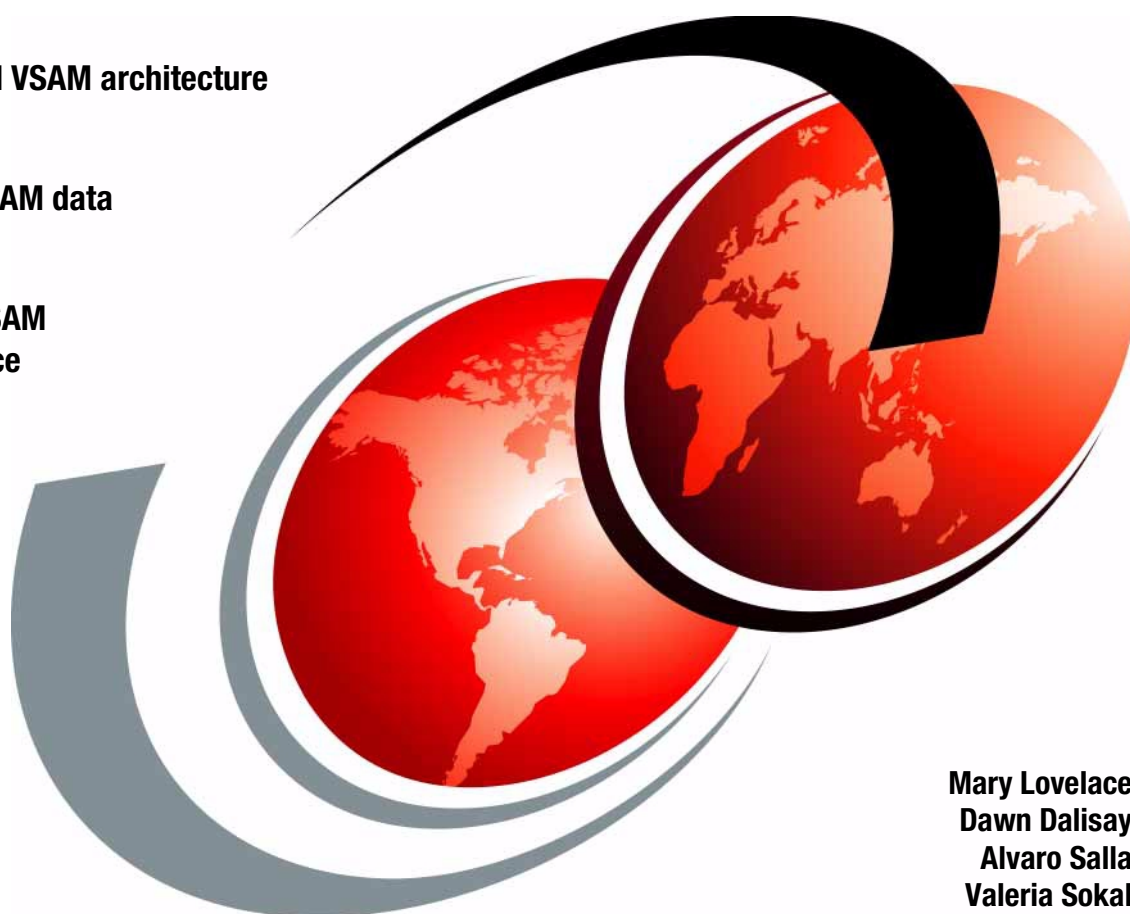


# VSAM Demystified

Understand VSAM architecture

Manage VSAM data

Improve VSAM  
performance



Mary Lovelace  
Dawn Dalisay  
Alvaro Salla  
Valeria Sokal

[ibm.com/redbooks](http://ibm.com/redbooks)

**Redbooks**





International Technical Support Organization

## **VSAM Demystified**

January 2001

**Take Note!**

Before using this information and the product it supports, be sure to read the general information in Appendix C, "Special notices" on page 253.

**First Edition (January 2001)**

This edition applies to DFSMS/MVS Version 1, Release Number 5, Program Number 5695-DF1 for use with the OS/390 Operating System.

Comments may be addressed to:  
IBM Corporation, International Technical Support Organization  
Dept. QXXE Building 80-E2  
650 Harry Road  
San Jose, California 95120-6099

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

© Copyright International Business Machines Corporation 2001. All rights reserved.

Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

---

## Contents

<b>Figures</b> .....	ix
<b>Tables</b> .....	xi
<b>Preface</b> .....	xiii
The team that wrote this redbook .....	xiii
Comments welcome .....	xiv
<b>Chapter 1. VSAM basics</b> .....	1
1.1 VSAM overview .....	1
1.2 What is VSAM? .....	1
1.3 VSAM terminology .....	2
1.3.1 Logical record .....	2
1.3.2 Physical record .....	2
1.3.3 Control interval .....	3
1.3.4 Control area .....	4
1.3.5 Splits .....	4
1.3.6 Spanned records .....	5
1.3.7 Relative byte address .....	5
1.3.8 Component .....	6
1.3.9 Cluster .....	8
1.3.10 Keys .....	9
1.3.11 Sphere .....	10
1.3.12 Alternate indexes .....	10
1.3.13 Alternate index paths .....	11
1.4 Data set types .....	11
1.4.1 Entry sequenced data set (ESDS) .....	11
1.4.2 Keyed sequenced data set (KSDS) .....	13
1.4.3 Relative record data set (RRDS) .....	14
1.4.4 Variable relative record data set (VRRDS) .....	16
1.4.5 Linear data set (LDS) .....	16
1.5 Extended format data set .....	18
1.6 Extended addressability (EA) .....	19
1.7 Comparing VSAM data set organizations .....	23
1.8 A brief history of VSAM .....	24
1.9 Choosing a VSAM data set type .....	25
1.10 Accessing VSAM data .....	27
1.10.1 IDCAMS .....	27
1.10.2 Accessing HFS files through VSAM .....	29
1.10.3 DITTO/ESA .....	29
1.11 Defining VSAM data sets .....	33

