit is on for a closed cluster. This is usually caused by an abend, without a normal close, or a task accessing from another system. For more information, see 7.4.5, “Abend task scenario” on page 368.
- RBAs fields in the VVR do not match the data set attributes, as in these cases:
  - If the RBA of the high-level index CI is corrupted, you cannot perform direct requests against the data set.
  - If the RBA of the sequence set index CI is corrupted, you cannot perform sequential access.

  These last two discrepancies are not covered here. For more information, see 7.2.6, “Structural damage” on page 354.

A LISTCAT output (or CSI report) can help with the documentation for problem determination.

What to do for recovery?

IDCAMS VERIFY is a requirement in this type of problem. In this case, VERIFY can correct HURBA and verify the open-for-output bit.

- Different time stamps: Open does not abend your task, so continuation or abend of the application depends on the program that issues the OPEN. In general, it keeps processing, but another problem is likely to occur.

  Run VERIFY to document the mismatch, and then run EXAMINE to ensure that no structural damage exists for KSDS and VRRDS, with a test for the index option only (INDEXTEST).

  Completion of EXAMINE without error proves that there are no structural damages. If the index component shows damage, it must be restored before further use. For more information, see 7.4.4, “Broken Index scenario” on page 366. Note that EXAMINE might provide messages that contain only informational data that might not require restoring the cluster.
- HURBA and HARBA not correctly updated: If the HURBA is not updated, when the data set is then opened and the user's program attempts to process records beyond end-of-data or end-of-key range, a read operation results in a “no record found” error. A write operation might then write records over previously written records. To avoid this problem, you can use the VERIFY command, which corrects the catalog information.
- Open-for-output bit on for a closed cluster: At the next OPEN, VSAM implicitly issues a VERIFY command. This command detects an open-for-output indicator on, and issues an informational message (maybe the one that you are seeing) stating whether the VERIFY command is successful.

  If a subsequent OPEN is issued for output, VSAM turns off the open-for-output indicator at successful CLOSE. If the data set is opened for input, however, the open-for-output indicator is left on.

7.2.4 Hardware errors

The following messages indicate hardware errors:

- `IOS000I dev,chp,err,cmd,stat, dcbctfd,ser,mbe,eod, jobname,sens text`

  The system found an uncorrectable I/O error in device error recovery. Text is one of the following types:

  – Channel interface, or protocol error
  – Device has exceeded long busy timeout
  – Permanent error — volume fenced
  – Permanent error — device reported unknown message code = cde
  – Channel control, data, chaining, program, protection, interface check
  – Unable to obtain sense data from the device
- `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code`

  – 184 (Open): An uncorrectable I/O error occurred while VSAM was completing outstanding I/O requests.

  – 246 (Close): The compression management services (CMS) close function failed.

  – 184 (Open): An uncorrectable I/O error occurred while VSAM was completing an I/O request.

  – 245 (I/O): A severe error was detected by CMS during compression processing).

  - 246 (I/O): A severe error was detected by CMS during decompression processing.
  - 250 (I/O): A valid dictionary token does not exist for the compressed data set. The data record cannot be extracted.

What happened?

Hardware I/O errors usually mean that the I/O hardware (channel, controller, device) had a problem while running that I/O. For compressed VSAM data sets, you might have hardware problems with the CPU compression assist function.

Break down the message to get more details about the error. An IDCAMS LISTCAT (if possible) of the data set is also helpful to give the attributes of the file in the logical 3390/3380. These attributes include the physical record size, the device type, and the CCHH of all extents. LOGREC output is also valuable. A GTF CCW trace might be necessary to run DIAGNOSE on the problem. However, for such tools, you might need to re-create the situation.

VERIFY IGGCSIVS is a program from SYS1.SAMPLIB that accesses the Catalog Search Interface (CSI). It produces a list of data set names that are defined in a catalog on the specified volume. Such a list might be helpful in a recovery situation that affects that volume.

When you access a VSAM data set where a physical error was detected with VSAM macros, register 15 contains a return code equal to 12.

What to do for recovery?

For a media error, do not use ICKDFS to run an Analyze and Inspect function. The characteristics of the physical devices that make up the RAID devices family does not allow the use of the ICKDSF commands that run installation, media maintenance, and problem determination functions. These commands include Install, Analyze, and Inspect.

If the problem happens with the compress assist feature, run the program again, switching off data compression in the data class.

What to do to avoid future problems?

A solution for some media problems is to implement RAID-1 dual copy in the same controller, or remote copy in two controllers. Use this solution mainly for your most critical data, such as logs.

7.2.5 Bad data or bad channel program

The following message can indicate bad data or bad channel program:

```
IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code
```

- 140 (Open): The catalog indicates that this data set has an incorrect physical record size.
- 16 (I/O): Record not found.
- 88 (EOV): A previous extend error occurred during EOV processing of the data set.

What happened?

This problem can be pervasive, but in general, it is usually caused by two major reasons:

- Duplicate VVRs
- A bad channel, causing data overlays and corrupting indexes

The existence of damaged or duplicate VVRs on a volume can cause data sets to be overlaid with data from other data sets.

A VSAM volume record (VVR) is a logical record within VVDS. VVDS is a data set that describes the dynamic characteristics of VSAM and system-managed data sets on a DASD volume. Together with the BCS, it is a part of an integrated catalog facility.

There are many things that can cause the channel program to be bad. The most common cause is that the data that describes the data set is bad. If a bad VVR is picked up at Open time, VSAM might try to access cylinders and tracks that do not belong to the data set, generating various I/O errors.

If another program overlays the VSAM data set, the channel program might fail at the spot where the other data exists. For instance, if the CI size of the VSAM file that is broken is 4 KB, the channel program is built to read records of that size. If another program has overlaid the file with records of, say, 16-KB size, the channel programs for the record size of 4 KB fails on all cylinder/tracks/heads that do not have this record size. This situation is usually referred to as bad data.

Theoretically, system code problems can cause VSAM to build channel programs incorrectly. In some cases, they might be built correctly, but are modified incorrectly and redriven by ERP or even third-party products. Luckily, these reasons are not too common, even with new device types.

For more information about corrupted indexes, see 7.2.6, “Structural damage” on page 354.

With these problems, get as much information about the data set as you can before you restore it by using these data sources:

- LISTCAT or CSI.
- DFSMSdss (PRINT command) or DITTO to print the 3390/3380 logical cylinder/track/head that the I/O error is occurring. The output can indicate whether there is any data at all on the track, or if the data that is there belongs to a different data set.
- SMF records, mainly the type 6x connected to catalog and VSAM data sets.

After the data set is recovered, the only “history” that can be EXAMINEed is the SMF records. If the problem “clears up” after the data set is closed, but without the data set being recovered, the problem might be with an internal control block being overlaid.

What to do for recovery?

This overlay is a hard failure, and the data set must be manually restored from a backup. Often, in this case, the bad data itself gives a clue as to which data set or application caused the overlay. To avoid the restore for a bad KSDS data component, try to skip past the bad data records, and recover only those records that can be properly read.

A DIAGNOSE command, even after the data set has been recovered, can check for this problem. This command can be used because only a DELETE VVR can get rid of an orphan after one occurs. Luckily, many enhancements have been introduced to Catalog and Open processes in the last few years to check for duplicate VVRs at OPEN time. Therefore, this should be less of a problem.

What to do to avoid future problems?

Most of these errors are caused by improper sharing. For more information about sharing your VSAM data set, see 7.2.7, “Improper sharing” on page 355.

Because such types of errors can also be caused by system errors, investigate the possibility of APARs and PTFs related to the problem.

7.2.6 Structural damage

The following message might indicate structural damage:

```
IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code
```

- 128 (Close): Index search horizontal chain pointer loop encountered.
- 190 (Open): An incorrect high-allocated RBA was found in the catalog entry for this data set. The catalog entry is bad and must be restored.
- 76 (Open): Attention message: The interrupt recognition flag (IRF) was detected for a data set that was opened for input processing. This message indicates that DELETE processing was interrupted.
- 4 (I/O): End of data set encountered (during 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 and it returned to VSAM and the processing program issued another GET.
- 32 (I/O): An RBA is specified that does not give the address of any data record in the data set.
- 128 (I/O)): A loop exists in the index horizontal pointer chain during index search processing.
- 144 (I/O): An incorrect pointer (no associated base record) in an alternate index.
- 156 (I/O): An addressed GET UPD request that failed because the control interval flag was on, or an incorrect control interval was detected during keyed processing. In the latter case, the control interval is incorrect for one of the following reasons:
  - A key is not greater than the previous key.
  - A key is not in the current control interval.
  - A spanned record RDF is present.
  - A free space pointer is incorrect.
  - The number of records does not match a group RDF record count.
  - A record definition field is incorrect.
  - An index CI format is incorrect (logical I/O error).

What happened?

KSDS or VRRDS VSAM data set organizations can “break” in more ways than other data sets. This vulnerability is because they have an index component with logical pointers to other data and index CIs. If these pointers become corrupted, data can be lost or duplicated.

Also, structural information about such data sets is in the ICF catalog. For example, two RBA fields in the VVR are important in accessing a KSDS data set:

- If the RBA of the high-level index CI is corrupted, you cannot perform direct requests against the data set.
- If the RBA of the sequence set index CI is corrupted, you cannot perform sequential access.

For more information about RBA data in the VVDS catalog, see 7.6, “SMF record types that are related to VSAM data sets” on page 372.

Then, if these fields are corrupted, errors can prevent you from accessing the data even though the data is intact. Also, it is possible that the index CI’s horizontal chain is damaged, inhibiting the access to the data. These fields are often corrupted by overlays of the AMDSB control block while the data set was open, which then get updated back to the VVDS at close time. Another way is through improper sharing of the data set during initial load mode processing.

Because these fields are stored in the ″Statistics Block″ (AMDSB), jobs that only opened the data set for input still update this information at close time. For this reason, do not dismiss any possibilities just because the job was not updating the file.

SMF records are helpful when you are diagnosing VVR damage. Investigate SMF records such as type 60, 62 and 64 from all systems, to identify improper access of the data set and the time frame of the corruption.

What to do for recovery?

When dealing with KSDS/VRRDS, it is crucial that EXAMINE is run on the data set as part of the diagnosis. Also, the sooner the EXAMINE is run after the data set is broken, the better. Some types of damage can actually cause more breakage until the data set is so badly broken it is impossible to tell what actually happened first. The EXAMINE command provides details about the nature of data set damage.

Sometimes, the IDCAMS DIAGNOSE command can be used to check the data set for structural error in the catalog itself.

When you lose the index in a KSDS/VRRDS, read the data in physical sequential mode by using its data component. Use an assembler program (MACRF=ADR in ACB and OPTCD=ADR in RPL), or use IDCAMS Repro. Then, classify by the key and use an IDCAMS Define and REPRO to re-create the KSDS. For more information, see 7.4.4, “Broken Index scenario” on page 366.

What to do to avoid future problems?

Some of these errors are caused by improper sharing. If you are sharing your VSAM data set, see 7.2.7, “Improper sharing” on page 355.

Because such types of errors can be caused by system errors, investigate the possibility of APARs and PTFs related to the problem.

7.2.7 Improper sharing

The following messages might indicate improper sharing:

- `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code`
  - 16 (I/O): Record not found.
  - 20 (I/O): The record is already held in exclusive control by another requester.
  - 28 (I/O): Data set cannot be extended because VSAM cannot allocate more DASD space.
  - 88 (Open): A previous extend error occurred during EOV processing of the data set.
  - 96 (Open): Attention message: An unusable data set was opened for input.
  - 116 (Open): Attention message: The data set was not properly closed or was not opened. If the data set was not properly closed, data might be lost if processing continues.
  - 236 (I/O): Validity check error for SHAREOPTIONS 3 or 4.
- `IDC11705I INDEX RECORD CONTAINS DUPLICATE INDEX POINTERS pointer-value`

What happened?

Improper sharing is one of the most common causes of broken data sets. This section covers some of the things to check for to ensure that the share options are correct. For more information about share options, see “VSAM SHAREOPTIONS” on page 60. The following are some causes of sharing problems:

- Sharing a data set across regions (cross-region) or across systems (cross-system) without using the correct enqueuing procedures to protect data set integrity. For share options shr(3 3) shr(4 3), and shr(3 4), see related information in OY36328.
- Sharing a data set across systems, even using the appropriated share option, but without propagating the ENQ name SYSVSAM around the GRS ring. This is the most common user error. It results in duplicate index pointers in the high-level index records. See message IDC11705I.
- When a data set is defined using the model parameter, and the original data set has SPEED as an attribute of the index, data set damage can occur. VSAM does not support speed as an attribute for the index (speed is only supported for the data).
- Another area that is related to broken data sets that is specific to CBUF processing is the VSI control block. Every time a data set is opened on a system for CBUF processing, a VSI is built for the data set and added to the VSI chain. This control block is then updated by the user to communicate information from one region to another. If the user does this improperly, a broken data set can result. For more information, see 2.2.8, “Protecting VSAM data set through DISP parameter” on page 63.

Documentation is of paramount importance to address sharing problems. Obtain the necessary documents for each system (or address space) accessing the troubled data set. The major document that is needed is the IDCAMS LISTC or CSI report. List the catalog entry for the affected data set to show the allocation and RBA data (beware OY61232).

The SMF 62 and 64 records can help you determine whether the users have the data set open for output from different applications at the same time. A common user error is running applications or ISV products that were not intended to be run from multiple systems on multiple systems. These products have no logic to serialize updates.

What to do for recovery?

Use the Access Method Services VERIFY command to attempt to close the data set properly. In a cross-system shared DASD environment, the ACBERFLG = 116 might mean that the data set was not properly closed. The warning `data set not properly closed` might indicate one of the following errors in a VSAM data set:

- The “end of the data set” indicators (data set high-used RBA/CI) might be invalid.
- There might be missing records.
- There might be duplicate records.
- The data set statistics fields in the catalog might be invalid.

If VSAM OPEN cannot successfully do a VERIFY, a user cannot do a VERIFY either.

To do a recovery, perform these steps:

1. Run the IDCAMS EXAMINE command on the data set. Completion of EXAMINE without error proves that damage did not occur in a previous job. If the data set still shows damage, it must be restored before further use.
2. Proceed with the application job execution.
3. Run IDCAMS EXAMINE on the data set when the job completes.
4. If damage to the cluster has occurred, run EXAMINE on SMF records from all systems that can access the DASD volume. If shared access to the data set has occurred, correct or eliminate the contention for the data set.

What to do to avoid future problems?

Issue the VERIFY command every time you open a VSAM cluster that is shared across systems or address spaces. For more information, see 2.2, "Sharing VSAM data sets" on page 50. Use all of the serializing techniques to avoid the structural damage and the data integrity of your data set.

7.2.8 Mismatch between catalog and VTOC

This book does not cover much catalog recovery. However, some of the catalog problems are mentioned, including the ones that are associated with the IDC3009I message. For more information, see 7.6, "SMF record types that are related to VSAM data sets" on page 372. That section includes a more detailed description of the return and reason codes from this message.

- `IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code`
  - 132 (Open): One of the following errors occurred:
    - Unable to read JFCB or Scheduler Work Block.
    - Unable to connect to redo log during DFSMStvs open.
    - The forward recovery log associated with the data set was altered.
    - DFSMStvs failed to write a record to the forward recovery log.
    - The caller TASK was canceled.
    - The data set is in a forward recovery required state.
- `IDC3009I VSAM CATALOG RETURN CODE IS return-code – REASON CODE IS IGG0CLaa – reason-code`

7.2.9 VSAM does not produce expected output

Incorrect output failures can be identified by the following results:

- 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. It 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, and other system responses. These sources might indicate the source of the failure. They can help you isolate or resolve your problem, and the IBM Support Center will request them if trap or trace information is needed.

Gather the following information about the problem:

- When was the problem first noticed?
- How was the problem identified (good output versus bad output)?
- Were any system changes or maintenance recently applied, such as a new device, software product, APAR, or PTF?
- Does the problem occur with a specific data set, device, time of day, and so on?
- Does the problem occur in batch or TSO mode?
- Is the problem solid or intermittent?
- Can the problem be re-created?

- EXAMINE the system and console logs for failure-related abends, messages, and 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.
- Determine whether failure-related record management return codes and reason codes exist.

  VSAM 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 VSAM record management error indications that are returned to the caller of record management. Reason codes that are returned to the caller of record management in the RPL indicate VSAM record management errors.
- 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 *DFSMS Access Method Services for Catalogs*, SC26-7394, and the chapter on checking a VSAM key-sequenced data set cluster for structural errors in *DFSMS Using Data Sets*, SC26-7410. 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 use by the IBM Support Center. Be prepared to describe the type of data set damage. Attempt to recover the data set and rerun the failing job to determine whether the problem is resolved.

7.2.10 VSAM RLS problems

For information about VASAM ERLS problems, see 5.5, “RLS problem determination and recovery” on page 238

7.2.11 Enqueue issues

OPEN message IEC161I 052-084 is a common informational message. Most often it means that another job already has the data set open when this job is trying to open it. This is a scheduling problem rather than a system problem. This section provides information about how to determine what jobs are in contention for the same data set.

VSAM OPEN processing determines this condition by checking the GRS data. Depending on the shareoptions (SHR) of the data set and the attributes of the OPEN, VSAM enqueues against major name SYSVSAM with a minor name of the data set name/catalog name plus other information. Thus, GRS is the mechanism OPEN processing uses to ensure serialization of the OPEN process itself and the shareoptions of the file.

When VSAM issues an ENQUEUE for a SYSVSAM resource, VSAM adds an ENQRNIND indicator to the wanted resource name:

- ENQRNIND = B = BUSY
- ENQRNIND = I = INPUT
- ENQRNIND = N = Non-RLS Open
- ENQRNIND = O = OUTPUT
- ENQRNIND = R = RESERVE
- ENQRNIND = S = Sphere
- ENQRNIND = C = CLOSE

- ENQRNIND = R = RLS Read
- ENQRNIND = W = RLS Write

The ENQ for BUSY is held throughout the period of the initial operation being performed, which can be OPEN, EOV, CLOSE, TCLOSE, or CHECKPOINT RESTART. When the operation is complete or is failed, the 'B' resource is dequeued. These resources include the following items:

- ENQ DATASET/CAT/B (OPEN) EXCLUSIVE
- ENQ DATASET/CAT/O
- DEQ DATASET/CAT/B (process data)
- ENQ DATASET/CAT/B (CLOSE) EXCLUSIVE
- DEQ DATASET/CAT/O
- DEQ DATASET/CAT/B

The first place to look to determine which job is enqueued on the data set is GRS data. You can view this data with the DISPLAY GRS command or through a monitor program. Look at both the global and local queues under the major name of SYSVSAM for the data set that is getting the OPEN failure. Unfortunately, the GRS data is transient and might change before you can find the job that was responsible for the message. However, try the GRS command. Issue this GRS command from all systems that can share the DASD:

D GRS,RES=(SYSVSAM,*)

Remember: If you are using IPCS to view GRSDATA, remember that SYSVSAM will be there *twice*: Once for Local System Resources and a second time for Global Systems Resources.

The next place to look is the console log. You might be able to determine whether there were any backups, copies, or dumps being taken at the time. You might even determine whether there were unexpected batch jobs that were running at the time.

Another place to look for information is the SMF type 62 (SMF62) and SMF64 records. These records show what jobs were accessing the data set at the time of the error. However, they might not be conclusive because some access of the data set does not produce SMF records such as Media Manager, DB2, and DFSMS/dss.

Ensure the availability of the resource with JCL DD statements. Check your JCL and ensure that you have requested the correct disposition, which is DISP=SHR.

Use the AMS command ALTER to reset the Update Inhibit indicator in the data set's catalog record. Then, rerun the job by running LISTC against the failed cluster, and checking for an attribute of INH-UPDATE. If found ON, this data set is in READONLY mode and will fail an OPEN for UPDATE request. Use the AMS command ALTER UNINHIBIT to place the file in READ/WRITE mode, then rerun the job. Example 7-1 shows a sample job for Cluster DFP1.JIMONE.KSDS (on a D/T3390 with volser=339000).

Example 7-1 Sample AMS JCL

```
//SYSIN DD *
      PRINT  INDATASET(SYS1.VVDS.V339000) DUMP
      ALTER  DFP1.JIMONE.KSDS.DATA  UNINHIBIT
      ALTER  DFP1.JIMONE.KSDS.INDEX UNINHIBIT
      VERIFY data set(DFP1.JIMONE.KSDS)
      LISTC  ENT(DFP1.JIMONE.KSDS) ALL
 /*
```

In a VSAM catalog, the INHIBIT UPDATE bit is in the BCS DINC record. In an ICF catalog, INHIBIT UPDATE bit is in the VVDS VVR record.

If you are not able to determine what job is the cause of the IEC161I message, open a problem record with the Support Center. Include your RMID of IDA0192B. You will be given a SLIP to capture a memory dump that includes the GRSQ data) at the time the message is issued.

The SLIP is set in CSECT IDA0192B of load module IDA0192A upon entry to a procedure named PROBDT2B. IBM needs to know the PTF level of IDA0192B to resolve the offset of xxx in the SLIP. The user needs an AMBLIST of IDA0192A. A sample of the slip is shown in Example 7-2.

Example 7-2 SLIP to determine the job causing IEC16II

```
SLIP SET,IF,A=SYNCSVCD,L=(IDA0192A,xxx),DATA=(13R?+F8,EQ,34),
     SDATA=(ALLNUC,CSA,GRSQ,LPA,LSQA,RGN,SQA,SUM,TRT),END
```

IDA0192B is an offset into load module IDA0192A, and PROBDT2B is an offset into IDA0192B. xxx is the sum of these two offsets. The value of xxx and values that are specified in the SLIP DATA parms can change with different releases and maintenance levels of CSECT IDA0192B. Be sure to check with IBM Support Personnel to ensure that you get the correct SLIP for your level.

7.2.12 Migration issues

See the relevant migration manuals to be aware of changes you must make when you change to a new release of z/OS.

7.2.13 Performance considerations

A performance problem can appear as a lack of availability. For more information about several aspects of VSAM performance, see Chapter 4, “VSAM performance” on page 127. For more information specifically about performance monitors, see 4.6, “Performance monitors” on page 206.

7.2.14 Deadlocks

Deadlocks are a performance and also a problem determination subject.

This section defines a deadlock, and addresses how to prevent them, how to detect a deadlock, and what to do when you have one.

What is a deadlock?

Locks are used in VSAM by the DFSMS lock manager when the same control block structures are shared by strings in task programs. In such a case, Share Options do not apply. For more information, see 2.2.5, “Sharing data in a single VSAM control block structure” on page 58. If the VSAM data set is processed in non-RLS mode, there is one lock per CI. When it is processed in RLS mode, there is a lock for each logical record.

Contention for VSAM CIs locks (or logical records locks) can lead to deadlocks. For example, A might delay B in one lock, and B delays A in the other lock.

How to prevent a deadlock

Generally, avoid locking more than one logical records concurrently. Also, avoid consistent reads. If the RLS exploiter cannot follow such rules, it can create a hierarchy of locks. Then, when more than one lock is required concurrently, they are used in a pre-determined sequence.

How to detect a deadlock

The DFSMS lock manager provides a deadlock detection routine. The frequency with which it runs is determined by the installation. This routine considers a string task program to be in a deadlock situation if it is waiting for a lock for more than an installation-specified amount of time. There are two deadlock detection routines:

- Deadlocks within a system
- Deadlocks between systems

The frequencies of the deadlock detection routines are specified in GDSMSxx parameter of SYS1.PARMLIB.

In a Sysplex, the first system that is initialized with an IGDSMSxx member that has a valid DEADLOCK_DETECTION specification determines this keyword for the other systems in the Sysplex. You can change this value through the SETSMS or SET SMS commands.

This keyword specifies the intervals for running local and global deadlock detection routine.

The first subparameter, nnnn, is the local deadlock detection cycle and specifies the interval in seconds for detecting deadlocks within a system.

The second subparameter is the global deadlock detection cycle, and specifies the interval for detecting deadlocks between systems. This value is specified as the number of local detection cycles that occur before global deadlock detection is initiated.

To determine whether a string task program is in a dead lock situation, DFSMS lock manager compares the RLSTMOUT keyword with the amount of time a string task program has waited for a lock. The RLSRMOUT command has the following format:

```
RLSTMOUT({nnnn|0})
```

The command specifies the maximum time, in seconds, that a VSAM RLS request must wait for a required lock before the request is assumed to be in deadlock. You can specify a value between 0 to 9999 (in seconds). A value of 0 means that the VSAM RLS request has no timeout value. In this case, the request waits indefinitely to obtain the required lock.

RLSTMOUT can be specified only once in a sysplex and applies across all systems in the sysplex.

What to do when a deadlock is detected

When a lock request is found in a deadlock, VSAM rejects the request that is waiting. This results in the VSAM request completing with a deadlock error response. Your applications must be prepared to accept locking error return codes that might be returned on GET or POINT NRI requests. However, normally such errors do not occur.

7.2.15 VSAM restriction: REPRO to empty VSAM data set

Keep in mind this important restriction when using the REPRO command. If you try to repro into an empty VSAM data set, you get the error IDC3351I ** VSAM OPEN RETURN CODE =160. This is a permanent restriction of VSAM. One work-around is to “prime” the data set by adding and deleting one record. This process increments the HI-U-RBA to a value other than zero, and allows the REPRO to proceed.

7.3 What documentation to collect

Familiarize yourself with getting the required documentation, such as logs, memory dumps, traces, and messages that are associated with errors. There is an information APAR II12927 in IBMLINK that provides the information you need to collect data for general VSAM problems.

Tip: Generally, include an AMBLIST of the VSAM load modules IDA019L1 and IDA0192A when you are sending a VSAM problem to IBM level2 support.

7.3.1 IDCAMS problems

The following are documentation requirements for IDCAMS problems:

- Syslog (including all messages, JCL, and commands)
- VERIFY output
- LISTCAT ALL output
- EXAMINE (ITEST and DTEST)
- DIAGNOSE

7.3.2 Broken VSAM data set

Collect the following documentation for broken VSAM data sets:

- SYSLOG/JOBLOG (including all messages, JCL, and commands)
- EXAMINE (both ITEST and DTEST) output
- LISTCAT ALL output
- DIAGNOSE output

For more information, see informational APAR II08859.

7.3.3 Obtaining a VSAM record management trace

Sometimes you might need to collect VSAM record management traces. This section describes how to collect VSAM record management traces. To do so, perform these steps:

1. Activate GTF before you allocate the data set that is being traced. Start GTF with the TRACE=USRP,USR=FF5, END command.
2. Set the SYS1.TRACE data set to approximately 100 cylinders, or larger if the space is available. If, however, the activity on this file will be heavy, there is a possibility of “lost trace entries.” This means that the number of trace records that are coming in is more than GTF can keep up with. Splitting the SYS1.TRACE data set into multiple data sets usually prevents this contention. For more information, see the informational APAR II10072. An example is shown in Example 7-3 on page 363.
3. Place an AMP=('TRACE=(subparameters)') on the DD statement of the data set that you want to trace. VSAM Level 2 provides you with the coding of the AMP parameter.

Requirement: If the data set being traced is dynamically allocated, a usermode must be provided from L2 and applied to your system. For example:

```
//ddname DD ...,
//   AMP=('TRACE=(PARM1=64C4A1802000,HOOK=(0,1,3,9,10,11,25,26))')
```

4. Restart the applicable CICS regions, and run batch jobs. GTF must be started BEFORE any data set allocation occurs. Otherwise, the control blocks for tracing are not built and no tracing can occur.
5. Re-create the problem.
6. Stop the trace as soon as the job terminates. An example of JCL that can be used to send the GTF output to more than one data set is shown in Example 7-3. This configuration is often necessary on high output traces:

Example 7-3 Sampler JCL for GTF trace

```
//GTFABC PROC MEMBER=GTFPARM
//IEFPROC EXEC PGM=AHLGTF,REGION=2880K,TIME=1440,
// PARM=('MODE=EXT,DEBUG=NO,TIME=NO')
//IEFRDER DD DSNAME=SYS1.GTFTRC,UNIT=SYSDA,
// SPACE=(4096,20),DISP=(NEW,KEEP)
//SYSLIB DD DSN=SYS1.PARMLIB(&MEMBER),DISP=SHR
//GTFOUT1 DD DSNAME=SYS1.TRACE1,UNIT=SYSDA,DISP=(NEW,KEEP)
//GTFOUT2 DD DSNAME=SYS1.TRACE2,UNIT=SYSDA,DISP=(NEW,KEEP)
//GTFOUT3 DD DSNAME=SYS1.TRACE3,UNIT=SYSDA,DISP=(NEW,KEEP)
```

GTF can be started with the command S GTFABC.

SYS1.PARMLIB(GTFPARM) is the parmlib member that contains the trace parms to be used:

```
TRACE=USRP,USR=(FF5),END.
```

7.4 Recovering a damaged VSAM data set

IDCAMS has three important commands that are used to recover VSAM clusters and catalogs. They are EXAMINE, DIAGNOSE, and VERIFY.

7.4.1 EXAMINE command

EXAMINE is an IDCAMS command that allows the user to analyze and collect information about the structural consistency of KSDS data set clusters and of a VVRDS data set cluster. In addition, EXAMINE can analyze and report on the structural integrity of the BCS of an ICF catalog. This function cannot share a cluster that is opened for output or update by another task.

The EXAMINE function runs two possible tests:

- Test the Index portion (INDEXTEST), the default: Evaluates the full index component of the KSDS/VRRDS cluster by cross-checking vertical and horizontal pointers that are contained within the index control intervals. It also performs analysis of the index information. This test usually requires medium to few resources.

- Test the Data portion (DATATEST): Evaluates the index sequence set and data component of the key-sequenced data set cluster by sequentially reading all data control intervals, including free space control intervals. Tests are then carried out to ensure the following items:
  - Record and control interval integrity
  - Free space conditions
  - Spanned record update capacity
  - The integrity of various internal VSAM pointers that are contained within the control interval.

  Usually this test requires considerable resources.

You can also limit the number of error messages generated (ERRORLIMIT) by EXAMINE. For more information about what to do when EXAMINE is reporting errors in the Index or Data portion, see 7.3.2, “Broken VSAM data set” on page 362.

7.4.2 DIAGNOSE command

To analyze a catalog for synchronization errors, you can use the IDCAMS DIAGNOSE command. With this command, you can analyze the content of catalog records in the BCS and VVDS, and compare VVDS information with DSCB information in the VTOC. Besides checking for synchronization errors, DIAGNOSE also checks for invalid data and invalid relationships between entries.

The DIAGNOSE command checks the content of the catalog records. Because these records might, for example, contain damaged length field values, the DIAGNOSE job might abend. For more information about using DIAGNOSE, see z/OS *DFSMS Managing Catalogs*, SC26-7409.

7.4.3 VERIFY command

Some clarification of the word VERIFY is necessary. VERIFY is a record management macro (just like GET or PUT). It can be used with certain types of opened VSAM data sets to ensure that various fields in the VSAM control blocks in catalog are accurate. It checks the ICF catalog against the VSAM clusters.

After receiving this macro, record management starts reading the data set CI by CI, starting with the current high used RBA value stored in a memory control block. If the data set is a KSDS, then both the index and the data are VERIFYed.

VERIFY is also an IDCAMS command. In this case, IDCAMS opens the requested data set for output, issues the record management VERIFY macro, and then closes the data set. When the data set is closed, VSAM Close processing uses its catalog interface to call Catalog to update the VVR information from the new information in the VSAM control blocks. It is important to understand that IDCAMS is not updating VSAM control blocks or the catalog directly.

Clusters, alternate indexes, entry-sequenced data sets, and catalogs can be verified. Paths over an alternate index and linear data sets cannot be verified, and the same is true of data sets in RLS mode. Paths that are defined directly over a base cluster can be verified.

When a data set is closed, its end-of-data and end-of-key-range information is used to update the data sets cataloged information in the VVR AMDSB cell. This information includes the following fields:

- High used RBA/CI for the data set
- High key RBA/CI
- Number of index levels
- RBA and the CI number of the first sequence set record
- System time stamp

For more information, see 7.6, “SMF record types that are related to VSAM data sets” on page 372.

A successful VERIFY implies these characteristics:

- The “end of the data set” indicators (data set high-used RBA/CI) are probably valid.
- There might be missing records.
- There might be duplicate records.
- The data set statistics fields in the catalog might be invalid.

The recovery actions are the same for a VSAM data set that was not properly closed, whether VERIFY runs successfully or not.

Implicit VERIFY versus explicit VERIFY

Another area that is confusing when discussing VERIFY involves the terms *explicit VERIFY* and *implicit VERIFY*. An explicit VERIFY refers to the user starting the VERIFY by issuing an IDCAMS VERIFY job against the data set. An implicit VERIFY is when VSAM Open processing internally starts the VERIFY macro against the data set when it determines that the data set was not previously closed properly.

An improper close means that a previous job that had the data set open for output either abended or failed to close the data set for some reason. Open processing uses a bit in the catalog to determine whether a data set was not properly closed. When a job opens a data set for output processing, this bit (the “open for output” bit) is turned on. It is not turned off until the job closes the data set normally.

Open processing, if it finds this bit on in a subsequent open, uses GRS (enqueuing against the data set name) to determine whether the data set is open for output. If not, then it concludes that the last close was abnormal and issues the VERIFY before it completes the Open process. It also issues IEC161I messages (rc56 and rc62). All Open jobs against the data set get this implicit VERIFY and the associated messages until the data set is opened for Output.

VSAM OPEN will NOT do an implicit VERIFY for the following OPENs:

- The data set is being opened for input.
- The data set is being opened for RESET processing.
- The user requested that the verify be suppressed.
- The data set is being opened for improved control interval processing.
- The data set is being opened for RESTART processing.
- The data set is being opened for non-ESDS create mode processing.
- The data set is a Linear data set.

VSAM OPEN does an implicit VERIFY without issuing a message when the ACB being opened is the first open for output connecting to an existing control block structure with shareoptions=2,x.

Implicit VERIFY problems

The following problems with the implicit VERIFY are not inherently obvious. Normally, a VERIFY does not take many processor resources because it is reading in CI mode from the HURBA in the catalog to the “real” HURBA. However, if the data set had much activity in the output job before it abended (for example, a log file that was being loaded), the VERIFY can take several minutes. In this situation, it must basically read the entire file. This processor usage can have a severe effect if there were many files open in the application.

CICS provides a good example of this problem. If the CICS region comes down hard (for example, if a CEMT P SHUT IMM is issued), all TCBs are abended. In this scenario, all files open for output at the time have the “open for output” bit on in the VVR. They must all be implicitly VERIFY'd as the region is brought back up. If so, have a procedure in place to explicitly VERIFY the important files in the case of an abend situation. This procedure allows the system to come up and the important applications to start running immediately.

ACTION=REFRESH

The ACTION=REFRESH parameter was added to the VERIFY macro quite some time ago. Without this parameter, the VERIFY macro only reads up to the current HARBA of the data set (as set in VSAM control block). This value might have changed since the last time the data set was opened by the current job. Therefore, this facility allows the control blocks for a data set to be updated to reflect the current structure of the data or index by using the values in the catalog/VVR. To accomplish this update, record management through EOV uses catalog interface to update the storage control blocks with the current boundaries of the data set.

Use the cluster or alternate index name as the target of your VERIFY command. Although the data and index components of a key-sequenced cluster or alternate index can be verified, the timestamps of the two components are different following the separate VERIFYs. This discrepancy might cause further OPEN errors.

7.4.4 Broken Index scenario

A broken index scenario involves the following conditions:

- You have a program that opens a KSDS data set for update. The access argument is through RBA. However, the user prepared the JCL pointing to the index name instead of cluster name or data name.

- The program finishes with a return code equal to zero, but with an unknown damaged index. However, the index time stamp is now different from the one in the data component, as LISTCAT shows. Nevertheless, RACF does not prohibit this action because the user is the owner of the cluster. See Figure 7-1.

```
//*JCL -----------------------------------
//SEQ     EXEC PGM=TSTIDX
//VSAM     DD DSN=KSDS.K4REP.INDEX,DISP=SHR

TSTIDX PROGRAM:
ACB,DDNAME=VSAM,MACRF=(ADR,SEQ,OUT)
RPL,ACB=(R2),OPTCD=(ADR,SEQ,UPD,MVE)

SYSOUT MESSAGE:
JOBNAME  STEPNAME PROCSTEP    RC
TSTIDX            SEQ         00

LISTCAT ENTRY(‘KSDS.K4REP’) ALL:
DATA: SYSTEM-TIMESTAMP: X'B3E245958A0845C4'
INDEX: SYSTEM-TIMESTAMP: X'B3E278BEC0D91601'
```

Figure 7-1 JCL, program, message, and catalog entry

- The next time that you open the cluster or the data, the message IEC161I 058(018)-061 is issued and the OPEN return code is 4. The processing continues. See the message in Figure 7-2.

```
IEC161I 058(018)-061:

058      The time stamp for the index does not match the time stamp for
          the data set. This could occur if the data set was updated
          without the index being open.

          System Action: OPEN processing continues. The error flags
         in the ACB (access method control block) for the data set are set to 108.

          Programmer Response: You can continue to process the data
         set, but errors can occur if the data set and index do not
          correspond. Check for possible duplicate VVRs.
```

Figure 7-2 IEC161I message

- During processing, in a GET or PUT, the program receives a return code 8, reason code X’9C’, indicating that an invalid control interval was detected. The program reading the data issues a message that indicates the error. When the program is IDCAMS, the processing stops and the messages that are shown in Figure 7-3 are issued in the SYSPRINT ddname file. If the data set is open for output, the time stamp is corrected, but the I/O error remains after the index is damaged.

```
IDC3300I  ERROR OPENING KSDS.K4REP
IDC3351I ** VSAM OPEN RETURN CODE IS 108
IDC3302I  ACTION ERROR ON KSDS.K4REP
IDC3351I ** VSAM I/O RETURN CODE IS 156 - RPLFDBWD = X'D708009C'
IDC31467I MAXIMUM ERROR LIMIT REACHED.
```

Figure 7-3 IDC message in the console

- If you run an EXAMINE IDCAMS command, you get back the messages shown. The error in the test environment was caused by filling the first control interval with X'00' as shown in Figure 7-4.

```
INDEXTEST BEGINS
HIGH-LEVEL INDEX CI EXPECTED BUT NOT ACQUIRED
CURRENT INDEX LEVEL IS 3
INDEX CONTROL INTERVAL DISPLAY AT RBA/CI 245760 FOLLOWS
 000000  00000000 00000000 00000000 00000000   00000000 00000000 00000000 000000
00   *................................*
 000020  00000000 00000000 00000000 00000000   00000000 00000000 00000000 000000
...

ERROR LOCATED AT OFFSET 00000010
MAJOR ERRORS FOUND BY INDEXTEST
LASTCC=8
```

Figure 7-4 Examine messages

To recover the index, perform the following steps:

1. Use the REPRO Command to copy just the *data* component of the KSDS to a sequential data set. Specify the data component name (not the Cluster Name) in the REPRO INFILE parameter.
2. Sort the sequential data set by key.
3. DELETE and re-DEFINE the damaged cluster.
4. REPRO from the Sorted Sequential File to the newly defined cluster. Record Management rebuilds the index component.