1.11.1 Using IDCAMS . . . . .	33
1.11.2 System-managed data sets . . . . .	34
1.11.3 Parameters of interest . . . . .	34
<b>Chapter 2. Performance . . . . .</b>	<b>37</b>
2.1 Service level agreement (SLA) . . . . .	37
2.2 Transaction performance . . . . .	38
2.3 Performance management . . . . .	39
2.3.1 I/O performance . . . . .	40
2.4 VSAM performance management . . . . .	40
2.5 VSAM rule-of-thumb (ROT) mode . . . . .	41
2.5.1 Invalid rules-of-thumb (IROTs) . . . . .	41
2.6 Parameters affecting performance . . . . .	42
2.6.1 Allocation units . . . . .	42
2.6.2 Buffer space . . . . .	48
2.6.3 Control interval size . . . . .	48
2.6.4 Free space . . . . .	50
2.6.5 Index options . . . . .	52
2.6.6 Share options . . . . .	53
2.6.7 Initial load option . . . . .	54
2.6.8 Region size . . . . .	55
2.6.9 Buffering options . . . . .	58
2.6.10 Data compression . . . . .	84
2.6.11 Data striping . . . . .	92
2.7 VSAM performance management . . . . .	103
2.7.1 Performance scenario using RMF reports . . . . .	103
2.7.2 Reduce the number of I/Os . . . . .	107
2.7.3 I/O wait time (IOSQ) for VSAM files . . . . .	111
2.7.4 I/O wait time (PEND) for VSAM files . . . . .	112
2.7.5 I/O service time (disconnect) for VSAM files . . . . .	112
2.7.6 I/O service time (connect) for VSAM files . . . . .	117
2.7.7 How to decrease VSAM CPU time . . . . .	119
2.8 VSAM and SmartBatch . . . . .	121
2.8.1 SmartBatch highlights . . . . .	121
2.8.2 SmartBatch components and VSAM . . . . .	122
<b>Chapter 3. Recovery of VSAM data sets . . . . .</b>	<b>127</b>
3.1 Basic recommendations . . . . .	127
3.2 VSAM recovery information sources . . . . .	128
3.3 How to back up VSAM data sets . . . . .	128
3.3.1 IDCAMS EXPORT and IMPORT . . . . .	128
3.3.2 Backup-while-open concepts . . . . .	130
3.4 Space Constraint Relief parameter (fewer X'037' abends) . . . . .	131

3.5 IDCAMS recovery commands . . . . .	133
3.5.1 EXAMINE command . . . . .	134
3.5.2 DIAGNOSE command . . . . .	134
3.5.3 VERIFY command . . . . .	135
3.6 Useful documents . . . . .	137
3.7 Broken data sets . . . . .	139
3.7.1 Lack of virtual storage . . . . .	139
3.7.2 Initial loading problems . . . . .	140
3.7.3 Mismatch between catalog and data set . . . . .	142
3.7.4 Hardware errors . . . . .	144
3.7.5 Bad data or bad channel program . . . . .	145
3.7.6 Structural damage . . . . .	147
3.7.7 Improper sharing . . . . .	150
3.7.8 Mismatch between catalog and VTOC . . . . .	152
3.7.9 VSAM does not produce expected output . . . . .	153
3.7.10 Recovery scenarios . . . . .	154
3.7.11 Recovering ICF catalogs . . . . .	158
3.7.12 Recovering damaged VVDS entries . . . . .	158
3.8 IDC3009I message . . . . .	159
3.9 IDCAMS LISTCAT output fields . . . . .	166
3.9.1 High used RBA value (HURBA) for KSDS . . . . .	171
3.9.2 High allocated RBA value (HARBA) . . . . .	172
3.9.3 FREESPC . . . . .	173
3.9.4 High key RBA/CI . . . . .	173
3.9.5 High-level index RBA value . . . . .	173
3.9.6 Sequence set first RBA value . . . . .	173
3.9.7 Number of index levels . . . . .	173
3.9.8 Time stamps . . . . .	173
3.10 DFSMSDss PRINT command . . . . .	174
3.11 SMF record types related to VSAM data sets . . . . .	174
3.11.1 SMF record type 60 . . . . .	174
3.11.2 SMF record type 61 . . . . .	175
3.11.3 SMF record type 62 . . . . .	175
3.11.4 SMF record type 63 . . . . .	175
3.11.5 SMF record type 64 . . . . .	176
3.12 Resource Recovery Management Services (RRMS) and VSAM . . .	179
<b>Chapter 4. Managing your VSAM data sets . . . . .</b>	<b>181</b>
4.1 Reorganization considerations . . . . .	181
4.1.1 CI/CA splits . . . . .	181
4.1.2 The loss of useful space in data CA . . . . .	181
4.1.3 CI/CA splits causing free space increase . . . . .	183
4.2 Sharing VSAM data sets . . . . .	183

4.2.1	Write and read integrity . . . . .	184
4.2.2	Who is sharing the data set? . . . . .	185
4.2.3	Intra-address space sharing . . . . .	185
4.2.4	Cross-region options . . . . .	188
4.2.5	Cross-system options . . . . .	189
4.2.6	General share options — considerations . . . . .	191
4.2.7	Control Block Update Facility (CBUF) . . . . .	192
4.3	Catalog Search Interface . . . . .	193
4.3.1	CSI setup . . . . .	193
4.4	VSAM exploiters . . . . .	195
4.4.1	DB2 . . . . .	195
4.4.2	Hierarchical File System (HFS) . . . . .	195
4.4.3	CICS . . . . .	195
4.4.4	DFSMSshm . . . . .	195
4.4.5	DFSMSrmm . . . . .	197
4.4.6	OS/390 data sets . . . . .	197
4.4.7	Java/VSAM . . . . .	197
4.5	Media Manager, Open, Close, EOVS in VSAM . . . . .	200
4.5.1	OPEN macro . . . . .	201
4.5.2	CLOSE macro . . . . .	201
4.5.3	End-of-Volume (EOV) macro . . . . .	202
4.6	Transactional VSAM . . . . .	202
<b>Appendix A. Sample code . . . . .</b>		<b>209</b>
A.1	JRIO API examples . . . . .	209
A.1.1	Locate a record by key in keyed access record file . . . . .	209
A.1.2	Position to a record in a random access record file . . . . .	209
A.1.3	Read a record from a keyed access record file . . . . .	210
A.1.4	Read a record from a random access record file . . . . .	211
A.1.5	Update a record in a keyed access record file . . . . .	212
A.2	Accessing the VSAM Shared Information (VSI) . . . . .	213
A.3	Sample program to extract information from SMF record type 64 . . . . .	214
<b>Appendix B. Miscellaneous performance items . . . . .</b>		<b>223</b>
B.1	Our laboratory . . . . .	223
B.1.1	General lab description . . . . .	223
B.1.2	What do we measure? . . . . .	224
B.1.3	DASD cache concepts . . . . .	226
B.1.4	Cache Modes . . . . .	229
B.1.5	Using cache modes in a non-SMS data set . . . . .	235
B.1.6	Using cache in an SMS data set . . . . .	236
B.2	Cache analogy . . . . .	241
B.3	Share options analogy . . . . .	244



B.4 Symptoms (messages) from a broken data set . . . . .	245
B.5 IDCAMS Examine messages . . . . .	251
<b>Appendix C. Special notices . . . . .</b>	<b>253</b>
<b>Appendix D. Related publications . . . . .</b>	<b>257</b>
D.1 IBM Redbooks . . . . .	257
4.7 IBM Redbooks collections . . . . .	257
D.2 Other resources . . . . .	257
D.3 Referenced Web sites . . . . .	258
<b>How to get IBM Redbooks . . . . .</b>	<b>259</b>
IBM Redbooks fax order form . . . . .	260
<b>Index . . . . .</b>	<b>261</b>
<b>IBM Redbooks review . . . . .</b>	<b>265</b>



---

## Figures

1. General format of a control interval . . . . .	4
2. Sequence set . . . . .	7
3. Index set . . . . .	8
4. KSDS structure showing cluster components . . . . .	9
5. Entry sequenced data set (ESDS) . . . . .	12
6. Relative record data set (RRDS) . . . . .	15
7. Linear data set (LDS) . . . . .	17
8. DATA CLASS DEFINE ISMF panel . . . . .	20
9. AMS commands . . . . .	28
10. DITTO selection panel . . . . .	30
11. DITTO edit function . . . . .	31
12. VSAM edit panel . . . . .	32
13. DEFINE command required parameters . . . . .	34
14. Response time components . . . . .	38
15. Address space layout . . . . .	56
16. NSR buffering . . . . .	64
17. VSAM shared resources (LSR/GSR) . . . . .	72
18. CMS dictionary selection . . . . .	87
19. Striped VSAM data set . . . . .	94
20. Layering in VSAM data set striping . . . . .	95
21. Hiperbatch example . . . . .	109
22. HARBA, HURBA, and free space . . . . .	182
23. Java class model example . . . . .	198
24. Transactional VSAM environment . . . . .	203
25. Modes of access to VSAM data sets . . . . .	204
26. System logger overview . . . . .	207
27. Types of writes . . . . .	233
28. Association between MSR and cache usage attributes . . . . .	237
29. The tale . . . . .	242
30. Sharing VSAM data sets . . . . .	244



## **Tables**

1. Comparison of ESDS, KSDS, RRDS, VRRDS, and linear data sets. . . . .	23
2. Region JCL parameter. . . . .	57
3. Parameters affecting buffer allocation . . . . .	60
4. NSR — read sequential varying the number of buffers — STRNO=1 . . . .	66
5. NSR - Initial Load mode varying the number of buffers. . . . .	68
6. NSR buffering with direct access — STRNO=1. . . . .	70
7. Direct access: benefits of using SMB — updates and insertions. . . . .	79
8. Some effects of ACB's MACRF and storage class BIAS parameters . . . .	80
9. Initial load mode comparing SMB with no-SMB buffering . . . . .	80
10. Comparing compression . . . . .	90
11. Random processing: extended format vs. non-extended format data sets	118
12. NSR — read sequential varying the number of buffers . . . . .	120
13. Direct access: benefits of using SMB — updates and insertions. . . . .	120
14. Direct access: benefits of using SMB — updates and insertions. . . . .	121
15. IDC3009I message . . . . .	160
16. Relationship between share options and VSAM functions . . . . .	191



---

## Preface

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

This IBM Redbook will give you the information required to understand, evaluate, and use VSAM properly. It will clarify VSAM functions for application programmers who will be working with VSAM. The practical, straightforward approach should dispel much of the complexity sometimes associated with VSAM. Wherever possible an example is used to reinforce a description of a VSAM function.