Restriction: This method does not work for a VSAM Catalog, integrated catalog facility (ICF), or for a Spanned KSDS.

7.4.5 Abend task scenario

At task abend, RTM does not properly close the VSAM data set. This improper close causes the following problems:

- Buffers are not flushed except for cross systems that have shareoptions 4, and the HURBA is not updated in the catalog. In Language Environment, the TRAP ON option forces the close, with flush and HURBA actualization, by using a STAE exit.

  If the HURBA was not updated, when the data set is later opened and the user's program attempts to process records beyond end-of-data or end-of-key range, a read operation results in a “no record found” error. In addition, a write operation might write records over previously written records. To avoid this situation, use the VERIFY command to correct the catalog information. For more information about recovering a data set, see *DFSMS/MVS Managing Catalogs*, SC26-4914.

- If the data set was opened for output, the open-for-output indicator is left on. In this case, at next Open, VSAM implicitly issues a VERIFY command when it detects an open-for-output indicator on. It also issues a message that states whether the VERIFY command is successful. However, a successful VERIFY does not mean that the data set is error free.

  If a subsequent Open is issued for update, VSAM turns off the open-for-output indicator at successful Close. If the data set is opened for input, however, the open-for-output indicator is left on. For more information, see 7.2.3, “Mismatch between catalog and data set” on page 350.

7.4.6 Recovering damaged BCS entries

To recover damaged BCS entries, perform these steps:

1. Remove the sphere or base record, if it exists.

   The damage that is detected might not be in a sphere or base record. If it is not, the entry name of the sphere or base record is indicated in messages IDC21364I and IDC21365I.

2. Remove any remaining association records.

   You can re-execute the DIAGNOSE command after you remove the sphere or base record to identify any unwanted truename or association entries in the BCS. You can remove these entries by using the DELETE command with the TRUENAME parameter.

3. Reintroduce the removed entries into the catalog.

   After the damaged entries are removed, you can redefine the data sets. For VSAM and SMS-managed non-VSAM data sets, specify the RECATALOG option of the DEFINE command.

If you are recovering generation data group entries, use the same procedure. However, you must reintroduce the current generation data sets into the catalog in the correct order after the generation data group is redefined. You can use the LISTCAT command to determine the current generation data sets.

7.4.7 Recovering damaged VVDS entries

To recover damaged VVDS entries, perform these steps:

1. Remove the entries in the BCS for the data set, if they exist. Before the damaged VVDS records can be removed, you must remove the entries in the BCS. For more information about removing BCS entries, see 7.4.6, “Recovering damaged BCS entries” on page 369.
2. Remove the damaged VVDS records. After you remove the BCS entries, you can remove the VVDS records by using the Delete command and specifying VVR or NVR. Deleting VVR or NVR also removes the Format 1 DSCB from the VTOC.
3. Recover the data set from a backup copy.

If a backup copy of the data set does not exist and the data set can be opened, you can attempt to recover some of the data. Depending on the extent and type of damage in the VVDS record, you might be unable to recover any data. In addition, the data that you do recover might be damaged or out of sequence.

7.5 Preventing future problems

You can take a number of actions to prevent VSAM problems:

- Backing up your VSAM data sets
- Keep your system at current maintenance levels

7.5.1 Backing up your VSAM data sets

Back up all your critical data by using one or more of the following methods:

- SMS management class with ABARS for backups to allow restore of your data in the case of a hardware error and application error. This type of backup is also useful for disaster recovery.
- Remote copy for disaster recovery.

Backup has issues in the following areas that you need to consider:

- IDCAMS export and import
- Backup-while-open considerations

IDCAMS export and import

The EXPORT and IMPORT commands have the following unique characteristics:

- The EXPORT command either exports a cluster or an alternate index, or creates a backup copy of an integrated catalog facility catalog.

  Exporting involves storing the cluster or AIX data in other media in a non-processable format, together with catalog information about the data set. An empty candidate volume cannot be exported. Access Method Services acknowledge and preserve the SMS classes during EXPORT.

  Figure 7-5 is an example in which a key-sequenced cluster, ZZZ.EXAMPLE.KSDS1, is exported from a user catalog, HHHUCAT1. The cluster is copied to a portable file, TAPE2. Its catalog entries are modified to prevent the cluster's data records from being updated, added to, or erased.

```
//EXPORT1  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//RECEIVE  DD    DSNAME=TAPE2,UNIT=(TAPE,,DEFER),
//         DISP=NEW,VOL=SER=003030,
//         DCB=(BLKSIZE=6000,DEN=3),LABEL=(1,SL)
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
EXPORT -
ZZZ.EXAMPLE.KSDS1 -
OUTFILE(RECEIVE) -
TEMPORARY -
INHIBITSOURCE
/*
```

Figure 7-5 Exporting KSDS

- IMPORT is the opposite operation to EXPORT. Here, you reload the cluster or AIX data and replace its information in an active catalog.

  Figure 7-6 on page 371 is an example in which a key-sequenced cluster, BCN.EXAMPLE.KSDS1, that was previously EXPORTed, is IMPORTed. The OUTFILE and its associated DD statement are provided to allocate the data set. The original copy of BCN.EXAMPLE.KSDS1 is replaced with the imported copy, TAPE2. Access Method Services finds and deletes the duplicate name, BCN.EXAMPLE.KSDS1, in the catalog VCBUCAT1.

  A duplicate name exists because TEMPORARY was specified when the cluster was exported. Access Method Services then redefines BCN.EXAMPLE.KSDS1 by using the catalog information from the portable file TAPE2.

```
//IMPORT2  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SOURCE   DD    DSNAME=TAPE2,UNIT=(TAPE,,DEFER),
//   VOL=SER=003030,DISP=OLD,
//   DCB=(BLKSIZE=6000,LRECL=479,DEN=3),LABEL=(1,SL)
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
IMPORT -
INFILE(SOURCE) -
OUTDATASET(BCN.EXAMPLE.KSDS1) -
CATALOG(VCBUCAT1)
/*
```

Figure 7-6 IMPORT of KSDS cluster

Backup-while-open considerations

Backup-while-open (BWO) allows DFSMSdss logical dump for an IMS or a CICS/VSAM data set while open-for-update. BWO works with Concurrent Copy in 9390, Snapshot in RVAs, or FlashCopy in ESS. The VSAM data set must be SMS-managed.

The DFSMSdss BWO function does not apply to catalogs, VVDSs, LDSs, physical dump, and restore.

When you define the cluster with IDCAMS, you must declare it as a BWO. You can select the following options:

- TYPECICS

  Use the TYPECICS parameter to specify BWO in a CICS environment. For RLS processing, this option activates BWO processing for CICS. For non-RLS processing, CICS determines whether to use this specification or the specification in the CICS control data named FCT.

  Remember: If CICS determines that it uses the specification in the CICS FCT, the specification might override the TYPECICS or NO parameters.

- TYPEIMS

  If you want to use BWO processing in an IMS environment, use the TYPEIMS parameter.

- NO

  Use this parameter when BWO does not apply to the cluster.

To have BWO with total security and integrity, the following products are modified:

- DFSMSdfp, where catalog services are changed to prevent unauthorized alterations of BWO indicators. DFSMSdfp does not allow deletion of a data set that DFSMSdss dumps as a BWO data set.
- DFSMSdss enqueue serialization has been changed to prevent data integrity exposures when you perform defrags, memory dumps, or restore operations.
- DFSMShsm, used for incremental backups and aggregate backups of a data set, starts DFSMSdss to perform BWO.

In the catalog, the following fields refer to BWO:

- BWO

  Data set is enabled for backup-while-open.
- BWO STATUS

  Indicates the status of the data set:

  - Data set is enabled for backup-while-open.
  - Control interval or control area split is in progress.
  - Data set has been restored and is down level. It might need to be updated with forward recovery logs.
- BWO TIMESTAMP

  A CICS time stamp that indicates the time from which forward recovery logs must be applied to a restored copy of the data set.

7.5.2 Keep your system at current maintenance levels

Keep your system at the current maintenance level. Apply PTF selective service in your operating system, especially the ones that are associated with VSAM. To find fixes associated with broken VSAM data sets, use the search word *dsbreaker* in either IBM RETAIN® or through IBMLINK.

7.6 SMF record types that are related to VSAM data sets

The following SMF records that are related to VSAM recovery and VSAM performance:

- SMF record type 60
- SMF record type 61
- SMF record type 62
- SMF record type 63
- SMF record type 64
- SMF record type 65
- SMF record type 66
- SMF record type 67
- SMF record type 68
- SMF record type 69
- SMF record type 42

7.6.1 SMF record type 60

Record type 60 is written when a record is inserted, updated, or deleted from a VVDS. This process occurs, for example, when a VSAM cluster is defined, closed, or deleted.

VVDS is a part of ICF catalog structure (the other is BCS) that is in the volume that contains the described data sets. It contains dynamic information, such as statistics, about these data sets. VSAM data sets and SMS data sets must be cataloged in an ICF catalog. The record that is related to a VSAM data set is a VVR. The record that is related to non-VSAM data sets is a non-VSAM volume record (NVR).

One type 60 record is written for each VVR or NVR written or deleted. This record provides the following data:

- Identifies the VVDS in which the VVR or NVR is written or deleted.
- Contains the new or updated catalog record.
- Identifies the job by job log and user identifiers.

7.6.2 SMF record type 61

One type 61 record is written for each record that is inserted or updated in a catalog. This record provides the following data:

- Identifies the entry that is being defined and the catalog in which the catalog record is written.
- Contains the new or updated catalog record.
- Identifies the job by job log and user identifiers.

7.6.3 SMF record type 62

Record type 62 is written at the successful or unsuccessful opening of a VSAM component or cluster. The record provides the following data:

- Identifies the VSAM component or cluster.
- Indicates whether it was successfully opened.
- Names the VSAM catalog in which the object is defined, and the volumes on which the catalog and object are stored.
- It identifies the job that issued the OPEN macro by job log identification and user identification.

This record is not generated when a system task issues the OPEN macro.

7.6.4 SMF record type 63

Record type 63 is written when a VSAM catalog entry is defined by the DEFINE Access Method Services command, and when that definition is altered. The catalog entry can be a component, cluster, catalog, alternate index, path, or non-VSAM data set. An example is when a VSAM catalog entry is altered with new space allocation information. That is, when the VSAM End-Of-Volume (EOV) routine extends the entries object. Another example is if the entry is changed by the Alter Access Method Services command. One record type 63 is written for each newly created or altered entry. This record is not written when a VSAM catalog is renamed. In that case record type 68 is written. This record provides the following data:

- Identifies the catalog in which the object is defined.
- Lists the catalog record for the newly defined object, and, for an alteration, lists the parts of the old catalog record before they were altered.
- Identifies the job and the user that caused the record to be written. If it was caused by a system task, the job-name and the user-identification fields contain blanks and the time and date fields contain zeros.

7.6.5 SMF record type 64

Record type 64 is written in these circumstances:

- A VSAM component or cluster is closed.
- VSAM must switch to another volume to continue to read or write.
- The component ran out of space and EOV is called to extend the component.

When a cluster is closed, one record is written for each component in the SMF record type 64. The reason why the record was created is indicated in the record.

SMF record type 64 description

The record describes the device and volumes on which the object is stored, and gives the extents of the object on the volumes. It provides statistics about various processing events that have occurred since the object was defined. These events can include the number of records in the data component, the number of records that were inserted, and the number of control intervals that were split.

The record that is written when the VSAM component or cluster is closed contains changes in statistics from OPEN to the time of EOV and CLOSE.

SMF64 sample program

The IDCAMS LISTCAT command shows the cumulative number of EXCPs since the initial load. Sometimes you need to know the number of EXCPs run between OPEN and CLOSE, mainly when you are doing tuning in buffering. For more information, including sample source assembler code, see "Sample programs extract from SMF record type 64" on page 384. It can be used for showing more information about your VSAM data sets. By using JCL parameters, you can determine an EXCPs threshold. The program then shows only data that covers the data sets with equal or more EXCPS than the threshold that occurred between open and close. The report that is generated is based on the SMF 64 record, and is generated each time a VSAM data set is closed. It can help you to determine the characteristics of the data sets with high I/O activity.

To reduce the I/O activity, use these techniques:

- You can use SMB if the data sets are already in extended format.
- You can convert data sets to extended format, and then use SMB.
- With LSR buffering and direct access, you can discover whether the data sets use defer write, and if not, whether they can.
- You can determine whether the VSAM buffers are below 16 MB, and whether to move them above 16 MB.
- For KSDS and VRRDS data sets, when the total number of free control intervals is much higher than free space defined to the data set, consider reorganization. Do not reorganize data sets if it is not necessary.
- When you are making changes in buffering, use the report to see how the numbers of EXCPs decreased.

For more information about SMF type 64 fields, see *OS/390 MVS System Management Facilities (SMF)*, GC28-1783.

7.6.6 SMF record type 65

Record type 65 is written during any processing that results in a DELETE request to catalog management services, which includes these commands:

- IDCAMS DELETE
- IEHPROGM UNCATLG

One type 65 record is written for each record that is updated or deleted from a catalog. The record provides the following data:

- Identifies the entry that is being deleted.
- Identifies the catalog in which the catalog record is updated or deleted.
- Identifies the updated or deleted catalog record.
- Indicates whether a VSAM cluster or non-VSAM data set was scratched, or whether only catalog information was deleted.
- Identifies the job and the user. If a system task caused the record to be written, the job name and user identification fields contain blanks, and the time and date fields contain zeros.

7.6.7 SMF record type 66

Record type 66 is written during any processing that results in an ALTER request to Catalog Management Services, such as IDCAMS ALTER.

One type 66 record is written for each record that is written or deleted from a catalog. The record provides the following data:

- Identifies the entry that is being altered.
- Identifies the catalog in which the catalog record is written or deleted.
- Identifies the new, updated, or deleted catalog record.
- Indicates whether the entry was renamed and, if so, gives the old and new names of the entry.
- Identifies the job and the user. If a system task caused the record to be written, the job name and user identification fields contain blanks and the time and date fields contain zeros.

7.6.8 SMF record type 67

Record type 67 is written when a VSAM catalog entry (a component, cluster, catalog, alternate index, path, or non-VSAM data set) is deleted. A record is written for each entry that is affected by the DELETE Access Method Services command. The record provides the following data:

- Identifies the deleted entry.
- Identifies the VSAM catalog in which the entry was defined.
- Gives the total logical VSAM catalog record.
- Identifies the job and user that caused the record to be written. If it was caused by a system task, the job-name and the user-identification fields contain blanks and the time and date fields contain zeros.

7.6.9 SMF record type 68

Record type 68 is written when a VSAM catalog entry is renamed by using the ALTER Access Method Services command. The entry can be a component, cluster, catalog, alternate index, path, or non-VSAM data set. This record provides the following data:

- Identifies the VSAM catalog in which the object is defined.
- Provides the old and new names for the object.
- Identifies the job and user that renamed the data set. If a system task caused the record to be written, the job-name and user-identification fields contain blanks and the time and date fields contain zeros.

7.6.10 SMF record type 69

Record type 69 is written when a VSAM data space is defined, extended, or deleted by using the DEFINE or DELETE Access Method Services commands. Record type 69 is not written when a catalog or a unique data set is defined or deleted. This record provides the following data:

- Identifies the catalog in which the data space is defined.
- Identifies the volume on which it is (or was) allocated.
- Gives the number of free data space extents and the amount of deallocated space on the affected volume after the definition, extension, or deletion.
- Identifies the job and the user that caused the record to be written. If it is caused by a system task, the job-name and user-identification fields contain blanks and the time and date fields.

7.6.11 SMF record type 42

This record provides VSAM record level sharing (RLS) statistics, in addition to other information.

7.7 VSAM data trap

The VSAM data trap is a RAS enhancement for VSAM record management. VSAM Record Management checks all data component CIs before they are written to DASD. If they are in error, they are not written. Data CIs are checked for integrity. If there is any corruption, an SVC dump is captured and a message is issued to the console to notify the user that a problem was encountered with the KSDS. Data component CIs are checked for records out of sequence, records in the freespace area, and any problems with RDF or CIDF fields.

This enhancement has been provided to resolve the problem where a VSAM KSDS data component might fail, which will cause the VSAM index to fail. If the index trap has failed, this might result in multiple data collection outages. Usually, the data is collected too late after the initial error and is usually not sufficient to diagnose what happened to the data component. Using the data trap, VSAM can capture a data failure for certain types of data component corruption.

The VSAM Data Trap default is OFF. The trap is enabled easily by command. If performance is adversely affected, the trap is easily disabled by a command.

7.7.1 Enabling, disabling, and displaying the trap status

Data trap default is off. The following are the MVS commands that are provided to turn Data Trap on or off, or to verify its status.

```
V SMS,MONDS(IGWVSAM.BASE.DATA.TRAP),ON|OFF
D SMS,MONDS(IGWVSAM.BASE.VSAM.DEBUG.FEATURES)
```

7.7.2 VSAM KSDS data trap features

The VSAM KSDS data trap includes the following features and limitations:

- The trap is only for KSDS.
- The trap does not test spanned records or when there can be only one record in a CI.
- The trap cannot test for sequence errors between control intervals.
- The trap tests the key sequence in the CI as it is being written to DASD.
- It might cause performance issues, especially during WRTBFR processing.
- The trap can be turned on and off by command.
- It can test for data key sequence errors between CIs (control intervals).
- It tests only data set component names that are identified by users.

A new control block, PLHDTRAP is obtained during OPEN processing in IDA0192Y for every PLH.

The data trap code is in VSAM module IDA019RV, and is called from VSAM module IDAM19R3 - the physical I/O driver.

When IDAM19R3 is entered, it runs the following checks:

- Data trap was activated by user
- Not a catalog data set
- Data component
- A keyed data set
- Data work areas were already built
- A write request
- Not spanned records before a call to IDA019RV is made

There are further checks in trap code in IDA019RV. The trap tests to see whether the I/O is for a write of a data component of a KSDS. It is bypassed if it is not a KSDS, or if it is a KSDS with spanned records. The trap checks the data CI in the buffer by analyzing the control interval definition field (CIDF) values. It checks whether there are any records in the free space between the last record and the last RDF entry. The trap obtains the AMDSB from the current AMB, and determines the key length and the key offset in the records for the data set. These values are kept in the trap control block, and are used in comparing the key sequence of the data records in the CI.

The trap locates the first record definition field (RDF) entry and searches through the RDF entries to locate the individual data records. The RDF is either a single entry for variable length records, or is an RDF pair for equal length records. It is possible to have both types of RDF entries in a CI, so the RDF is analyzed for the type of entry. For equal length records, there is a count of the equal length records. The trap must track of the count while it is stepping through the data records.

The trap might encounter a key sequence error that consists of consecutive equal keys or the keys are not in ascending sequence. In this situation, message IDAI1003E is generated to the system console and an SVC dump is taken for problem analysis. The message is similar to the INDEX TRAP message IDAI1002E, and identifies the failing data set.

When a corrupted data CI is encountered, the bad data buffer is not written to DASD. Information for the console message and the parameters for the SVC dump are contained in the trap control block. The storage address pointer to the trap control block is in the data AMB for the individual data sets.

A console message is issued when the VSAM KSDS 'data trap' hits, and the message identifies the failing data set. The message is also saved in the job output.

7.8 VSAM trace enhancement

VSAM record management tracing and VSAM internal tracing have been enhanced to improve RAS for VSAM. VSAM tracing usability is expanded by adding new hook points within various VSAM processing modules for record management. Serialization of the tracing routine is also improved. Improvements have also been made to IPCS trace formatted output. Some minor enhancement of the VSAM internal (VSAM footstep) tracing has been made to improve first failure data capture.