This redbook is intended as a supplement to existing product manuals. It is intended to be used as an initial point of reference for VSAM functions. For example, parameters used in data set allocation to improve performance are described, and code examples provided, but the actual manual, *DFSMS/MVS Access Method Services for the Integrated Catalog Facility*, SC26-4906, must be consulted for complete syntax rules.

---

### The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization San Jose Center.

**Mary Lovelace** is a Consulting I/T Specialist at the International Technical Support Organization, San Jose Center. She has more than 20 years of experience with IBM in large systems and storage product education, system engineering and consultancy, marketing support, and systems programming.

**Dawn Dalisay** is an IT Specialist in IBM Philippines. She has 4 years experience with IBM in S/390 technical support. Her areas of expertise include VSAM, CICS, and VSE. She teaches a VSAM Fundamentals course for IBM Learning Services.

**Alvaro Salla** is an IBM retiree. He worked in IBM for more than 30 years, always in large systems. Alvaro wrote many redbooks and spent many years teaching, from S/360 to S/390. He has a Chemistry Engineer degree from the University of Sao Paulo, Brasil.

**Valeria Sokal** is a Business Partner from Brasil. She has 12 years experience as an OS/390 system programmer. Her previous residencies include OS/390 Workload Manager Exploitation and Implementation and ABC's for OS/390 System Programmers.

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

Ed Daray  
Storage Systems Group - San Jose

Savur Rao  
Storage Systems Group - San Jose

Charlie Burger  
Storage Systems Advanced Technical Systems Center - San Jose

Helen Witter  
Storage Systems Group - San Jose

Toby Marek  
Tivoli Systems - San Jose

Bob Haimowitz  
International Technical Support Organization- Raleigh

Paul Rogers  
International Technical Support Organization - Poughkeepsie

---

## Comments welcome

### Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in "IBM Redbooks review" on page 265 to the fax number shown on the form.
- Use the online evaluation form found at [ibm.com/redbooks](http://ibm.com/redbooks)
- Send your comments in an Internet note to [redbook@us.ibm.com](mailto:redbook@us.ibm.com)



---

## Chapter 1. VSAM basics

In this chapter we will build the foundation for the rest of the book. The concepts and terminology associated with VSAM data sets are reviewed. We explain how a VSAM data set is different from other data set types. The various types of VSAM data sets, and what makes them unique, are discussed, along with an explanation of how data is stored and accessed in a VSAM data set.

---

### 1.1 VSAM overview

In the early 1970s, IBM introduced a collection of three data set organizations — sequential, indexed, and direct-access, together with the access methods and utilities to be used on the mainframe operating systems.

This collection of data set organizations is called the Virtual Storage Access Method (VSAM). The word *virtual* relates to the fact that VSAM was introduced at approximately the same time as the initial IBM virtual storage operating systems OS/VS1 and OS/VS2.

VSAM was developed to replace the Indexed Sequential Access Method (ISAM), which is a much older technology. ISAM has major processing overheads which IBM wanted to improve.

---

### 1.2 What is VSAM?

VSAM is one of several access methods that defines the technique by which data is stored and retrieved. It is a GET/PUT interface used to transfer data from a direct access storage device (DASD) to an application program. VSAM does not support data stored on tape.

VSAM is used to organize and access data, and maintain information about this data which is stored or referenced in a catalog. VSAM data sets must be cataloged in an integrated catalog facility (ICF) structure.

Records are arranged by an index key or by relative byte addressing. VSAM uses direct or sequential processing of fixed and variable length records on DASD.

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

- **Catalog Management**

The catalog, which contains information about the data sets, can be an ICF or a VSAM catalog. DFSMS/MVS deals only with ICF catalogs. All VSAM data sets must be defined in an ICF catalog.

In this book, we will not go into detail on catalog management. For more information, refer to the IBM Redbooks *Enhanced Catalog Sharing and Management*, SG24-5594, and *Integrated Catalog Facility Backup and Recovery*, SG24-5644.

- **Record management**

The purpose of record management is to maintain records in a VSAM data set for an application or a system program. Today, VSAM supports five data set organizations:

- Key-sequenced data set (KSDS)
- Entry-sequenced data set (ESDS)
- Fixed-length relative record data set (RRDS)
- Variable-length relative record data set (VRRDS)
- Linear data set (LDS)

The primary difference between the VSAM data set organizations is the way in which their records are stored and accessed.

---

## 1.3 VSAM terminology

Before we discuss VSAM data set organizations in detail, we need to review some terms that will be used throughout the book.

### 1.3.1 Logical record

Logical records in VSAM data sets are stored differently than logical records in non-VSAM data sets. A logical record is a unit of information used to store data in a VSAM data set. The terms *logical record* and *record* are used interchangeably in this book.

### 1.3.2 Physical record

A physical record is device dependent and calculated by catalog at the time the data set is defined. VSAM uses a control interval as its smallest unit of information to transfer.

### 1.3.3 Control interval

Logical records are contained in a control interval (CI). The fundamental building block of every component of a VSAM data set is the control interval. A control interval is the unit of information that VSAM transfers between the storage device and the processor. One CI can be made of one or more physical blocks of DASD.

A CI consists of the following:

- Logical records stored from beginning to end
- Unused space, referred to as free space, for data records to be inserted into or lengthened
- Control information, which is made up of two types of fields; one control interval definition field (CIDF) per CI, and one or more record definition fields (RDF) per logical record.

- CIDF is a 4-byte field.

It contains information about the length of data in the CI and the amount and location of free space.