In the past, VSAM record management tracing was enabled by using the VSAM AMP parameters in JCL. For data sets like catalogs that have no DD JCL statement associated with them, a usermod had to be applied for a specific data set. This process required an IPL and at times was not practical.

7.8.1 New hook points

New hook points are added to VSAM record management trace to enable tracing of CI reclaim and compression processing in VSAM. The new hook points, their module location, and point in processing are detailed in Table 7-1.

Table 7-1 New hook points

Trace Point	Module	Description
20	IDAVCCMS	Before data record compression
21	IDAVCCMS	After data record compression
22	IDAVCCMS	Before data record decompression
23	IDAVCCMS	After data record decompression
26	IDA019SC	Start of a CI reclaim
27	IDA019SC	After completion of a CI reclaim

Use the HOOK subparameter of the TRACE subparameter of the JCL AMP DD parameter to select which hook points are to be used in a VSAM record management trace.

7.8.2 Control blocks traced

With this support, you can trace the following control blocks under certain circumstances:

- CMWA (compression work area) is traced when byte 1 bit 4 is on in PARM1/PARM2 of VSAM record management trace subparameter and hook points 20-23 are being traced. See Example 7-4 on page 379 showing the HOOK and PARM1 setting at the appropriate values to trace the CMWA. The necessary values are in bold font.
- The RPL ECB is traced when RPLECB points to a user ECB
- The ECB used in the IOMB is traced when the user specifies the IOMB is to be traced.

7.8.3 Starting the trace

Starting the VSAM record management trace requires GTF to be active and specification of the required hook points in the TRACE subparameter of the AMP JCL parameter. To start the trace, perform these steps:

1. Code GTF parameters in a member. You can use the USR parm to trace all GTF user records with a TRACE statement:

   TRACE=USR

   You can also use USRP for more specific information and to stop the collection of any other USR events. Use the following commands:

   TRACE=USRP

   USR=FF5

2. Start GTF. Ensure that the GTF common storage buffer is at least 1024 KB to handle the trace records. Use the BLOK parameter on the START GTF command:

   S GTF,,,(MODE=INT,BLOK=10M,SA=10M,SD=10M),MEMBER=xxxx,DSN=NULLFILE

   This command starts an internal GTF that can be formatted in an SVC dump in which up to 10 MB of trace records is available. You also must specify both SADMP(SA) and SDUMP (SD) parameters. The SDUMP value cannot be larger than SADMP value specified. Alternatively, you can use the following command for external output use:

   S GTF,,,(BLOK=10M),MEMBER=xxxx

   Do not use the NOPROMPT parameter because it causes the default of 40 K to be used for BLOK.

3. Add the TRACE subparameter to the AMP keyword in JCL to the DD statement for the VSAM KSDS to be traced as shown in Example 7-4. This example traces all the new hook points in Table 7-1 on page 378. It uses a PARM1 that traces the CMWA (note that bit 3 of byte 1 - X'10' is on) in addition to other control blocks.

Example 7-4 Tracing the CMWA and compression hook points

```
//KSDS01 DD DSN=VSAM.DATA.SET,DISP=SHR,
//     AMP=('TRACE=(PARM1=701001801122',
//     'HOOK=(20,21,22,23)')
```

PARM1=701001801122 specifies that the following items are traced:

- ACB
- AMB
- AMBL
- CMWA

- PLH
- RPL
- User's key
- User's record
- Index
- Data control blocks

It also runs a validity check on all control blocks and lists any that fail.

7.9 Related publications

Here we list documents and APARS that you might find helpful for further information.

7.9.1 IBM manuals and sources of relevant information

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ICF catalogs. Refer to *ICF Catalog Backup and Recovery: A Practical Guide*, SG24-5644.
- VSAM data sets in Record Level Sharing (RLS) mode. Refer to *CICS and VSAM Record Level Sharing: Recovery Considerations*, SG24-4768.
- APAR II08859 which has a methodology to assist you in fixing broken VSAM clusters.
- *z/OS FSMSdfp Diagnosis,* GY27-7618.

7.9.2 VSAM information on the Internet

These websites and URLs are also relevant as further information sources:

- DFSMS support sites
- Flashes:

  http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/Flashes

A

Sample code

This appendix contains useful JCL and code samples that can be modified and used in your installation.

This appendix contains the following sections:

- JRIO API examples
- Accessing the VSAM Shared Information (VSI)
- Sample programs extract from SMF record type 64
- REXX code to list compression ratio
- GTF procedure example

JRIO API examples

Here some examples of JRIO APIs for accessing VSAM data sets:

Locate a record by key in keyed access record file

```
IKeyedAccessRecordFile karf =
    new KeyedAccessRecordFile("//JOE.KSDS",
                      ;         JRIO_READ_MODE);
IRecordFile index = karf.getPrimaryIndex();
karf.positionForward(index, key);
karf.read(index, buffer);
karf.close();
```

Position to a record in a random access record file

```
IRandomAccessRecordFile rarf =
    new RandomAccessRecordFile("//JOE.KSDS",
                      ;         JRIO_READ_MODE);

rarf.read(buffer);   // reads the first record (record 0)

rarf.positionLast();
rarf.positionPrev();
rarf.read(buffer);   // reads the last record (record n-1)

rarf.positionFirst();
rarf.positionNext();
rarf.read(buffer);   // reads the second record (record 1)

rarf.seek(5L);
rarf.read(buffer);   // reads the sixth record (record 5)

rarf.close();
```

Read a record from a keyed access record file

```
IKeyedAccessRecordFile karf =
      new KeyedAccessRecordFile("//JOE.KSDS");
IRecordFile index = karf.getPrimaryIndex();
// or: IRecordFile index =
//       karf.getAlternateIndex("//JOE.KSDS.AIX");


byte[ ] buffer = new byte[JRIO_MAX_RECORD_LENGTH];

for(;;)
{
    // optional: karf.positionForward(key);
    // or:       karf.positionForwardGE(key);
    int bytesRead = karf.read(index, buffer);

    if (bytesRead != JRIO_READ_EOF)
    {
        // process(buffer);
    }
    else
    {
        break;
    }
}
karf.close();
```

Read a record from a random access record file

```
IRandomAccessRecordFile rarf =
       new RandomAccessRecordFile("//JOE.SEQ");

 byte[ ] buffer = new byte[JRIO_MAX_RECORD_LENGTH];

 for(;;)
 {
     // rarf.seek(recNo);
     int bytesRead = rarf.read(buffer);

     if (bytesRead != JRIO_READ_EOF)
     {
         // process(buffer);
     }
     else
     {
         break;
     }
 }
 rarf.close();
```

Update a record in a keyed access record file

```
IKeyedAccessRecordFile karf =
    new KeyedAccessRecordFile("//JOE.KSDS",
                  ;         JRIO_READ_WRITE_MODE);
IRecordFile index = karf.getPrimaryIndex();
karf.read(index, buffer);
// modify non-key field(s) in buffer
karf.update(index, buffer);
karf.close();
```

Accessing the VSAM Shared Information (VSI)

You can code the following instructions to get the length and address of VSI the data to be sent to another OS/390 image:

1. Load ACB address into register RY.
2. Locate the VSI for a data component:

L RX,04(,RY)	Put AMBL address into register RX
L 1,52(,RX)	Get data AMB address
L 1,68(,1)	Get VSI address
LH 0,62(,1)	Load data length
LA 1,62(,1)	Point to data to be communicated

3. Locate the VSI information for an index component of a key-sequenced data set:

`L  RX,04(,RY)`	Put AMBL address into register RX
`L  1,56(,RX)`	Get index AMB address
`L  1,68(,1)`	Get VSI address
`LH 0,62(,1)`	Load data length
`LA 1,62(,1)`	Point to data to be communicated

Similarly, the location of the VSI on the receiving processor can be located. The VSI level number must be incremented in the receiving VSI to inform the receiving processor that the VSI has changed. To update the level number, assuming that the address of the VSI is in register 1:

`LA 0,1`	Place increment into register 0
`AL 0,64(,1)`	Add level number to increment
`ST 0,64(,1)`	Save new level number

All processing of the VSI must be protected by using ENQ/DEQ to prevent simultaneous updates to the transmitted data.

Sample programs extract from SMF record type 64

This section provides two sample programs:

1. SMF64 helps you identify VSAM data sets with heavy access and good candidates for SMB.
2. SMFLSR helps you identify VSAM data sets opened in LSR mode. After they are identified, use the data set names as input for the IBM tool to see the probable change in CIsize with the new CI size calculation algorithm.

SMF64 sample code

The sample in Example A-1 can be used to find more information about jobs that use heavily accessed VSAM data sets. It is based on SMF record type 64 issued when the data set component is closed.

Example A-1 contains the following code:

- The parms that you can use to extract information
- The JCL you use to extract SMF records from SYS1.MAN*
- The JCL to run the sample program
- The report description fields