- RDF is a 3-byte field.

It describes the length of records and how many adjacent records are of the same length.

For more information on the structure of the control information fields, refer to *DFSMS/MVS Using Data Sets*, SC26-4922.

Figure 1 shows the general format of a CI. The CI components and properties may vary depending on the data set organization. For example, an LDS does not contain CIDFs and RDFs in its CI. All of the bytes in the LDS CI are data bytes. Refer to 1.4, “Data set types” on page 11 for more details.

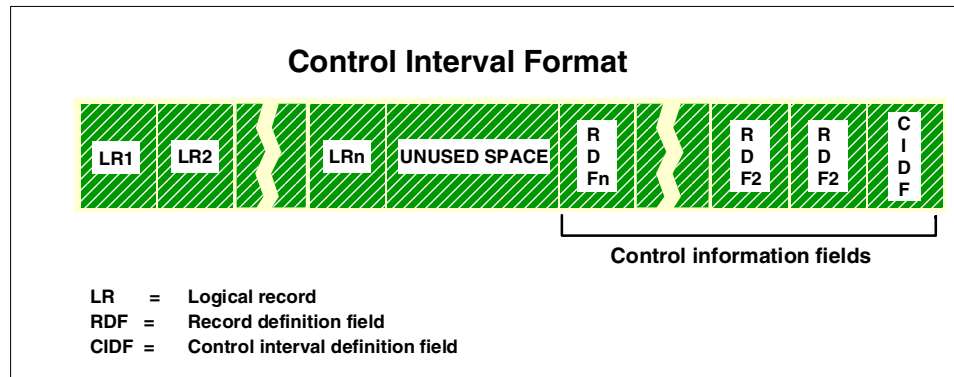


Figure 1. General format of a control interval

The size of CIs can vary from one data set to another, but all the CIs within the data component of a particular data set must be of the same length. Refer to 1.3.8, “Component” on page 6 for details on the data component.

You can request the CI size using the AMS DEFINE command, you can let VSAM determine the CI size, or you can specify a data class, thereby using the CISIZE defined by your storage administrator.

#### 1.3.4 Control area

A control area (CA) is two or more CIs put together into fixed-length contiguous areas of DASD. A VSAM data set is composed of one or more CAs. The number of CIs in a CA is fixed by VSAM.

The maximum size of a CA is one cylinder and the minimum size is one track. The CA size is implicitly defined when you specify the size of a data set at data set definition.

#### 1.3.5 Splits

CI and CA splits occur as a result of data record insertions. If a record is to be inserted and there is not enough free space in the CI, the CI will be split. Approximately half of the records in the CI are transferred to a free CI and the record to be inserted is placed in the original CI.

If there are no free CIs and a record is to be inserted, a CA split occurs.

The physical sequence of records and CIs is no longer the same as the logical sequence after a split. A new index entry is inserted in the sequence set for the new CI, and the existing index entry is updated.

### **1.3.6 Spanned records**

Spanned records are records that are larger than the CI size. In order to have spanned records the file must be defined with the SPANNED attribute at the time it is created. Spanned records are allowed to extend across or span control interval boundaries. The RDF's will 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. In other words, the free space at the end of the last segment will not be filled with the next record. This free space can only be used to extend the spanned record.

If spanned records are used for KSDS, the primary key must be within the first control interval.

### **1.3.7 Relative byte address**

The relative byte address (RBA) is used to determine the position of a record in a VSAM data set. It is the physical location of a logical record contained in a data set. The RBA is the offset of a logical record from the beginning of the data set.

The first record in the VSAM data set has an RBA of zero; the second record has an RBA equal to the length of the first record, and so on. The RBA of a logical record depends only on the record's position in the sequence of records. The RBA is always expressed as a fullword binary integer.

The RBA of a record includes the free space and control information in the CI.

RBAs might change when a control interval split occurs or when records are added, deleted, or changed in size. With compressed data sets, the RBAs for compressed records are not predictable. Therefore, access by address is not suggested for normal use.

### 1.3.8 Component

A component is a name given to the individual parts of a VSAM data set. KSDS and VRRDS have data and index components. ESDS, RRDS and LDS only have data components.

#### 1.3.8.1 Data component

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

#### 1.3.8.2 Index component

The index component is a collection of logically sequenced keys. The key is a value taken from a fixed defined field in each logical record. The key determines the record's position in the data set.

Using the index, VSAM is able to randomly retrieve a record from the data component when a request is made for a record with a certain key. VSAM divides the index CI into sections in order to speed up the search of a key.

A VSAM index can consist of more than one level. Each level contains pointers to the next lower level.

The index component consists of two parts: sequence set and index set.

- **Sequence set**

The sequence set is the lowest level of index CIs and directly points to the data CI in the CA. There is one CI in the sequence set for each data CA.

It 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 (see Figure 2).