Example: A-1 SMF64 sample code

```
//*-----------------------------------------------------------------*
//*  JCL:                                                           *
//*  //RELAT   EXEC PGM=SMF64,PARM=SEE_BELOW                        *
//*  //STEPLIB  DD DSN=LOAD_LIBRARY,DISP=SHR                        *
//*  //SMF      DD DISP=SHR,DSN=QSAM_SMFDUMP                        *
//*  //REPORT   DD SYSOUT=*,LRECL=133                               *
//*                                                                 *
//*  THE PARM IS OPTIONAL: DEFAULT(EXCP=10000)                      *
//* ==> YOU CAN USE **ONLY** 1 OF 3 LISTED BELOW                    *
//*    DSN=DATA_SET_NAME                                            *
//*        LISTS INFORMATION ABOUT VSAM DATA SETS WHOSE NAME BEGINS  *
//*        WITH "DATA_SET_NAME".                                     *
//*        EX.: 'DSN=MY.DATA'  LISTS INFORMATION ABOUT ALL VSAM DATA *
//*             BEGINING WITH MY.DATA*                               *
//*        EX.: 'DSN=MY.DATA '  LIST ONLY DATA SET MY.DATA           *
//*                                                                  *
//*    JOB=MYJOB                                                     *
//*        LISTS ACCESS TO VSAM DATA SETS DONE BY ALL JOBS MYJOB*    *
//*        EX: PARM='JOB=MY.JOB'  LIST ALL JOBS MYJOB*               *
//*        EX: PARM='JOB=MYJOB ' LIST ONLY ABOUT MYJOB               *
//*                                                                  *
//*    EXCP=MAX_6_BYTES  LIST ONLY FOR DATA SET WITH EXCP>=PARM_EXCP *
//*                                                                  *
//* ==> YOU CAN A DATE AND/OR AN INTERVAL TO LIST                    *
//*    JD=AADDD  JULIAN DATE                                         *
//*    BH=HH INTERVAL BEGINING HOUR                                  *
//*    EH=HH INTERVAL FINAL HOUR                                     *
//*                                                                  *
//*    EXAMPLES:                                                     *
//*    PARM='DSN=XXX,BH=07,EH=10'                                    *
//*    PARM='DSN=XXX,JD=02360'                                       *
//*    PARM='EXCP=NNNN'                                              *
//*    PARM='JOB=JJJ,JD=02360,BH=07,EH=07'  (FROM 07:00 TO 07:59)    *
//*  ---------------------------------------------------------       *
//*  ---------------------------------------------------------       *
//*  JCL:  DUMP  SMF 64 RECORD FROM SYS1.MAN                         *
//*  //STEP1     EXEC  PGM=IFASMFDP
//*  //INDD      DD  DISP=SHR,DSN=SYS1.MAN?????
//*  //DUMPOUT   DD  DSN=QSAM_SMFDUMP_TYPE64,DISP=(,CATLG),
//*  //  SPACE=(CYL,(10,10),RLSE),RECFM=VBS,UNIT=SYSDA
//*  //SYSPRINT  DD    SYSOUT=A
//*  //SYSIN     DD  *
//*    INDD(INDD,OPTIONS(DUMP))
//*    OUTDD(DUMPOUT,TYPE(64:64))                                    *
//*                                                                  *
//* REPORT DESCRIPTION                                               *
//* XF=EXTENDED FORMAT DATA SET    N-XF(NON EXTENDED FORMAT)         *
//* XP=COMPRESSED FORMAT DATA SET  N-XP(NON COMPRESSED)              *
//* XA=EXTENDED ADDRESSABLE DATA SET  N-XA(NON EXTENDED ADDRESSAB.)  *
//* INSERT,DELETE,READ,UPDATE,GET, EXCP, CA AND CI SPLITS  ARE FROM  *
//*    OPEN TO CLOSE (ARE NOT ACCUMULATED)                           *
//* ACCESS BY; KEY, RBA, SEQ, DIR, SKIP                              *
//* PROCESSING OPTIONS:                                              *
//*   CNV: CONTROL INTERVAL ACCESS                                   *
//*   IN: INPUT                                                      *
//*   OUT: OUTPUT                                                    *
//*   ICI: IMPROVED CONTROL INTERVAL PROCESSING                      *
//*   DW: DEFERRED WRITE                                             *
//*   SIS: SEQUENTIAL INSERT STRATEGY                                *
//*   UB:  USER MANAGED BUFFERING                                    *
//*   CBSHR: VSAM STRUCTURE CONTROL BLOCK SHARED                     *
//*   CBFIX: VSAM CONTROL BLOCKS AND BUFFERS FIXED IN REAL STORAGE   *
//*   BUFFERING MANAGEMENT: LSR, GSR, NSR, RLS                       *
//*   BUFF31: 31-BIT ADDRESSING MODE FOR BUFFERS                     *
//*   BUFF24: 24-BIT ADDRESSING MODE FOR BUFFERS                     *
//*   SMB: OPTIMIZATION  USED:DO,DW,SO,SW,(OR CO,CR WHEN LOAD)       *
//*        VSP: USER SPECIFIED AMOUNT OF VIRTUAL STORAGE THRU SMBVSP *
//*        HWT: USER SPECIFIED HIPERSPACE BUFFER SMBHWT              *
//*        BUFF31: SMB USED 31-BIT ADDRESSING MODE FOR BUFFERS       *
//*        CB31: SMB USED 31-BIT ADDRESSING MODE FOR CONTROL BLOCKS  *
//*        IR: INSUFFICIENT VIRTUAL STORAGE FOR DO PROCESSING        *
//*------------------------------------------------------------------*
//COMP        EXEC  ASMACL,DPRTY=9,
// PARM.L='LIST,LET,XREF,MAP,AMODE=31,RMODE=24'
//C.SYSLIB  DD  DISP=SHR,DSN=SYS1.MACLIB
//C.SYSIN   DD  *
         TITLE ' SMF - RECORD TYPE 64 - VSAM STATS'
SMF64 CSECT
SMF64 AMODE 31
SMF64 RMODE 24
         STM   R14,R12,12(R13)     * LINKAGE CONVENTION
         LR    R12,R15             * R12 BASE
         USING SMF64,R12
         LA    R15,SAVE
         ST    R15,8(R13)
         ST    R13,SAVE+4
         LR    R13,R15
         L     R7,LIM_EXCP         * DEFAULT EXCPS
         LA    R6,RECORD           * REPORT MAPPING
         L     R5,0(R1)            * PARM
         LH    R4,0(R5)            * LENGTH
         LTR   R4,R4               * NO PARM?
         BZ    DO_OPEN             * YES, USE DEFAULT
         BCTR  R4,0                * -1   EXECUTE
         LA    R5,2(,R5)           *
         LA    R14,DO_OPEN
SEARCHP  EX    R4,TRT_PARM
         BNZ   CALC_LEN
         LR    R1,R4
         LA    R4,0
         B     ID_PARM
*
CALC_LEN LA    R8,1(,R1)
         SR    R1,R5
         SR    R4,R1
         BCTR  R1,0
         BAS   R14,ID_PARM
         LTR   R4,R4
         BZ    DO_OPEN
         LR    R5,R8
         BCTR  R4,0
         B     SEARCHP
*
ID_PARM  CLC   0(4,R5),=C'DSN='
         BNE   PARMJOB
         LA    R3,4
         SR    R1,R3
         BZR   R14
         LR    R11,R1
         EX    R1,MVDSN
         OI    FLAG,X'01'            * SELECT BY DSN
         MVI   TIT_OPT,X'40'
         MVC   TIT_OPT+1(L'TIT_OPT-1),TIT_OPT
         MVC   TIT_OPT(4),=C'DSN:'
         MVC   TIT_OPT+4(L'RDSN),RDSN
         BR    R14
*
PARMJOB  CLC   0(4,R5),=C'JOB='
         BNE   PARMEXCP
         LA    R3,4
         SR    R1,R3
         BNPR  R14
         LR    R11,R1
         EX    R1,MVJOBN
         OI    FLAG,X'02'          * SELECT BY JOBN
         MVI   TIT_OPT,X'40'
         MVC   TIT_OPT+1(L'TIT_OPT-1),TIT_OPT
         MVC   TIT_OPT(8),=C'JOBNAME:'
         MVC   TIT_OPT+9(L'RJOB),RJOB
         BR    R14
*
PARMEXCP CLC   0(5,R5),=C'EXCP='
         BNE   PARMJD
         LA    R3,5
         SR    R1,R3
         BNPR  R14
         MVI   T_EXCP,X'40'
         MVC   T_EXCP+1(L'T_EXCP-1),T_EXCP
         EX    R1,MVEXCP
         EX    R1,PACK
         CVB   R7,DOUBLE
         ST    R7,LIM_EXCP
         BR    R14
*
PARMJD   CLC   0(3,R5),=C'JD='
         BNE   PARMBH
         LA    R3,7
         CR    R1,R3
         BNER  R14                 * INVALID. IGNORES
         PACK  DATAJ,3(5,R5)
         OI    DATAJ+3,X'0F'
         OI    FLAG,X'04'
         BR    R14
*
PARMBH   CLC   0(3,R5),=C'BH='
         BNE   PARMEH              * INVALID, IGNORES
         LA    R3,4
         CR    R1,R3
         BNER  R14                 * INVALID. IGNORES
         TM    3(R5),X'F0'
         BNOR  R14
         TM    4(R5),X'F0'
         BNOR  R14
         PACK  DOUBLE,3(2,R5)
         CVB   R3,DOUBLE
         LTR   R3,R3
         BZR   R14
         M     R2,F3600            * SECONDS
         M     R2,F100             * SECONDS * 100
         ST    R3,T_INIT
         OI    FLAG,X'08'          * FLAG INITIAL INTERVAL
         BR    R14
*
PARMEH   CLC   0(3,R5),=C'EH='
         BNER  R14
         LA    R3,4
         CR    R1,R3
         BNER  R14                 * INVALID. IGNORES
         TM    3(R5),X'F0'
         BNOR  R14
         TM    4(R5),X'F0'
         BNOR  R14
         PACK  DOUBLE,3(2,R5)
         CVB   R3,DOUBLE
         LTR   R3,R3
         BZR   R14
         LA    R3,1(,R3)           * HOUR LIMIT FOR COMPARATION
         M     R2,F100             *
         M     R2,F3600            * SECONDS * 100
         BCTR  R3,0                *
         ST    R3,T_FIM
         OI    FLAG,X'80'          * FINAL INTERVAL FLAG
         BR    R14
*
DO_OPEN  OPEN  (SMFDUMP,,REC,(OUTPUT))
         LTR   R15,R15
         BNZ   ABEND
         LA    R8,0                * TITLE CONTROL
NUMLIN   LA    R9,55               * 55 LINES PER PAGE
READ     GET   SMFDUMP
         LR    R10,R1              * IDASMF64
         USING SMFRCD64,R10
         CLC   SMF64LEN,HDUMP
         BNH   READ
         CLI   SMF64RTY,64         * 64?
         BNE   READ
         CLI   2(R10),X'00'
         BNE   READ
         TM    SMF64RIN,X'80'      * COMPONENT CLOSE?
         BZ    READ
         TM    SMF64DTY,X'80'      * DATA SET?
         BNO   READ
         LA    R1,0
         ICM   R1,B'0011',SMF64ESL
         LA    R15,0(R1,R10)
         TM    FLAG,X'01'          * DSN?
         BO    PR_DSN
         TM    FLAG,X'02'          * JOB?
         BNO   PR_EXCP
*
PR_JOB   EQU   *
         EX    R11,CPJOB           * JOBNAME MATCHES?
         BNE   READ
         B     COMMON
*
PR_DSN   EQU   *
         EX    R11,CPDSN           * DSN MATCHES?
         BNE   READ
         B     COMMON
*
PR_EXCP  EQU   *
         DROP  R10
         USING SMFRCD64,R15
         L     R3,SMF64DEP          EXCP FOR DATA SET
         CR    R3,R7                COMPARE WITH LIMIT
         BL    READ
         B     COMMON
         DROP  R15
*
COMMON DS      0H
```

```
         USING SMFRCD64,R10        SMF RECORD MAPPING
         MVC   R_SYSID,SMF64SID    SYSID
         MVC   R_JOB,SMF64JBN      JOBNAME
         TM    FLAG,X'04'          DATE?
         BNO   MV_DATA
         CP    SMF64DTE+1(3),DATAJ+1(3)  COMPARE DATE
         BNE   READ
MV_DATA  UNPK  DATA,SMF64DTE
         TM    FLAG,X'88'          INTERVAL SELECTION?
         BNO   MV_TME              IGNORES IF NOT COMPLETE INTERVAL
         CLC   SMF64TME,T_INIT
         BL    READ
         CLC   SMF64TME,T_FIM
         BH    READ
MV_TME   ICM   R3,B'1111',SMF64TME
         L     R1,F100
         LA    R2,0
         DR    R2,R1              * SECONDS
         L     R1,F3600           * SECONDS IN  HOURS
         LA    R2,0
         DR    R2,R1              * HOUR EM R3
         CVD   R3,DOUBLE
         UNPK  HH,DOUBLE+6(2)
         OI    HH+1,X'F0'
         L     R1,F60
         LR    R3,R2               * REMAINING
         LA    R2,0
         DR    R2,R1               * MINUTS
         CVD   R3,DOUBLE
         UNPK  MM,DOUBLE+6(2)
         OI    MM+1,X'F0'
         CVD   R2,DOUBLE           * REMAINING SECOUNDS
         UNPK  SS,DOUBLE+6(2)
         OI    SS+1,X'F0'
         LA    R4,R_ACCESS
         TM    SMF64DTY,X'20'      EXTENDED FORMAT?
         BNO   N_XF                DOES NOT USES EXT FACILITIES
         MVC   0(L'XF,R4),XF
         LA    R4,L'XF+1(,R4)
         TM    SMF64DTY,X'10'      * COMPRESSED?
         BNO   IF_XA                 NO. JUMP
         MVC   0(2,R4),=C'XP'      SET COMPRESSED
         LA    R4,3(,R4)
IF_XA    TM    SMF64DTY,X'02'      EXTENDED ADDRESS DATA SET?
         BNO   N_XA
         MVC   0(L'XA,R4),XA       SET EXTENDED ADDRESSABILITY
         LA    R4,L'XA+1(,R4)
         B     IF_RLS
*
N_XF     MVC   0(L'NXF,R4),NXF     SET NON EXTENDED FORMAT
         LA    R4,L'NXF+1(,R4)
*
N_XA     MVC   0(L'NXA,R4),NXA     SET NON EXTENDED ADDRESS DATA
         LA    R4,L'NXA+1(,R4)
IF_RLS   TM    SMF64DTY,X'08'      RECORD LEVEL SHARED?
         BNO   M_EXCP
         MVC   0(L'RLS,R4),RLS     SET RLS
         LA    R4,L'RLS+1(,R4)
*
         DROP  R10
         USING SMFRCD64,R15
*
M_EXCP   L     R3,SMF64DEP          EXCPS
         CVD   R3,DOUBLE            ZONED DECIMAL
         MVC   EXCPS,MODEL
         ED    EXCPS,DOUBLE+2
         MVC   R_DDN,SMF64DDN      * DDNAME
         MVC   R_DSN,SMF64CLN      * CLUSTER NAME
         L     R3,SMF64DIN         * INSERTIONS
         CVD   R3,DOUBLE
         MVC   INSERT,MODEL
         ED    INSERT,DOUBLE+2
         L     R3,SMF64DDE         * RECORDS DELETED
         CVD   R3,DOUBLE
         MVC   DELETE,MODEL
         ED    DELETE,DOUBLE+2
         L     R3,SMF64DUP         * RECORDS UPDATED
         CVD   R3,DOUBLE
         MVC   UPDATE,MODEL
         ED    UPDATE,DOUBLE+2
         L     R3,SMF64DRE         * RECORDS RETRIEVED
         CVD   R3,DOUBLE
         MVC   READS,MODEL
         ED    READS,DOUBLE+2
         L     R3,SMF64DCS         * CONTROL INTERVAL SPLITS
         CVD   R3,DOUBLE
         MVC   CI_SPLIT,MODEL
         ED    CI_SPLIT,DOUBLE+2
         L     R3,SMF64DAS         * CONTROL AREA SPLITS
         CVD   R3,DOUBLE
         MVC   CA_SPLIT,MODEL
         ED    CA_SPLIT,DOUBLE+2
         TM    SMF64MC1,X'80'      * KEY ACCESS?
         BNO   IF_RBA
         MVC   0(3,R4),=C'KEY'
         LA    R4,4(R4)
         B     TYPEACC
*
IF_RBA   TM    SMF64MC1,X'40'      * RBA?
         BNO   IF_CI
         MVC   0(3,R4),=C'RBA'
         LA    R4,4(R4)
         B     TYPEACC
*
IF_CI    TM    SMF64MC1,X'20'      * ACCESS BY CONTROL INTERVAL?
         BNO   TYPEACC
         MVC   0(3,R4),=C'CNV'
         LA    R4,4(R4)
         B     TYPEACC
*
TYPEACC  TM    SMF64MC1,X'10'      * SEQUENTIAL ACCESS?
         BNO   IF_DIR
         MVC   0(3,R4),=C'SEQ'
         LA    R4,4(R4)
*
IF_DIR   TM    SMF64MC1,X'08'      * DIRECT?
         BNO   IF_INP
         MVC   0(3,R4),=C'DIR'
         LA    R4,4(R4)
IF_INP   TM    SMF64MC1,X'04'      * INPUT?
```

```
         BNO   IF_OUT
         MVC   0(2,R4),=C'IN'
         LA    R4,3(R4)
*
IF_OUT   TM    SMF64MC1,X'02'       * OUTPUT?
         BNO   IF_UBUF
         MVC   0(3,R4),=C'OUT'
         LA    R4,4(R4)
IF_UBUF  TM    SMF64MC1,X'01'       * USER MANAGED BUFFERING
         BNO   MC2
         MVC   0(2,R4),=C'UB'
         LA    R4,3(R4)
MC2      TM    SMF64MC2,X'10'      SKIP SEQUENTIAL?
         BNO   IF_CBS
         MVC   0(4,R4),=C'SKIP'
         LA    R4,5(R4)
IF_CBS   TM    SMF64MC2,X'02'      SHARED CONTROL BLOCKS?
         BNO   IF_LSR
         MVC   0(5,R4),=C'CBSHR'
         LA    R4,6(R4)
IF_LSR   TM    SMF64MC3,X'40'      LSR?
         BNO   IF_GSR
         MVC   0(3,R4),=C'LSR'
         LA    R4,4(R4)
         B     IF_DW
*
IF_GSR   TM    SMF64MC3,X'20'      GSR
         BNO   IF_ICI
         MVC   0(3,R4),=C'GSR'
         LA    R4,4(R4)
         B     IF_DW
*
IF_ICI   TM    SMF64MC3,X'10'      IMPROVED CONTROL-INTERVAL
         BNO   NSR
         MVC   0(3,R4),=C'ICI'
         LA    R4,4(R4)
         B     IF_DW
*
NSR      MVC   0(3,R4),=C'NSR'
         LA    R4,4(R4)
*
IF_DW    TM    SMF64MC3,X'08'      DEFERRED-WRITE?
         BNO   IF_SIS
         MVC   0(2,R4),=C'DW'
         LA    R4,3(R4)
IF_SIS   TM    SMF64MC3,X'04'      SEQUENTIAL INSERT STRATEGY?
         BNO   IF_FIX
         MVC   0(3,R4),=C'SIS'
         LA    R4,4(R4)
*
IF_FIX   TM    SMF64MC3,X'02'      CB FIXED IN REAL?
         BNO   IF_BUFF
         MVC   0(5,R4),=C'CBFIX'
         LA    R4,6(R4)
*
IF_BUFF  MVC   0(6,R4),BUFF31
         TM    SMF64MC3,X'01'      BUFFERS ABOVE?
         BO    IF_SMB
         MVC   0(6,R4),BUFF24
IF_SMB   LA    R4,7(,R4)
```

```
         TM    SMF64SMB,X'80'     SMB USED?
         BNO   PUTREC
         MVC   0(3,R4),=C'SMB'
         LA    R4,4(,R4)
         TM    SMF64SMB,X'02'     LOAD OPTIMIZED FOR SPEED?
         BNO   IF_CR
         MVC   4(2,R4),=C'CO'
         B     IF_DO
IF_CR    TM    SMF64SMB,X'01'     LOAD OPTIMIZED FOR RECOVERY?
         BNO   IF_CR
         MVC   4(2,R4),=C'CR'
IF_DO    LA    R4,3(,R4)
         TM    SMF64SMB,X'20'     DO BIAS USED?
         BNO   IF_SO
         MVC   4(2,R4),=C'DO'
         B     IF_VSP
IF_SO    TM    SMF64SMB,X'10'     SO BIAS USED?
         BNO   IF_SW
         MVC   4(2,R4),=C'SO'
         B     IF_VSP
IF_SW    TM    SMF64SMB,X'08'     SW BIAS USED?
         BNO   IF_DFW
         MVC   4(2,R4),=C'SW'
         B     IF_VSP
IF_DFW   TM    SMF64SMB,X'04'     DW BIAS USED?
         BNO   IF_VSP
         MVC   4(2,R4),=C'DW'
         B     TM_VSP
IF_VSP   LA    R4,3(,R4)
TM_VSP   TM    SMF64RSC,X'80'     AMOUNT OF VIRTUAL BUFFER INFORMED
         BNO   IF_HWT
         MVC   0(3,R4),VSP
         LA    R4,4(,R4)
IF_HWT   TM    SMF64RSC,X'40'     USED HIPERSPACE BUFFERS?
         BNO   IF_RE31
         MVC   0(3,R4),HWT
         LA    R4,4(,R4)
IF_RE31  TM    SMF64RSC,X'20'     RMODE31=BUFF USED
         BNO   IF_CB31
         MVC   0(7,R4),BUFF31
IF_CB31  TM    SMF64RSC,X'10'     RMODE31=CB   USED
         BNO   IF_878
         MVC   0(5,R4),CB31
IF_878   TM    SMF64RSC,X'10'     RMODE31=CB   USED
         BNO   PUTREC
         MVC   0(5,R4),IREG
*
PUTREC   LTR   R8,R8              TITLE?
         BNZ   GR_REC
         PUT   REC,TIT
         PUT   REC,CAB
         LA    R8,15
GR_REC   PUT   REC,RECORD
         PUT   REC,REGL2
         PUT   REC,REGL3
         MVI   R_ACCESS,C' '
         MVC   R_ACCESS+1(L'R_ACCESS-1),R_ACCESS
         BCT   R9,READ
         PUT   REC,TIT
         PUT   REC,CAB
         B     NUMLIN
         DROP  R15
*
GOBACK   CLOSE (SMFDUMP,,REC)
         SVC   3
ABEND    LR   R7,R15
         WTO 'OPEN ERROR',ROUTCDE=11
         ABEND 1000,DUMP,STEP
*
*-------------------------------------------------------------------*
*  EXECUTE INSTRUCTIONS
*-------------------------------------------------------------------*
MVDSN    MVC   RDSN(0),4(R5)
MVJOBN   MVC   RJOB(0),4(R5)
MVEXCP   MVC   T_EXCP(0),5(R5)
PACK     PACK  DOUBLE,5(0,R5)
         USING SMFRCD64,R10
CPJOB    CLC   RJOB(0),SMF64JBN
         DROP  R10
         USING SMFRCD64,R15
CPDSN    CLC   RDSN(0),SMF64CLN
         DROP  R15
TRT_PARM TRT   0(0,R5),TBVIRG
*-------------------------------------------------------------------*
*  DCBS
*-------------------------------------------------------------------*
SMFDUMP  DCB   DSORG=PS,EODAD=GOBACK,MACRF=(GL),DDNAME=SMF
REC      DCB   DSORG=PS,MACRF=(PM),DDNAME=REPORT,LRECL=133
*-------------------------------------------------------------------*
*  WORK AREA
*-------------------------------------------------------------------*
SAVE DC      18F'0'      LINKAGE CONVENTION AREA
SPANNED  DC    F'0'
DOUBLE   DS    D
LIM_EXCP DC    F'10000'
HDUMP    DC    H'18'     SKIP DUMP AND TRAILER SMF RECORD
F3600    DC    F'3600'
F100     DC    F'100'
F60      DC    F'60'
FLAG     DC    X'00'
TIT      DC    134C' '
         ORG   TIT
         DC    C'1'
         ORG   TIT+10
         DC    C'VSAM - ACCESS STATISTICS '
TITEXCP  DC    C'EXCP COUNT >= '
T_EXCP   DC    C'10000   '
         ORG   TIT+36
TIT_OPT  DS    CL54
         ORG   ,
CAB      DS    0CL134
         DC    134C' '
         ORG   CAB+1
         DC    C'JOBNAME'
         DS    CL2
         DC    C'DATA SET NAME'
         DS    CL32
         DC    C'DDNAME'
         DS    CL3
         DC    C'SYS'
         DS    CL2
         DC    C'TIMESTAMP'
         DS    CL6
         DC    C'MISCELLANEOUS: ACCESS, STATS SINCE LAST OPEN'
         ORG   ,
         DS    0F
RECORD DS      0CL134
         DC    134C' '
         ORG   RECORD+1
R_JOB    DS    CL8
         DS    CL1
R_DSN    DS    CL44
         DS    CL1
R_DDN    DS    CL8
         DS    CL1
R_SYSID  DS    CL4
         DS    CL1
R_STAMP  DS    0CL14
DATA     DS    CL5
         DS    CL1
HH       DS    CL2
         DC    C':'
MM       DS    CL2
         DC    C':'
SS       DS    CL2
         ORG   ,
REGL2    DS    0CL134
         DC    134C' '
         ORG   REGL2+1
         DC    C'EXCPS:'
EXCPS    DS    CL12
         DS    CL1
R_ACCESS DS    CL114
         ORG   ,
REGL3    DS    0CL134
         DC    134C' '
         ORG   REGL3+1
         DC    C'INSERTS:'
INSERT   DS    CL12
         DS    CL1
         DC    C'DELETES:'
DELETE   DS    CL12
         DS    CL1
         DC    C'UPDATES:'
UPDATE   DS    CL12
         DS    CL1
         DC    C'READ:'
READS    DS    CL12
         DS    CL1
         DC    C'CI-SPLIT:'
CI_SPLIT DS    CL12
         DS    CL1
         DC    C'CA-SPLIT:'
CA_SPLIT DS    CL12
         ORG   ,
MODEL    DC  X'402020202020202020202120'
RDSN     DS    0CL44
         DC    44C' '
RJOB     DS    0CL8
         DC    8C' '
```

```
NXF      DC    CL4'N-XF'
         ORG   NXF+2
XF       DS    CL2
         ORG   ,
NXA      DC    CL4'N-XA'
         ORG   NXA+2
XA       DS    CL2
         ORG   ,
RLS      DC    C'RLS'
DATAJ    DS    XL4
         DS    0F
T_INIT   DS    X'00000000'
T_FIM    DS    X'00000000'
BUFF31   DC    C'BUFF31'
BUFF24   DC    C'BUFF24'
CB31     DC    C'CB31'
VSP      DC    C'VSP'
HWT      DC    C'HWT'
IREG     DC    C'IR'
TBVIRG   DC    256X'00'
         ORG   TBVIRG+C','
         DC    X'01'
         ORG   ,
*---------------------------------------------------------------------*
*  DUMMIES E  EQUATES
*---------------------------------------------------------------------*
         LTORG
         YREGS
         DSECT
         IDASMF64
         END   SMF64
//*-------------------------------------------------------------------*
//L.SYSLMOD DD DISP=SHR,DSN=LOAD_LIBRARY
//L.SYSLIB  DD DISP=SHR,DSN=LOAD_LIBRARY
//L.SYSIN   DD  *
  ENTRY   SMF64
  NAME    SMF64(R)
/*
```

SMFLSR sample program

Since z/OS V1 R3, the algorithm to calculate the minimum index CI size has changed. The assembler sample code in Example A-2 helps you identify VSAM data sets opened in LSR mode. The change in calculation can cause the following problems:

- Under CICS, if you are explicitly coding the number of buffers of each size in the LSRPOOL, two things can happen:
  - The number of buffers for the new size is not enough, leading to performance problems.
  - Buffers for the new index CI size do not exist, causing the open to fail. The buffer size must be the same size of the index CI size or larger.
- Under other applications, when the CI index size buffer is defined in BLDVRP and is lower than the new index CI size calculation.

Example: A-2 SMFLSR sample program

```
//*-------------------------------------------------------------------*
//*  JCL:                                                             *
//*  //RELAT   EXEC PGM=SMFLSR                                        *
//*  //STEPLIB  DD DSN=LOAD_LIBRARY,DISP=SHR                          *
//*  //SMF      DD DISP=SHR,DSN=QSAM_SMFDUMP                          *
//*  //REPORT   DD SYSOUT=*,LRECL=133                                 *
//*                                                                   *
//* REPORT DESCRIPTION:                                               *
//* SYSID JOBNAME DDNAME DATASET NAME CI-SIZE KEY-LENGTH               *
//*                                                                   *
//* RETURN CODES:                                                     *
//*    0 - FOUND DATA SETS IN LSR MODE                                *
//*    4 - NO FOUND                                                   *
//*=================================================================*
//*                                                                   *
//*  JCL: DUMP  SMF 64 RECORD FROM SYS1.MAN                           *
//*                                                                   *
//*  //STEP1     EXEC  PGM=IFASMFDP                                   *
//*  //INDD      DD  DISP=SHR,DSN=SYS1.MAN?????                       *
//*  //DUMPOUT   DD  DSN=QSAM_SMFDUMP_TYPE64,DISP=(,CATLG),           *
//*  //  SPACE=(CYL,(10,10),RLSE),RECFM=VBS,UNIT=SYSDA                *
//*  //SYSPRINT  DD    SYSOUT=A                                       *
//*  //SYSIN     DD  *                                                *
//*    INDD(INDD,OPTIONS(DUMP))                                       *
//*    OUTDD(DUMPOUT,TYPE(64:64))                                     *
//*                                                                   *
//*-----------------------------------------------------------------*
//COMP        EXEC  ASMACL,DPRTY=9,
// PARM.L='LIST,LET,XREF,MAP,AMODE=31,RMODE=24'
//C.SYSLIB  DD  DISP=SHR,DSN=SYS1.MACLIB
//C.SYSIN   DD  *
         TITLE ' SMF - RECORD TYPE 64 - VSAM OPEN FOR LSR MODE'
SMFLSR CSECT
SMFLSR AMODE 31
SMFLSR RMODE 24
         STM   R14,R12,12(R13)     * LINKAGE CONVENTION
         LR    R12,R15             * R12 BASE
         USING SMFLSR,R12
         LA    R15,SAVE            * END SAVE DESTE PGM
         ST    R15,8(R13)
         ST    R13,SAVE+4          * SALVA END SAVE ANTERIOR
         LR    R13,R15
         LA    R6,4                * RC NOT FOUND
DO_OPEN  OPEN  (SMFDUMP,,REC,(OUTPUT))
         LTR   R15,R15
         BNZ   ABEND
SET_TIT  LA    R8,0                * TITLE CONTROL
         LA    R9,66               * 66 LINES PER PAGE
READ     GET   SMFDUMP
         LR    R10,R1              * IDASMF64
         USING SMFRCD64,R10
         CLC   SMF64LEN,HDUMP
         BNH   READ
         CLI   SMF64RTY,64         * 64?
         BNE   READ
         CLI   2(R10),X'00'        * DISCARD SPANNED RECORDS
         BNE   READ
         TM    SMF64RIN,X'E0'      * CLOSE,VOL SWITCH,OUT OF SPACE
         BZ    READ                * NO. DISCARD
         TM    SMF64DTY,X'40'      * INDEX?
         BNO   READ                * NO.DISCARD
         LA    R1,0
         ICM   R1,B'0011',SMF64ESL
         LA    R15,0(R1,R10)
         DROP  R10
         USING SMFRCD64,R15
         TM    SMF64MC3,X'40'     LSR?
         BNO   READ               NO, DISCARD
         TM    SMF64SMB,X'80'     SMB USED?
         BO    READ               YES. DISCARD. SMB BUILDS BUFFER POOL
         DROP  R15                ACCORDING TO CISIZE
*
         USING SMFRCD64,R10        SMF RECORD MAPPING
         MVC   R_SYSID,SMF64SID    SYSID
         MVC   R_JOB,SMF64JBN      JOBNAME
*
         DROP  R10
         USING SMFRCD64,R15
*
         MVC   R_DDN,SMF64DDN      * DDNAME
         MVC   R_DSN,SMF64CLN      * CLUSTER NAME
         L     R3,SMF64DCI         * CI-SIZE
         CVD   R3,DOUBLE
         MVC   R_CI,MODEL
         ED    R_CI,DOUBLE+4
         LA    R3,0
         ICM   R3,B'0011',SMF64DKL         * KEY-SIZE
         CVD   R3,DOUBLE
         MVC   R_KEY,MODEL+4
         LTR   R8,R8              TITLE?
         BNZ   GR_REC
         PUT   REC,TIT
         PUT   REC,CAB
         LA    R8,15
GR_REC   PUT   REC,RECORD
         LA    R6,0
         BCT   R9,READ
         B     SET_TIT
         DROP  R15
*
GOBACK   CLOSE (SMFDUMP,,REC)
         LR    R15,R6             RETURN CODE
         L     R13,SAVE+4         LINKAGE CONVENTION
         L     R14,12(R13)
         LM    R0,R12,20(R13)
         BR    R14
*
ABEND    LR   R7,R15
         WTO 'OPEN ERROR',ROUTCDE=11
         ABEND 1000,DUMP,STEP
*
*-------------------------------------------------------------------*
*  DCBS
*-------------------------------------------------------------------*
SMFDUMP  DCB   DSORG=PS,EODAD=GOBACK,MACRF=(GL),DDNAME=SMF
REC      DCB   DSORG=PS,MACRF=(PM),DDNAME=REPORT,LRECL=133
*-------------------------------------------------------------------*
*  WORK AREA
*-------------------------------------------------------------------*
SAVE DC      18F'0'      LINKAGE CONVENTION AREA
SPANNED  DC    F'0'
DOUBLE   DS    D
```

```
HDUMP    DC    H'18'     SKIP DUMP AND TRAILER SMF RECORD
TIT      DC    134C' '
         ORG   TIT
         DC    C'1'
         ORG   TIT+10
         DC    C'VSAM - DATA SET WITH LSR ACCESS'
         ORG   ,
CAB      DS    0CL134
         DC    134C' '
         ORG   CAB+1
         DC    C'SYSTEM '
         DS    CL2
         DC    C'JOBNAME'
         DS    CL3
         DC    C'DDNAME'
         DS    CL4
         DC    C'DATA SET NAME'
         DS    CL32
         DC    C'KEY SIZE'
         DS    CL2
         DC    C'INDEX CI SIZE'
         ORG   ,
         DS    0F
RECORD DS      0CL134
         DC    134C' '
         ORG   RECORD+1
R_SYSID  DS    CL4
         DS    CL5
R_JOB    DS    CL8
         DS    CL2
R_DDN    DS    CL8
         DS    CL2
R_DSN    DS    CL44
         DS    CL1
R_KEY    DS    CL6
         DS    CL4
R_CI     DS    CL6
         ORG   ,
MODEL    DC  X'402020202120'
*-------------------------------------------------------------------*
*  DUMMIES E  EQUATES
*-------------------------------------------------------------------*
         LTORG
         YREGS
         DSECT
         IDASMF64
         END   SMFLSR
//*-----------------------------------------------------------------*
//L.SYSLMOD DD DISP=SHR,DSN=LOAD_DATA_SET
//L.SYSLIB  DD DISP=SHR,DSN=LOAD_DATA_SET
//L.SYSIN   DD  *
  ENTRY   SMFLSR
  NAME    SMFLSR(R)
/*
```

REXX code to list compression ratio

Compression is useful when the compression rate is high to compensate for the processor cycles that are used to run the compression. The compression rate depends on the type of data.

You can use the REXX in Example A-4 to list the compress ratio for these data sets:

- All data sets (VSAM KSDS or non-VSAM) from a catalog.
- A specific data set

You can run the REXX by using these methods:

- TSO
- Batch, using the JCL shown in Example A-3

Example: A-3 Sample JCL for running REXX in batch

```
//REXX      EXEC  PGM=IRXJCL,
//  PARM='REXX_NAME +UCAT.VSBOX01'
//SYSTSPRT  DD  SYSOUT=*
//SYSEXEC   DD DISP=SHR,DSN=PDS_REXX
//SYSTSIN   DD DUMMY
```

Example A-4 shows the REXX code.

Example: A-4 REXX code

```
/* REXX */
/*TRACE R  */
 /*****************************************************************/
 /* FUNCTION:                                                     */
 /* DISPLAY THE COMPRESSION RATIO FOR:                            */
 /*  1 -  DATA SET: EXEC REXX_NAME 'DATA_SET_NAME'                */
 /*  OR, WHEN THE PARM IS PREFIXED BY "+":                        */
 /*  2 -  FOR ALL COMPRESSED DATA SETS FROM A CATALOG:            */
 /*      EXEC REXX_NAME '+CATALOG_NAME'                           */
 /*****************************************************************/
 PARSE UPPER ARG DSN
 IF DSN = '' THEN
    DO
    SAY "NO DSN OR CATALOG INFORMED. "
    EXIT
    END
 PARSE VAR DSN '+' CAT
 IF CAT /= '' THEN  DSN = '**'
    ELSE CAT = ' '
MODRSNRC = SUBSTR(' ',1,4)
CSIFILTK = SUBSTR(DSN,1,44)
CSICATNM = SUBSTR(CAT,1,44)
CSIRESNM = SUBSTR(' ',1,44)
CSIDTYPS = SUBSTR(' ',1,16)
/*  OPTIONS */
CSICLDI  = SUBSTR(' ',1,1)
CSIRESUM = SUBSTR(' ',1,1)          /*   RESUME FLAG                */
CSIS1CAT = SUBSTR('Y',1,1)          /*   SEARCH MORE THAN 1 CATALOG  */
CSIRESRV = SUBSTR(' ',1,1)
CSINUMEN = '0003'X

CSIFLD1  = SUBSTR('COMPIND COMUDSIZUDATASIZ',1,24)

CSIOPTS  = CSICLDI || CSIRESUM || CSIS1CAT || CSIRESRV
```

```
CSIFIELD = CSIFILTK || CSICATNM || CSIRESNM || CSIDTYPS || CSIOPTS
CSIFIELD = CSIFIELD || CSINUMEN || CSIFLD1
WORKLEN = 1024
DWORK = '00000400'X || COPIES('00'X,WORKLEN-4)
FOUND = 0
NUMERIC DIGITS 16

RESUME = 'Y'

DO WHILE RESUME = 'Y'
   ADDRESS LINKPGM 'IGGCSI00  MODRSNRC  CSIFIELD  DWORK'
   LCC = RC
   IF LCC ¬= 0 THEN
      DO
      REASON = C2X(MODRSNRC)
      SAY 'IGGCSI00 NOT OK RC:' LCC 'REASON:' REASON
      EXIT
      END
   RESUME = SUBSTR(CSIFIELD,150,1)  /* RESUME FLAG LOOP CTL */
   USEDLEN = C2D(SUBSTR(DWORK,9,4)) /* USED WORK AREA LENGTH  */
   POS1=15                          /* AREA INFO ENTRY  */
   /***********************************************************/
   /*  PROCESS WORK AREA DATA                                 */
   /***********************************************************/
   DO WHILE POS1 < USEDLEN            /* PROCESS  WORK AREA */
      IF SUBSTR(DWORK,POS1+1,1) = '0' THEN POS1 = POS1 + 50
      ELSE
         DO
         CSIEFLAG = SUBSTR(DWORK,POS1,1)
         X = BITAND(CSIEFLAG,'20'X)            /*  VALID? */
         IF X /= '20'X THEN  POS1 = POS1 + 50
         ELSE
             DO
             X = BITAND(SUBSTR(DWORK,POS1+56,1),'60'X)
             TOTDATA = C2D(SUBSTR(DWORK,POS1+46,2))
             IF X = '60'X THEN
                DO
                X =  BITAND(SUBSTR(DWORK,POS1+57,1),'FF'X)
                IF X /= 'FF'X  THEN
                  DO
                  FOUND = 1
                  DSN = SPACE(SUBSTR(DWORK,POS1+2,44),0)
                  STLEN = C2D(SUBSTR(DWORK,POS1+50,2))
                  NDLEN = C2D(SUBSTR(DWORK,POS1+52,2))
                  RDLEN = C2D(SUBSTR(DWORK,POS1+54,2))
                  USIZE = C2D(SUBSTR(DWORK,POS1+65,RDLEN))
                  XPRESS = C2D(SUBSTR(DWORK,POS1+57,NDLEN))
                  RATIO = SPACE(FORMAT((1-(XPRESS/USIZE))*100,16,2),0)
                  SAY "COMPRESSION RATIO OF  " DSN ": " RATIO "%"
                  END
                END
             POS1 = POS1 + TOTDATA + 46
             END
         END
   END
ENDIF FOUND = 0 THEN SAY "NO COMPRESSED DATA SETS FOUND "
```

SMFRLS sample program

Example A-5 shows an SMFRLS sample program.

Example: A-5 SMFRLS sample program

```
//*-----------------------------------------------------------------*
//*  FUNCTION:                                                      *
//*  RLS CF STRUCTURE AND I/O STATISTICS                            *
//*                                                                 *
//*-----------------------------------------------------------------*
//* REPORT DESCRIPTION                                              *
//*      # OF REQUESTS WHERE DATA WAS OBTAINED FROM THE LOCAL       *
//*        SHARED BUFFER POOL                                       *
//*      # OF REQUESTS WHERE DATA WAS OBTAINED FROM DE CF CACHE     *
//*      # OF REQUESTS WHERE DATA WAS OBTAINED FROM DASD            *
//*      # OF UPDATES, INSERTS, DELETES, READS, SPLITS CI AND CA    *
//* SYSTEM ID, JOBNAME, DATE AND TIME WHEN JOB STARTED, DDNAME AND   *
//* DATA SET NAME                                                   *
//*-----------------------------------------------------------------*
//*  JCL:                                                           *
//*  //RELAT   EXEC PGM=SMFRLS                                      *
//*  //STEPLIB  DD DSN=LOAD_LIBRARY,DISP=SHR                        *
//*  //SMF      DD DISP=SHR,DSN=QSAM_SMFDUMP                        *
//*  //REPORT   DD SYSOUT=*,LRECL=133                               *
//*                                                                 *
//*  PARM: NO PARM USED                                             *
//*  --------------------------------------------------------       *
//*  JCL DUMP  SMF 64 RECORD FROM SYS1.MAN                          *
//*  //STEP1     EXEC  PGM=IFASMFDP
//*  //INDD      DD  DISP=SHR,DSN=SYS1.MAN?????
//*  //DUMPOUT   DD  DSN=QSAM_SMFDUMP_TYPE64,DISP=(,CATLG),
//*  //  SPACE=(CYL,(10,10),RLSE),RECFM=VBS,UNIT=SYSDA
//*  //SYSPRINT  DD    SYSOUT=A
//*  //SYSIN     DD  *
//*    INDD(INDD,OPTIONS(DUMP))
//*    OUTDD(DUMPOUT,TYPE(64:64))                                   *
//*-----------------------------------------------------------------*
//COMP        EXEC  ASMACL,DPRTY=9,
// PARM.L='LIST,LET,XREF,MAP,AMODE=31,RMODE=24'
//C.SYSLIB  DD  DISP=SHR,DSN=SYS1.MACLIB
//C.SYSIN   DD  *
         TITLE ' SMF - RECORD TYPE 64 - RLS HITS STATS'
SMFRLS CSECT
SMFRLS AMODE 31
SMFRLS RMODE 24
         STM   R14,R12,12(R13)     * LINKAGE CONVENTION
         LR    R12,R15             * 12 BASE REGISTER
         USING SMFRLS,R12
         LA    R15,SAVE
         ST    R15,8(R13)
         ST    R13,SAVE+4
         LR    R13,R15
*
DO_OPEN  OPEN  (SMFDUMP,,REC,(OUTPUT))
         LTR   R15,R15
         BNZ   ABEND
SET_TIT  LA    R8,0                * TITLE CONTROL
         LA    R9,14               * 60 LINES PER PAGE
READ     GET   SMFDUMP
         LR    R10,R1              * IDASMF64
         USING SMFRCD64,R10
         CLC   SMF64LEN,HDUMP
         BNH   READ
         CLI   SMF64RTY,64         * 64?
         BNE   READ
         CLI   2(R10),X'00'
         BNE   READ
         TM    SMF64RIN,X'80'      * COMPONENT CLOSE?
         BZ    READ
         TM    SMF64DTY,X'80'      * DATA SET?
         BNO   READ
         TM    SMF64DTY,X'08'      RECORD LEVEL SHARED?
         BNO   READ
         USING SMFRCD64,R10        SMF RECORD MAPPING
         MVC   R_SYSID,SMF64SID    SYSID
         MVC   R_JOB,SMF64JBN      JOBNAME
         MVC   R_DSN,SMF64DNM      * COMPONENT NAME
MV_DATA  UNPK  DATE,SMF64RSD         JOB START DATE (JULIAN)
MV_TME   ICM   R3,B'1111',SMF64RST   TIME HUNDREDTHS OF SECONDS
         L     R1,F100               SINCE MIDNIGHT
         LA    R2,0
         DR    R2,R1              * SECONDS
         L     R1,F3600           * SECONDS IN  HOURS
         LA    R2,0
         DR    R2,R1              * HOUR EM R3
         CVD   R3,DOUBLE
         UNPK  HH,DOUBLE+6(2)
         OI    HH+1,X'F0'
         L     R1,F60
         LR    R3,R2               * REMAINING
         LA    R2,0
         DR    R2,R1               * MINUTS
         CVD   R3,DOUBLE
         UNPK  MM,DOUBLE+6(2)
         OI    MM+1,X'F0'
         CVD   R2,DOUBLE           * REMAINING SECOUNDS
         UNPK  SS,DOUBLE+6(2)
         OI    SS+1,X'F0'
*
         LA    R1,0
         ICM   R1,B'0011',SMF64ESL
         LA    R15,0(R1,R10)
         DROP  R10
         USING SMFRCD64,R15
         MVC   R_DDN,SMF64DDN      * DDNAME
         L     R3,SMF64DIN         * INSERTIONS
         CVD   R3,DOUBLE
         MVC   INSERT,MODEL
         ED    INSERT,DOUBLE+2
         L     R3,SMF64DDE         * RECORDS DELETED
         CVD   R3,DOUBLE
         MVC   DELETE,MODEL
         ED    DELETE,DOUBLE+2
         L     R3,SMF64DUP         * RECORDS UPDATED
         CVD   R3,DOUBLE
         MVC   UPDATE,MODEL
         ED    UPDATE,DOUBLE+2
         L     R3,SMF64DRE         * RECORDS RETRIEVED
         CVD   R3,DOUBLE
         MVC   READS,MODEL
         ED    READS,DOUBLE+2
         L     R3,SMF64DCS         * CONTROL INTERVAL SPLITS
         CVD   R3,DOUBLE
         MVC   CI_SPLIT,MODEL2
         ED    CI_SPLIT,DOUBLE+4
         L     R3,SMF64DAS         * CONTROL AREA SPLITS
         CVD   R3,DOUBLE
         MVC   CA_SPLIT,MODEL2
         ED    CA_SPLIT,DOUBLE+4
         L     R3,SMF64BMH         * HIT LOCAL SHARED BUFFER POOL
         CVD   R3,DOUBLE
         MVC   HITLBP,MODEL
         ED    HITLBP,DOUBLE+2
         L     R3,SMF64CFH         * HIT CF CACHE STRUCTURE
         CVD   R3,DOUBLE
         MVC   HITCF,MODEL
         ED    HITCF,DOUBLE+2
         L     R3,SMF64RIO         * DATA RECORDS FROM DASD
         CVD   R3,DOUBLE
         MVC   DASD,MODEL
         ED    DASD,DOUBLE+2
*
PUTREC   LTR   R8,R8              TITLE?
         BNZ   GR_REC
         LA    R8,1
         PUT   REC,TIT
         PUT   REC,REC_SPC
         PUT   REC,HEADER
         PUT   REC,REC_SPC
         LA    R15,0
         ST    R15,RC
GR_REC   PUT   REC,REC_JOB
         PUT   REC,REC_RLS
         PUT   REC,REC_IO
         PUT   REC,REC_SPC
         BCT   R9,READ
         B     SET_TIT
         DROP  R15
*
GOBACK   CLOSE (SMFDUMP,,REC)
         L     R13,SAVE+4
         L     R14,12(R13)
         L     R15,RC
         LM    R0,R12,20(R13)
         BR    R14
*
ABEND    LR   R7,R15
         WTO 'OPEN ERROR',ROUTCDE=11
         ABEND 1000,DUMP,STEP
*
*---------------------------------------------------------------------*
*  EXECUTE INSTRUCTIONS
*---------------------------------------------------------------------*
PACK     PACK  DOUBLE,5(0,R5)
*---------------------------------------------------------------------*
*  DCBS
*---------------------------------------------------------------------*
SMFDUMP  DCB   DSORG=PS,EODAD=GOBACK,MACRF=(GL),DDNAME=SMF
REC      DCB   DSORG=PS,MACRF=(PM),DDNAME=REPORT,LRECL=133
*---------------------------------------------------------------------*
```

```
*  WORK AREA
*-----------------------------------------------------------------------*
SAVE DC      18F'0'      LINKAGE CONVENTION AREA
SPANNED  DC    F'0'
DOUBLE   DS    D
HDUMP    DC    H'18'     SKIP DUMP AND TRAILER SMF RECORD
F3600    DC    F'3600'
F100     DC    F'100'
F60      DC    F'60'
RC       DC    F'4'
TIT      DC    134C' '
         ORG   TIT
         DC    C'1'
         ORG   TIT+10
         DC    C'VSAM RLS - CF STRUCTURE STATISTICS '
         DC    C'SINCE LAST OPEN'
         ORG   ,
HEADER   DC    134C' '
         ORG   HEADER+1
         DC    C'SYSID'
         DS    CL1
         DC    C'JOBNAME'
         DS    CL2
         DC    C'START DT/TIME'
         DS    CL2
         DC    C'DDNAME'
         DS    CL3
         DC    C'DATA SET NAME'
         ORG   ,
         DS    0F
REC_JOB  DS    0CL134
         DC    134C' '
         ORG   REC_JOB+1
R_SYSID  DS    CL4               1
         DS    CL2               5
R_JOB    DS    CL8               7
         DS    CL1               15
DATE     DS    CL5               16
         DS    CL1               21
HH       DS    CL2               22
         DC    C':'              24
MM       DS    CL2               25
         DC    C':'              27
SS       DS    CL2               28
         DS    CL1               30
R_DDN    DS    CL8               31
         DS    CL1               39
R_DSN    DS    CL44              36
         ORG   ,
REC_RLS  DC    134C' '
         ORG   REC_RLS+1
         DC    C'LOCAL BP HIT:'   1
HITLBP   DS    CL12             14
         DS    CL1              26
         DC    C'CF CACHE HIT:'  27
HITCF    DS    CL12             40
         DS    CL1              52
         DC    C'I/O DASD:'      53
DASD     DS    CL12             62
         ORG   ,
REC_IO   DS    OCL134
         DC    134C' '
         ORG   REC_IO+1
         DC    C'INS:'          23
INSERT   DS    CL12             27
         DS    CL1              39
         DC    C'DEL:'          40
DELETE   DS    CL12             44
         DS    CL1              56
         DC    C'UPDT:'         57
UPDATE   DS    CL12             62
         DS    CL1              74
         DC    C'READ:'         75
READS    DS    CL12             80
         DS    CL1              92
         DC    C'CI-SPLT:'      93
CI_SPLIT DS    CL6             101
         DS    CL1             107
         DC    C'CA-SPLT:'     108
CA_SPLIT DS    CL6             116
         ORG   ,
REC_SPC  DC 134C' '
MODEL    DC  X'4020202020202020202020202120'
MODEL2   DC  X'402020202120'
*----------------------------------------------------------------------*
*  DUMMIES E  EQUATES
*----------------------------------------------------------------------*
         LTORG
         YREGS
         DSECT
         IDASMF64
         END   SMFRLS
//*--------------------------------------------------------------------*
//L.SYSLMOD DD DISP=SHR,DSN=LOAD_LIBRARY
//L.SYSLIB  DD DISP=SHR,DSN=LOAD_LIBRARY
//L.SYSIN   DD  *
  ENTRY   SMFRLS
  NAME    SMFRLS(R)
/*
```

GTF procedure example

Example A-6 shows an example of a GTF procedure.