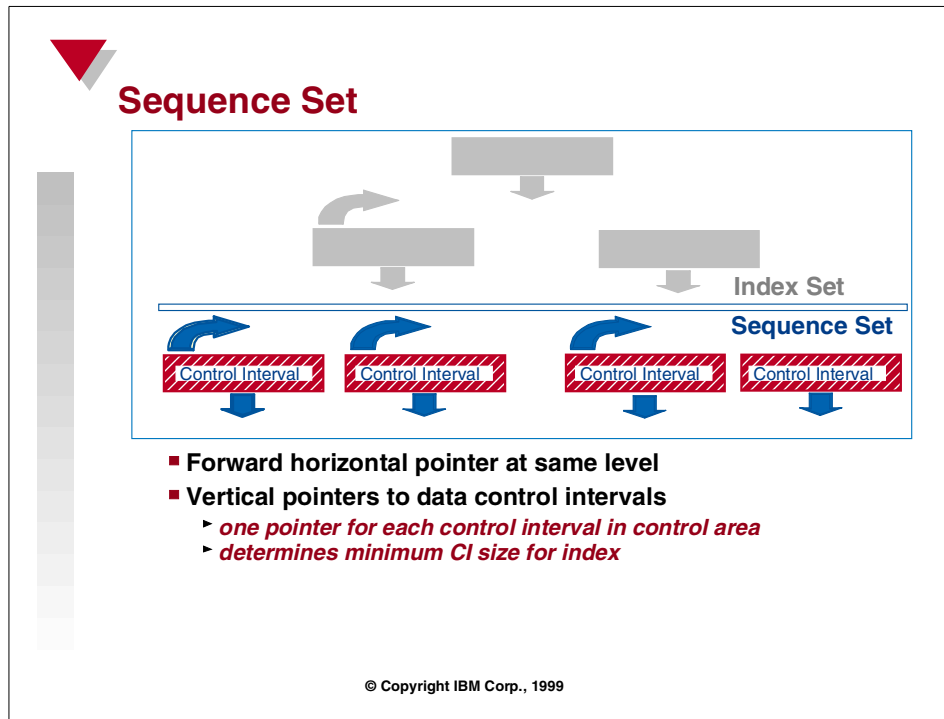


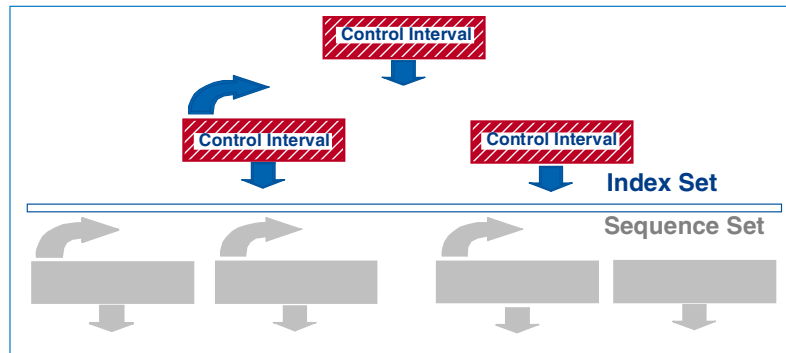
Figure 2. Sequence set

- **Index set**

The index set is the remainder of the index component. If there is more than one sequence set CI, VSAM automatically builds another index level. Each CI in the index set contains pointers and high key information for CIs in the next lower level of the index. See Figure 3.

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

## Index Set



- Forward horizontal pointer at same level
- Vertical pointers to next lower level index records
- Optionally *replicated* on DASD track

Figure 3. Index set

### 1.3.9 Cluster

All the VSAM data set types are defined as clusters. For a KSDS, a cluster is the combination of a data component and an index component. The cluster provides a way to treat index and data components as a single component with its own name. You can also give each component a name, and process the data portion separately from the index portion.

RRDS, ESDS, and LDS formats 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.



Figure 4 shows the components of a KSDS.

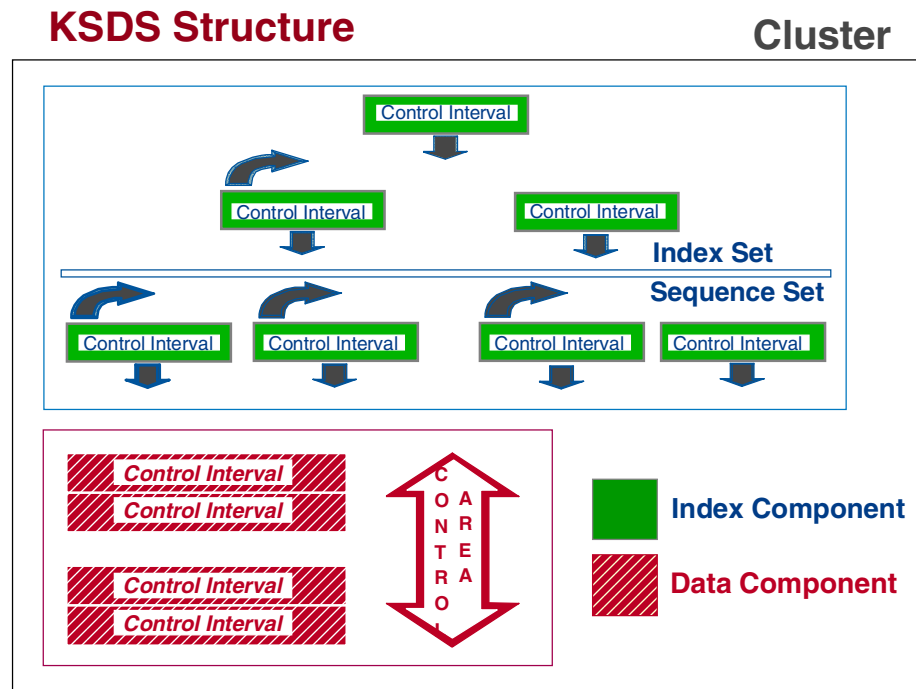


Figure 4. KSDS structure showing cluster components

### 1.3.10 Keys

To differentiate between keys used in VSAM Objects, the key used in a base cluster is called a primary key or base key. The key used in an alternate index is called an alternate key.

In VSAM key sequenced organization, a record must have a unique, imbedded fixed-length primary key located 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 keys may occur in more than one logical record. This allows the search with a given alternate key to read all base cluster records containing this alternate key.

### 1.3.11 Sphere

A sphere is a VSAM cluster and its associated data sets. These data sets are the alternate indexes (AIXs) of the cluster. An AIX is a KSDS containing index entries organized by the alternate keys of its associated base data records. It provides another way of locating records in the data component of a cluster.

An AIX can only be defined over a key-sequenced or entry-sequenced cluster.

### 1.3.12 Alternate indexes

AIXs enable the logical records of an ESDS or of a KSDS (in this context called a base cluster) to be accessed sequentially and directly by more than one key field. This eliminates the need to store the same data in different sequences in multiple data sets for the purposes of various applications.

Any field in the base cluster record can be used as an alternate key. It may also overlap the primary key (in a KSDS), or with any other alternate key. The alternate key field must be a contiguous field with the same offset in each record. In a spanned record, this field must be located totally in the first control interval.

Each alternate index consists of an index component and a data component. These two components together form an alternate index for the base cluster. It is a KSDS built by VSAM. An alternate index will contain the different alternate key values of a certain alternate key. For every alternate key field, a different alternate index is needed. The records in the data component contain an alternate key and one or more pointers to data in the base cluster. For an entry-sequenced base cluster, the pointers are RBA values. For a key-sequenced base cluster, the pointers are prime key values.

There may be more than one primary key or RBA per alternate key. The primary keys or RBAs will be in ascending sequence within an alternate index record after loading. This fact will not necessarily be true after the base cluster has been updated (especially the alternate key fields) or new records added, as any new primary key or RBA will be inserted at the end of the appropriate alternate index record.

The AMS program allows you to define, and then to create AIXs when the BLDINDEX command is specified. An AIX is defined only after its associated base cluster has been defined, and 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 which 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.

### **1.3.13 Alternate index paths**

Before accessing a KSDS or ESDS through an alternate index, a path must be defined. A path is the means by which a base cluster is accessed by way of its alternate indices. A path is defined and named using the AMS DEFINE PATH command. At least one path must be defined for each of the alternate indices through which the base cluster is to be accessed. 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.

---

## **1.4 Data set types**

We have mentioned that VSAM supports five data set organizations: entry-sequenced, key-sequenced, fixed-length and variable-length relative record, and linear. In this section we will discuss each of these in detail.

### **1.4.1 Entry sequenced data set (ESDS)**

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. This could be fixed or variable length records.

All new records are placed at the end of the data set. Existing records can never be deleted. If the application wants to delete a record, it must flag that record as inactive. As far as VSAM is concerned, the record is not deleted. It is the responsibility of the application program to identify that record as invalid.

Records can be updated, but without length change. To change the length of a record, you must either store it at the end of the data set as a new record, or override an existing record of the same length that you have flagged as inactive.

A record can be accessed sequentially or directly by its RBA:

- Sequential processing — VSAM automatically retrieves records in stored sequence. Sequential processing 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, positioning is necessary before sequential processing can be performed.
- Direct processing — When a record is loaded or added, VSAM indicates its RBA. To retrieve records directly, you must supply the RBA for the record as a search argument. Although an ESDS does not contain an index component, you can build an alternate index to keep track of these RBAs.

Skip sequential processing is not allowed for an ESDS. Refer to 1.4.2, “Keyed sequenced data set (KSDS)” on page 13 for more information on skip sequential processing.

Figure 5 shows the format of an ESDS.

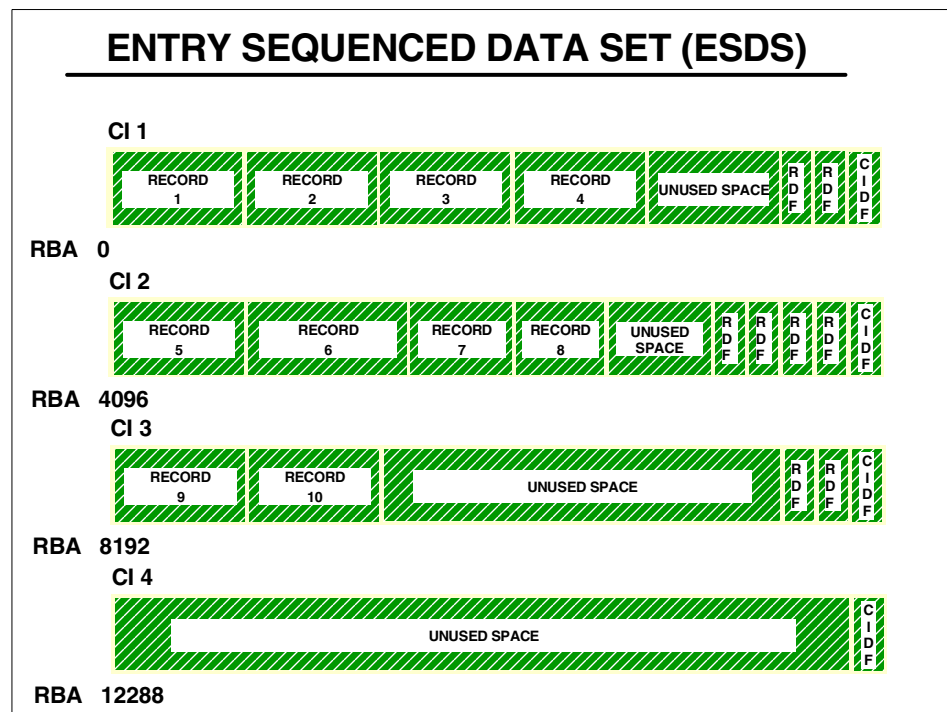


Figure 5. Entry sequenced data set (ESDS)

Empty spaces in the CI are referred to as unused space because they can never be used. This is a result of CI internal fragmentation (spanned is only for logical records greater than the CI).