Example: A-6 GTF procedure example

```
//*--------------------------------------------------------------------*
//*  TRACE START: UNDER SDSF  ==>    /S GTFIO
//*  REPLY:
//*    NN AHL125A  RESPECIFY TRACE OPTIONS OR REPLY U
//*    TO USE OPTIONS         ==>     NN,U
//*    TO CHANGE OPTIONS ==> NN,TRACE=SSCHP,IOP,PCI,CCWP,JOBNAMEP
//*    ANSWER THE REPLY:
//*    NN AHL101A  SPECIFY TRACE EVENT KEYWORDS SSCHP IOP CCWP,JOBNAMEP
//*    WITH ==>
//*    NN,SSCH=(ADRS),CCW=(SI,CCWN=512,DATA=16,PCITAB=5),JOBNAME=JJJJ
//*          ADRS; DEVICES ADDRESS TILL 128: ADDI-ADDF
//*    NEW REPLY:  RR AHL102A  CONTINUE TRACE DEFINITION OR REPLY END
//*    ANSWER  ==> RR,END
//*    AGAIN REPLY  MSG AHL125A,       ---------> N,U
//*    TO STOP THE TRACE, COMMAND      --------->  /S NNNNN
//*    NNNNN IS THE STEPNAME SHOWN IN "DA" SDSF OPTION
//*=================================================================
//*  MEMBER MHL CONTAINS THE GTF TRACE OPTIONS:
//*  TRACE=SSCHP,IOP,PCI,CCWP,JOBNAMEP
//*=================================================================
//GTFIO   PROC MEMBER=GTFVSAM,M=MHL
//IEFPROC EXEC PGM=AHLGTF,PARM='MODE=EXT,DEBUG=NO,TIME=YES',
//    TIME=1440,REGION=6M
//IEFRDER DD   DSNAME=SYS1.&M..TRACE,UNIT=SYSDA,SPACE=(CYL,(150)),
//        DISP=(NEW,CATLG)
//SYSLIB  DD   DSNAME=MHLRES2.TESTS.JCL(&MEMBER),DISP=SHR
```

B

Miscellaneous performance items

This appendix addresses the lab environment that was used for the measurements throughout the book. It also provides a general discussion on caching, common error messages that indicate broken data sets, and output from the EXAMINE command.

This appendix contains the following sections:

- DASD cache concepts
- Symptoms and messages from a broken data set
- Symptoms and messages from a broken data set
- IDCAMS EXAMINE messages

DASD cache concepts

A cache is fast storage (no mechanical movement) in the DASD controller with two functions:

- Minimize access to disks by having hits
- To serve as a *speed matching buffer* to synchronize elements with different speeds such as channels and disks in a cache miss.

In this appendix, the word disk does not have the same meaning as DASD. Disk implies RAID media (FBA, SSA, or SCSI) used by modern controllers. DASD means the logical 3390/3380.

To have random hits (saving disk access) for reads and writes, the I/O workload must revisit its data. However, in certain cases this process does not happen. Such workloads are called *cache unfriendly*. Usually, there are two types of hits for VSAM data when the application revisits:

- The same logical record in a CI already in cache
- The same data CI (already in cache) record (a sort of lucky)

For sequential access, cache does not save data CI disks I/O operations. The cache just tries to match the speed of the disks and channels.

With the new controllers, a cache miss implies accessing the RAID disks, not the logical 3390/3380 device (which do not exist physically). Because the mapping between the 3390/3380 tracks to the FBA RAID disks is not externalized, the relative locations of files are not important any more. You no longer need to avoid long seeks or even RPS misses.

Heavy cache access reduces DASD skew (unequal 3390/3380 device utilization) because data formerly in heavily used devices now became electronically accessible because of the LRU algorithm. Refer to the DASD Activity report in Figure B-1.

```
 I=92%  DEV             ACTV RESP IOSQ ---DELAY--- PEND DISC CONN
VOLSER NUM   MX  LCU   RATE TIME TIME DPB CUB DB  TIME TIME TIME
TOTTSJ 256C      005A 0.024   11    0 0.0 0.0 0.0  0.2  8.2  2.2
TOTJS1 250C      0059 5.764    6    0 0.0 0.0 1.2  1.5  0.1  4.7
TSMS50 650D   4  0144 139.6    1    0 0.0 0.0 0.0  0.3  0.0  0.7
```

Figure B-1 DASD Activity Report

Increasing the I/O rate, decreases the disconnect time (the less the disconnect, the more hits in cache). This decrease is because the LRU algorithm is keeping the most used elements in cache.

There are two types of cache: Volatile and non-volatile (NVS). NVS is used to keep DFW, dual copy, and XRC remote copy records. In this chapter, the word cache is used to mean the volatile cache, and NVS for the non-volatile.

The cache is made of CMOS DRAM storage for Data and Directory. The directory entries describe the data elements (4 KB in ESS) and are ordered in the following queues:

- LRU (pointing to active data elements)
- Free (available for staging from disk or writes)
- Pinned (changed data in cache and the respective disk is not available)
- Defective

Destage is the movement from cache to disk caused by previous DFW and CFW writes. It is asynchronous with the I/O operation that ran the writes. Remember that demotion is not a synonym of destaging. Demotion means to take a data element out from cache. If there is a valid copy in disk, there is not a destage. If not, a destage is run. There are three types of destaging:

- To relieve contention when NVS or Cache become full. This process is controlled by modified LRU algorithms, and occurs when the percentage of NVS occupancy is greater than X%. If greater than Y% (Y>X), the DFW I/O operation bypasses the NVS cache. It goes synchronously from volatile cache to disk (called a DFW bypass)
- Done in background by the controller when the percentage of NVS occupancy is above another threshold, Z% (Z < X)
- Forced by Z EOD command or by a hardware error

The amount of data that are destaged is usually more than one 3390/3380 track. A controller can Identify other records that have changed in adjacent tracks of the record to be destaged, and destage all of them. This process saves rotational delays in the disks, and bypasses the RAID write penalty.

Stage is the movement from DASD to cache. It can be synchronous (a read miss) for a random request, or asynchronous for a sequential look ahead read. Pay attention that random and direct have the same meaning.

A cache hit might overlap with staging/destaging operations for the same 3390/3380 logical device. You can have concurrent access to same 3390/3380 device data but not the same track. For example, you might have one hit in the cache and an asynchronous destage in the disks back storage (not in the same 3390/3380 track). This is not the ESS PAV feature because there is just one active I/O operation. That operation is the one with the hits in the cache. The destage is just controller housekeeping.

The controller records data about number of I/O operations, hits, misses, and destages, by volume or on a data set basis. These values are shown by IDCAMS LIISTDATA command and the RMF Cache report. They are also used in SMS for Dynamic Caching for data sets.

Set Subsystem CCW activates the use of caching in the controller (subsystem) and in individual volumes. This CCW allows cache modes as normal, DFW, and CFW. All these modes can be asked explicitly by software through the Define Extent CCW (per I/O operation). In some cases, the controller can adaptively change the mode to suit the observed pattern of access. Following is the description of such a CCW.

Define Extent CCW provides a 16-byte parameter that defines the limit on subsequent operations (extent information, avoidance of writes, for example), provides a blocksize value, and specifies caching control mode (or hints) for the channel program. The caching mode has the granularity of I/O operation. These modes are not mutually exclusive, and the same I/O operation might have more than one mode. The modes are:

- Track Level Cache
- Record Level Cache (RLC)
- Sequential:
  - Reads:
    - Sequential Access
    - Sequential Pre-stage
  - Writes
- Least recently used (LRU)
- Normal Caching
- CFW
- DFW, which has the possibility of Quick Writes

- Bypass Cache
- Inhibit Cache Loading

Cache Modes

These modes have the following characteristics:

- Track Level Cache (TLC)

  In TLC, the unit of transport between cache and disks are the 3390/3380 tracks. TLC is best for sequential access. When in TLC mode, the 3390/3380 track is staged to cache in one pass if the first miss is in record zero (R0). It is staged in two passes otherwise.

- Record Level Cache (RLC)

  In RLC, the unit is the referred logical 3390/3380 physical record. RLC is best for random (also called direct) processing.

  In “logical 3390/3380 physical record”, the word *logical* indicates that the 3390/3380 does not actually exist. The word *physical* indicates that you are talking about the physical record (the block) in the logical 3390/3380 track.

  RLC improves performance for applications that do not exhibit good locality of reference and where the cost of track caching outweighs the benefits. RLC reduces the costs that are associated with cache miss I/Os by staging less data into cache (less cache pollution). Doing so frees the volume and allows other activity to be processed more quickly. Reading less data into cache also enables data to stay in cache longer, which increases the chances of future cache hits.

  RLC is mutually exclusive from TLC.

  RLC can be activated for VSAM SMS-managed may-cache data sets. SMS, through DCME, decides in Define Extent when to use RLC for reads. For more information, see “Using cache in an SMS data set” on page 414. In addition to deciding whether to enable or to disable the cache for maybe-cache SMS data sets, SMS selects either RLC or TLC for reads.

- Sequential

  When in sequential mode, the controller uses the cache as a speed matching buffer.

  In *sequential read mode*, the controller does the pre-staging of some future referenced logical 3390/3380 tracks. After they are used, the tracks might be demoted from the cache depending on the sub mode:

  – *Sequential Access*: The used data is a strong candidate to be demoted

  – *Sequential Pre-stage*: The used data is protected by the LRU algorithm.

  Sequential read can be activated by using these methods:

  – Explicitly by software through Define Extent CCW (also called sequential hint), as declared by VSAM ESDS for Sequential Access. KSDS/VRRDS in certain conditions declare Sequential Pre-stage.

  – By sequential detect, where the controller detects sequential access (six sequential referred logical 3390/3380 physical tracks in the ESS).

    KSDS/VRRDS VSAM organizations in certain conditions do not use the Define Extent. If the Define Extent is not used, CI/CA splits can be avoided in data sets usually processed sequentially. Splits make the logical sequence different from the physical sequence, and the controllers detect only the physical sequence pattern.

- Least Recently Used (LRU)

  LRU is not actually a mode, but rather is a technique to maintain in the cache the most referenced elements. LRU is the main algorithm that controls cache demoting. It does so based on how often an element was referenced in the past, so that it is easily accessible in the future. LRU is automatically set off when sequential caching, inhibit cache load, and bypass cache modes are active

- Normal Caching

  In this mode, the NVS cache is not used. There are four cases to consider in this mode:

  – For a read hit, there is a data transfer from cache to channel followed by a channel end (CE)/ device end (DE) I/O interrupt. The directory entry is the LRU update, which means that the data is to be kept in cache for a while.

  – For a read miss, the channel is disconnected, and the disk is accessed to stage data to cache. There might be a delay caused by the disk being already busy or all lower interfaces being busy. After this process is complete, the channel is reconnected.

    If all the serving channels are busy, there is not an RPS miss because the data is already in the cache. With the new controllers, RPS misses occur only when the internal paths to disks are busy, and the disk needs a new revolution.

  – For a write hit, because the NVS cache is not used, it is similar to a miss. The channel moves data to volatile cache and disconnects. The disk is accessed synchronously to write data. There might be a delay that is caused by the disk being already busy or all lower interfaces being busy.

    The word *synchronously* means that the write to disk is done without the end of the I/O operation being posted to the application.

    After the write to disk, the channel is reconnected and CE/DE I/O interrupt is presented. This is an LRU update, so the data are kept in cache for a while.

  – For a write miss, the channel moves data to cache and disconnects. The disk is accessed synchronously to write data. There might be a delay that is caused by the disk being already busy or all lower interfaces being busy.

    The word *synchronously* means that the write to disk is done without the end of the I/O operation being posted to the application.

    After the write to disk, the channel is reconnected and CE/DE I/O interrupt is presented. There is no LRU update, so the cache copy of the data is ready to be demoted.

- Cache Fast Write (CFW)

  CFW is used for temporary data sets, and so does not use the NVS for writes. It must be allowed by the Set Subsystem CCW issued by IDCAMS.

  – For reads and write misses, it is identical to normal caching.

  – For a write hit, the data is transfer from the channel to the cache followed by a CE/DE I/O interrupt. The directory entry is then LRU updated, so the data is kept in cache for a while. Later, the data is asynchronously destaged to disks.

  This mode is used by Sort for temporary files and for creating PDSE members (before the Stow) when the Hiperspace is full. The exploiter must declare CFW in the Define Extent CCW.

- DASD Fast Write (DFW)

  DFW allows the use of the NVS cache for writes. It avoids accessing disks for a write hit. It must be allowed by the Set Subsystem CCW issued by IDCAMS.

  – For reads, this mode is identical to Normal Caching.
  – For a write hit, the data is transfer from the channel to the cache and NVS, followed by a CE/DE I/O interrupt. The directory entry is then LRU updated, so the data is kept in cache for a while. Some modern controllers such as ESS send the CE/DE earlier. One copy of the data in the NVS (other copy in the channel adapter buffer) does the copy to the volatile cache immediately after the CE/DE.

    Later on and asynchronously, the data is destaged to disks. However, if the NVS cache is under stress, the controller sends the data from volatile cache directly to disks (synchronously). This situation is called a DFW bypass. The channel stays disconnected during this data transfer.

- Almost 100% DFW Hits (Quick writes)

  This mode allows the use of the NVS cache for writes. It avoids accessing disks for a write hit and almost 100% of the write misses. In certain controllers, this logic is included in the RLC Licensed Internal Code (LIC).

  Almost 100% DFW hits is also called “quick writes”. It allows a DFW miss turn to a hit if adequate NVS space is available.

  The reason for the access of disks for a write data miss in a DFW mode is the verification of the record length. Figure B-2 on page 413 shows an illustration of the two types of write CCWs:

  – *Write format*, also called write count-key-data, formats the 3390/3380 track by overlaying the old records in the track. It creates a count with the respective data (usually with a zero content), and the rest of the track is erased. The length of the data record is included in the write count-key-data CCW and copied in the count. In this case, the previous record data length is not important because it is overlaid.

    All the write format I/O operations are considered a write hit.

  – *Write modified*, also called write data, changes the contents of the data portion of a pre-formatted record, the length of which is already described in the count. The length of the data record is also included in the write data CCW. Any mismatch between this length and the one already specified in the count of the formatted record is posted by the channel to the application. In some cases, this process might stop the execution of the channel program. You must access the 3390/3380 track in a DFW write modified miss because the controller is not able to verify the length in the write modified CCW and the count.

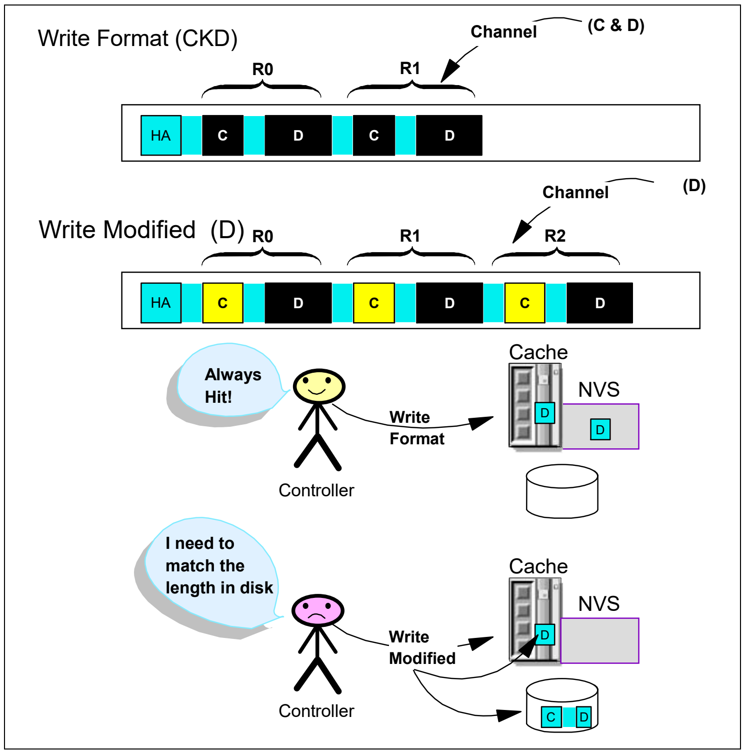


Figure B-2 Types of writes

For quick writes, the controller must know or be able to predict the length of the record, avoiding access to the disks.

Quick writes are implemented in two submodes that are associated with record level cache. Remember that in some IBM controllers, quick writes are associated with the record level cache LIC:

– Record Level Cache I: This submode is a joint VSAM/controller implementation.

  Because VSAM is a trusted partner (regular data format), all writes become quick writes. That is, the controller does not go to disk to verify the data record length. This function benefits IMS, DB2, and CICS users of VSAM. The data set can be SMS or not.

– Record Level Cache II: This submode is non-adaptive (the controller does it by itself), with no Define Extent CCW software intervention. When RLC II is activated, it cannot be disabled. In the first access to the record, there is a disk access (miss) and the record length is moved and kept in cache for the future requests. It aims to reach almost 100% writes hits. The controller automatically determines whether to use the track cache algorithm as it processes I/Os to the volumes with cache active. RLC II has these prerequisites:

  - Volume behind a controller with RLC II installed
  - TLC enabled for that volume
  - DFW enabled for that volume

- Inhibit Cache Load (ICL)

  If a read hit occurs, read from cache and LRU update. If it is a read miss or a write access the disks through volatile cache, there is no LRU update. Cache space is not allocated for any new tracks from DASD. Write operations that do not require access to DASD are not inhibited by this setting.

Used by DFDSS, DFSORT, and SMS (never-cache and some of the maybe-cache data sets). Generally, it can be used by an application that knows that the record to be requested is not going to be used soon (as for cache unfriendly accesses).

Ignore ICL is an option that is set in VPD when your workload is not aware of a huge cache size.

- Bypass Cache

  The I/O request must be run in the disks, even if the data is in the cache. However, the data always passes through cache without LRU update, so the copy is ready to be demoted. If there is a hit in the cache, the LRU is updated. Cache images of tracks that are modified on the device are invalidated. Tracks modified in cache but not on DASD are destaged to DASD before access is allowed. For Duplex volumes, this process includes destaging to both devices. DASD Fast Write operation is disabled when this attribute is active.

  This mode is used by ICKDSF, paging, and Write Check access method options.

  Ignore BYP is an option that is set in VPD. It is used when your workload is not aware of a huge cache size.

Using cache in an SMS data set

SMS uses the define extent CCW to set some cache modes during the I/O operation for an SMS data set. If there is a conflict between Define Extent and IDCAMS volume setting, the more restrictive option prevails. For example, if IDCAMS says non-DFW for the volume and define extent says DFW for the I/O operation, it will be non-DFW.

However, there are certain cache modes that are not set by SMS as bypass cache and CFW.

Cache usage attributes

An opened SMS data set can have one of three cache usage attributes, in regard to DASD cache:

- Must-cache data set: Uses cache/NVS for the I/O operations.
- Never-cache data set: Does not use cache except for buffering, and does not use NVS. The ICL mode is requested in Define Extent CCW.
- May-cache data set: Uses cache/NVS depending on the cache/NVS constraints.

The same data set can have one of these attributes for sequential accessing mode, and a different one for direct accessing mode. This is also true for read access and write access. These attributes are based in the SMS storage class installation parameters (MSR for Direct, MSR for sequential or BIAS). MSR is the wanted I/O service time in milliseconds, and BIAS is the expected preference between reads or writes.

The cache usage attributes are assumed at open time for a data set. These cache attributes are kept constant until the data set is closed.

Dynamic Cache Management Enhanced (DCME)

DCME is a function in the controller that produces cache information in a system on a data set basis. SMS uses data from DCME to adjust the use of the cache for may-cache data sets. With DCME, SMS distinguishes between good (cache-friendly) and poor (cache unfriendly) cache candidate data sets when it decides which I/Os to which data sets should be cached.

DCME produces two key measurements:

- Data set cache behavior

  DCME maintains information about the hit ratios that are achieved by I/Os to each may-cache data set. DCME continuously updates this information so that it can decide which I/Os to which data sets should be cached, based on the most recent I/O activity.

  When considering whether to cache I/Os to a data set, two controller resources must be accounted for: The cache and the NVS. A data set might well be a good user of the cache and a poor user of the NVS. SMS, therefore, maintains two criteria: One general (for all I/Os) and one specifically for writes.

  Two data set related indicators are calculated:

  – Overall Hit Ratio: Reads Hits + Writes Hits / Cacheable IOs

    Used to decide to cache a Read request.

  – Write Hit Ratio: Write Hits / Write IOs

    Used to decide to cache a Write request.

  A hit is perceived by a disconnect time less that 0.5 ms.

  These indicators are weighed averages of previous values to avoid sudden changes.

- Subsystem load

  In which the subsystem is the DASD controller.

The DCME in the controller produces global data about the cache usage.

A Subsystem Threshold indicator is calculated by SMS from the global DCME data. The higher the Subsystem Threshold, the more cache contention there is. These statistics are periodically collected by SMS. The time is controlled by DINTERVAL at IGDSMSxx Parmlib, with a default of 150 seconds.

Subsystem Threshold reflects the DASD cache performance. It is composed of the Read Hit Ratios and DFW bypass for all the cached volumes in the DASD subsystem.

The DFW bypass occurs when a DFW hit request requires NVS, but this storage is not available because of contention. In this case, the I/O request bypasses the NVS and is run directly from the volatile cache to the disks.

Also, based on the statistics, SMS maintains two global average indicators:

- Cache Control Indicator, the percent of may-cache I/O requests allowed to use cache.
- NVS Control Indicator, the percent of may-cache DFW I/O requests allowed to use NVS.

These values that are used for this calculation can be displayed by using the D SMS,CACHE command as shown in Figure B-3.

```
IGD002I 18:09:11 DISPLAY SMS 276
 SSID    DEVS     READ    WRITE    HIT RATIO    FW BYPASSES
 00FF      5      N/A     N/A         97%           0
 8900      8      N/A     N/A         98%           0
 8902      5      N/A     N/A         98%           0
 8904      4      N/A     N/A         99%           0
 8903      7      N/A     N/A         99%           0
 8901      4      N/A     N/A         98%           0
 000A      8      N/A     N/A         99%           0
 3000     21      N/A     N/A         99%           0
 8905      6      N/A     N/A         97%           0
 6004      8      N/A     N/A         90%           0
 0028      4      N/A     N/A         98%          24
 00FD      7      N/A     N/A         98%           0
 00FC      4      N/A     N/A         99%           0
 00FE      1      N/A     N/A         98%           0
```

Figure B-3 D SMS,CACHE output

SSID	Subsystem Identifier
DEVS	Number of managed devices that are attached to the subsystem
READ	Percent of data on managed devices eligible for caching
WRITE	Percent of data on managed devices eligible for DFW
HIT RATIO	Percent of reads with cache hits
FW BYPASSES	Number of fast write bypasses because of NVS overload

An I/O operation for a may-cache data set has three states:

- Normal: The data set I/O is allowed to use the cache through the define extent CCW.
- Inhibit: The data set I/O does not use the cache. That is, the Inhibit Cache Load bit is set on the define extent first CCW of a Read channel program to inhibit the staging of the cache. For a Write channel program, the Inhibit DFW bit is set on the define extent first CCW of a Write channel program to inhibit DFW.
- Force: The data set I/O is cached so that the indicators can be evaluated.

The system has the following logic:

- After open for the first 100 I/Os and the first 100 Writes, the I/Os are *forced*.
- The data set indicator is compared with subsystem threshold. If does not exceed that threshold, the data set I/Os are inhibited for the next 5000 I/Os. If it exceeds the threshold, the data set I/Os are going to be normal for a certain amount of time. The comparison is done again after this period.

If the Subsystem Threshold indicator is going up, the exclusion from cache for may-cache data sets is gradual. The DASD subsystem performance does not experience any sudden and dramatic changes.

As a DCME by-product, DFSM I/O statistics are collected in new SMF records in data set and storage class basis. These statistics include I/O rates, I/O response time, I/O service time components, and caching statistics for reads and writes.

Data sets to avoid caching

There are some data sets that are not good candidates for DASD caching:

- Data sets that are processed track-by-track, as the input to the memory dump function of DFDSS itself. However, in this case the installation does not need to care about this, because DFDSS uses the adequate option (inhibit cache load) in the Define Extent command.
- Data sets that have a poor direct revisit pattern.
- ASM paging data sets.

Symptoms and messages from a broken data set

The following are most common messages that are associated with broken data sets events:

- IDC3302I ACTION ERROR ON dsname

  An error was detected while attempting to access the data set. See the associated message in the program listing for explanation.

- IDC3308I ** DUPLICATE RECORD xxx

  The output data set of a Repro command already contains a record with the same key or record number. In the message text, xxx for an indexed data set, the first five bytes of the duplicate key, in hexadecimal format. For a relative record data set, the relative record number (in decimal) of the duplicate record.

  The system does not write the record. The system continues processing with the next record unless this is a copy catalog and a duplicate record is encountered or there has been a total of four errors. The system ends in either case. For example, if a duplicate record is encountered while Repro is copying a catalog, the system ends processing.

  If the record in the input file with the duplicated key is to replace the one in the data set, specify the replace option. If not, check your Repro input.

- IDC3314I RECORD xxx OUT OF SEQUENCE

  The key of the record to be written is less than or equal to the key of the last record written. In the message text, xxx is the first five bytes in hexadecimal format of the key of the record that is out of sequence.

  If the output data set is a Virtual Storage Access Method (VSAM) data set, the system ends processing of the command after four errors.

  Rearrange the records to be written so that they are in ascending key sequence. The record can be written to the data set by using skip sequential processing. Run the job again. The output data set then opens for skip sequential processing because the data exists in the data set. Records that were out of sequence are then written.

- IDC3351I ** VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code {RPLFDBWD=nnnnnnnn}

  An error was encountered during VSAM open, close, or action request processing, as indicated in the text of the message:

  nnnnnnnn The meaning can be found in DFSMS/MVS Macro Instructions for Data Sets.

The return code can be any of the following codes:

- For CLOSE errors, only the return codes that are associated with broken data sets are included:

128	Index search horizontal chain pointer loop encountered.
136	Not enough virtual storage was available in the address space of the program for a work area for CLOSE.
184	An uncorrectable I/O error occurred while VSAM was completing outstanding I/O requests.
246	The compression management services (CMS) close function failed.

- For OPEN errors, only the return codes that are associated with broken data sets are included:

76	Attention message: The interrupt recognition flag (IRF) was detected for a data set that was opened for input processing This message 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.
88	A previous extend error occurred during EOV processing of the data set.
96	Attention message: An unusable data set was opened for input.
104	Attention message: The time stamp of the volume on which a data set is stored does not match the system time stamp in the volume record in the catalog. This mismatch indicates that extent information in the catalog record might not agree with the extents indicated in the VTOC of the volume.
108	Attention message: The time stamps of a data component and an index component do not match. This mismatch indicates that either the data or the index has been updated separately from the other. Check for possible duplicate VVRs.
116	Attention message: The data set was not properly closed or was not opened. If the data set was not properly closed, data might be lost if processing continues. Use the access method services VERIFY command to attempt to close the data set properly. In a cross-system shared DASD environment, a return code of 116 can mean that The data set was not properly closed. Also, it can mean that the data set is opened for output on another processor.

Tip: If you use the VERIFY command, this message can be displayed again when VERIFY processing opens the data set. If VERIFY processing then successfully closes the data set, VERIFY processing issues condition code 0 at the end of its processing. In addition, an empty cluster cannot be verified.

132	Either there was not enough storage was available for work areas, or the format-1 DSCB or the catalog cluster record is incorrect.
136	Not enough virtual-storage space is available in the program's address space for work areas, control blocks, or buffers.
140	The catalog indicates that this data set has an incorrect physical record size.

160	The operands that are specified in the ACB or GENCB macro are inconsistent with each other or with the information in the catalog record. This error can also occur when the VSAM cluster is opened when it is empty.
164	An uncorrectable I/O error occurred while VSAM was reading the volume label.
168	The data set is not available for the type of processing specified. Or an attempt was made to open a reusable data set with the reset option while another user had the data set open.
184	An uncorrectable I/O error occurred while VSAM was completing an I/O request.
190	An incorrect high-allocated RBA was found in the catalog entry for this data set. The catalog entry is bad and must be restored.
192	An unusable data set was opened for output.
193	The IRF was detected for a data set that was opened for output processing.
194	Direct access of a compressed data component is not allowed.
200	The volume is unusable.
212	The ACB MACRF specification is GSR or LSR, and the data set requires create processing.
232	Reset (ACB MACRF=RST) was specified for a nonreusable data set, and the data set is not empty.
240	Format-4 DSCB and catalog time stamp verification that is failed during volume mount processing for output processing.

– For a Logical I/O Error

4	End of data set encountered (during 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 and it returned to VSAM and the processing program issued another GET.
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	This code can indicate the following error conditions: - You attempted to store a record out of ascending key sequence in skip-sequential mode - The record had a duplicate key - For skip-sequential processing, your GET, PUT, and POINT requests - are not referencing records in ascending sequence - For skip-sequential retrieval, the key requested is lower than the previous key requested. - For shared resources, the buffer pool is full.
16	Record not found.
20	The record is already held in exclusive control by another requester.
28	Data set cannot be extended because VSAM cannot allocate more direct-access storage space. There is not enough space left to make the secondary allocation request, you attempted to increase the size of a data set while processing with SHROPT=4 and DISP=SHR. Or the index CI is not large enough to hold the entire CA. This error might also

be because of a data set trying to extend beyond 4 GB on a system that does not support extended addressability.

32	An RBA was specified that does not give the address of any data record in the data set.
40	There is insufficient virtual storage in the user's address space to complete the request.
116	During initial data set loading (when records are being stored in the data set for the first time), GET, POINT, ERASE, direct PUT, and skip-sequential PUT with OPTCD=UPD are not allowed. During initial data set loading, VERIFY is not allowed except for an entry-sequenced data set (ESDS) defined with the RECOVERY option. For initial loading of a relative record data set, the request was other than a PUT insert.
128	A loop exists in the index horizontal pointer chain during index search processing.
144	Incorrect pointer (no associated base record) in an alternate index.
156	An addressed GET UPD request failed because the control interval flag was on, or an incorrect control interval was detected during keyed processing. In the latter case, the control interval is incorrect for one of the following reasons: - A key is not greater than the previous key. - A key is not in the current control interval. - A spanned record RDF is present. - A free space pointer is incorrect. - The number of records does not match a group RDF record count. - A record definition field is incorrect. - An index CI format is incorrect.
212	Unable to split index. Increase index CI size.
236	Validity check error for SHAREOPTIONS 3 or 4.
245	A severe error was detected by CMS during compression processing.
246	A severe error was detected by CMS during decompression processing.
250	A valid dictionary token does not exist for the compressed data set. The data record cannot be extracted.
254	I/O activity on the data set was not quiesced before the data set was closed.

- IDC3350I synad[SYNAD]message[from VSAM]

  An I/O error occurred for a VSAM data set. The message text, format, and explanation of VSAM I/O errors are provided in DFSMS/MVS Macro Instructions for Data Sets.

- IEC070I rc[(sfi)]- ccc,jjj,sss,ddname, dev,ser,xxx,dsname,cat

  An error occurred during EOV (end-of-volume) processing for a VSAM data set. The message text includes these components:

rc	Reason code. This field indicates the reason for the error. The reason codes, their meanings, and the corresponding system action and required responses are listed under message IEC161I.
sfi	Subfunction information, which is error information that is returned by another subsystem or component. This field is displayed only for

certain return codes, and its format is shown with those codes to which it applies.

ccc The Problem Determination (PDF) Function code. The PDF code is for use by IBM if further problem determination is required. If the PDF code has meaning for the user, it is documented with the corresponding Reason Code (rc).

```
IEC070I   RC32 , RC202 , RC8 , RC18 , RC24 , RC104 , RC203 MSGIEA000I IOS000I CMD REJ,
COMMAND REJECT

ADR970E   HSM  MISSING CI within SEQUENCE SET TRACK TRACKS TRK TRKS TRKS=0 TRACKS=0
EXTENT
```

IDCAMS EXAMINE messages

The following are the most frequent error messages that are issued by EXAMINE command:

```
IDC01714I  ERROR LOCATED at OFFSET xxx

IDC01720I  INDEX CONTROL INTERVAL DISPLAY at RBA xxx FOLLOWS

IDC11703I  DUPLICATE KEYS in INDEX

IDC11704I  INDEX KEYS are NOT in SEQUENCE

IDC11705I  INDEX RECORD CONTAINS DUPLICATE INDEX POINTERS

IDC11707I  DUPLICATE INDEX POINTERS FOUND in SEQUENCE SET

IDC11711I  INDEX CONTROL INTERVAL COUNT ERROR

IDC11715I INDEX HIGH-USED RBA is NOT a MULTIPLE of CI SIZE IDC11724I  DATA COMPONENT CA
NOT  KNOWN to SEQUENCE SET IDC11725I  SEQUENCE SET RBA INCONSISTENT with VSAM- MAINTAINED
RBA

IDC11727I  INDEX HIGH-USED RBA GREATER THAN HIGH-ALLOCATED

IDC11728I  DATA FOUND in EMPTY CI

IDC11733I  DATA COMPONENT KEY SEQUENCE ERROR MSGIDC11758I  SOFTWARE EOF FOUND in INDEX CI

IDC11763I  RBA of INDEX CI GREATER THAN HIGH-USED RBA IDC11771I  INVALID RBA GENERATED

IDC11772I  HORIZONTAL POINTER CHAIN LOOP
```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

For information about ordering these publications, see “How to get Redbooks” on page 424. Note that some of the documents referenced here might be available in softcopy only.

- *CICS and VSAM Record Level Sharing: Implementation Guide*, SG24-4766
- *CICS and VSAM Record Level Sharing: Planning Guide*, SG24-4765
- *CICS and VSAM Record Lavel Sharing: Recovery Considerations*, SG24-4768
- *Transactional VSAM Application Migration Guide*, SG24-6145
- *z/OS V1.12 DFSMS Technical Update*, SG24-7895

Other publications

These publications are also relevant as further information sources:

- *CICS VSAM Recovery Implementation Guide*, SH26-4126
- *CICS Recovery and Restart Guide*, SC33-1698
- *DFSMStvs Administration Guide*, GC26-7483
- *z/OS DFSMS Managing Catalogs*, SC26-7409
- *z/OS DFSMS Using Data Sets*, SC26-7410
- *z/OS DFSMS Access Method Services for Catalogs*, SC26-7394
- *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402
- *z/OS DFSMStvs Planning and Operating Guide*, SC26-7348
- *z/OS MVS Diagnosis: Tools and Service Aids*, GA22-7589
- *z/OS MVS Programming: Authorized Assembler Services*, SA22-7610
- *z/OS MVS Programming: MVS Assembler Services Guide*, SA22-7605
- *z/OS MVS Programming: Resource Recovery*, SA22-7616
- *z/OS MVS Setting Up a Sysplex*, SA22-7625
- *z/OS MVS System Management Facilities (SMF),* SA22-7630

Online resources

These websites and URLs are also relevant as further information sources:

- Flashes

  http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/Flashes
- IBM Coupling Facility sizer

  http://www-1.ibm.com/servers/eserver/zseries/cfsizer/vsamrls.html
- TRSMAIN Utility

  http://techsupport.services.ibm.com/390/trsmain.html
- z/OS RMF

  http://www.ibm.com/servers/eserver/zseries/rmf

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this website:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

VSAM Demystified

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

VSAM Demystified

Learn the latest VSAM functions and manage VSAM data

Understand, evaluate, and use VSAM properly

Learn problem determination and recommendations

Virtual Storage Access Method (VSAM) is one of the access methods used to process data. Many of us have used VSAM and work with VSAM data sets daily, but exactly how it works and why we use it instead of another access method is a mystery.

This book helps to demystify VSAM and gives you the information necessary to understand, evaluate, and use VSAM properly. It clarifies VSAM functions for application programmers who work with VSAM. This book also builds upon the subject of Record Level Sharing and DFSMStvs.

The practical, straightforward approach should dispel much of the complexity associated with VSAM. Wherever possible an example is used to reinforce a description of a VSAM function.

This IBM Redbooks publication is intended as a supplement to existing product manuals. It is intended to be used as an initial point of reference for VSAM functions.

SG24-6105-02 ISBN 0738437433