You specify ESDS organization using the IDCAMS DEFINE command and specifying the NONINDEXED parameter.

### 1.4.2 Keyed sequenced data set (KSDS)

In a KSDS, records are placed in the data set in ascending collating sequence by key. The key contains a unique value that determines the record's collating position in the data set. The key must be in the same position in each record.

The key data must be contiguous and each record's key must be unique. After it is specified, the value of the key cannot be altered, but the entire record can be deleted.

When a new record is added to the data set, it is inserted in its collating sequence by key. This could be fixed or variable length records.

Refer to Figure 4 on page 9 for the structure of a KSDS.

There are three methods by which to access a KSDS. These are sequential, direct, or skip-sequential.

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

VSAM uses the index to access data records in ascending or descending sequence by key. When retrieving records, you do not need to specify key values because VSAM automatically obtains the next logical record in sequence. The sequence set is used to find the next logical CI.

Sequential access allows you to avoid searching the index more than once. Sequential is faster than direct for accessing multiple data records in ascending key order.

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

You need to supply a key value for each record to be processed. You 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 CA containing the record.

Direct access saves you a lot of overhead 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 data set.

VSAM retrieves selected records, but in ascending sequence of key values. Skip sequential processing allows you to

- Avoid retrieving the entire data set sequentially in order to process a relatively small percentage of the total number of records
- Avoid retrieving the desired 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 if it contains the requested record. If the first skip-sequential search is the first access after opening the data set, a direct search is initiated by VSAM to find the first record. From then on the index sequence set level will be used to find the subsequent records. If other operations were performed before (for example, read sequential), either the last position of that operation will be used as a starting point to search the sequence set records, or a re-positioning is necessary.

You specify the KSDS organization using the IDCAMS DEFINE command with the INDEXED parameter.

### **1.4.3 Relative record data set (RRDS)**

An RRDS consists of a number of pre-formatted fixed-length slots. Each slot has a unique relative record number, and the slots are sequenced by ascending relative record number.

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 data record is fixed and its relative record number 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. 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.

Figure 6 shows the format of an RRDS.

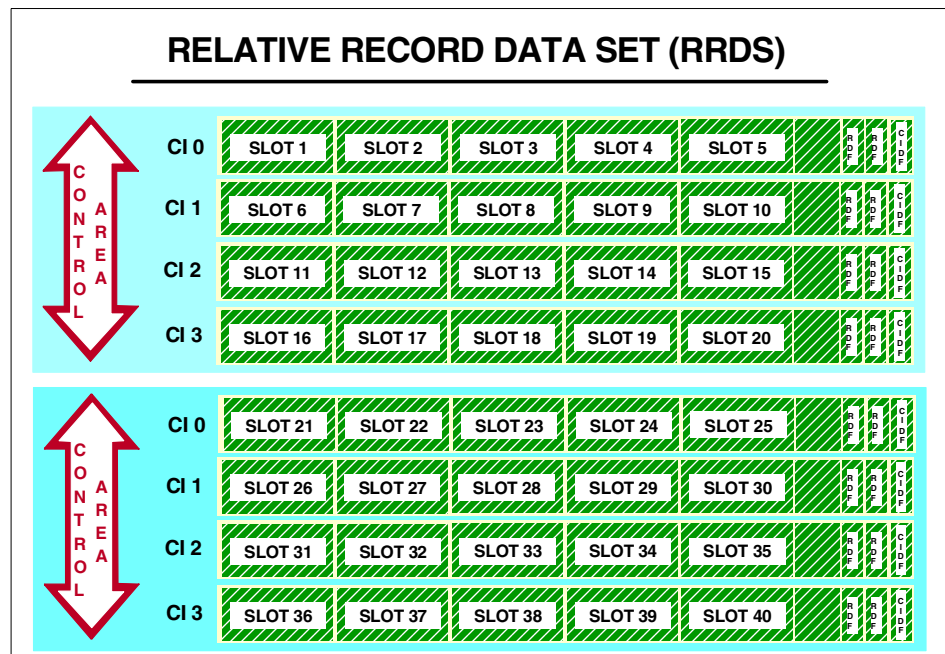


Figure 6. Relative record data set (RRDS)

#### 1.4.3.1 Typical RRDS processing

The application program inputs the relative record number of the target record and VSAM is able to find its location quickly using a formula that takes into consideration the geometry of the DASD device. The relative record number is always used as a search argument.

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

- RRDS sequential processing is treated the same way as ESDS sequential processing. 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. RRDS direct address processing by supplying the RBA is not supported.
- Skip-sequential processing is treated like an RRDS direct processing request, but the position is maintained. Records must be in ascending sequence.

You specify the RRDS organization using the IDCAMS DEFINE command with the NUMBERED option.

#### **1.4.4 Variable relative record data set (VRRDS)**

A VRRDS is similar to a fixed-length 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 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.

You can specify free space for inserting records and increasing the length of a record.

VRRDS is a KSDS processed as an RRDS, so an index will be created.

You specify the VRRDS organization with the IDCAMS DEFINE command with the NUMBERED option and variable length record.

#### **1.4.5 Linear data set (LDS)**

A linear data set (LDS) contains data that can be accessed as byte-addressable strings in virtual storage. It is a VSAM data set with a control interval size of 4096 bytes. An LDS has no imbedded 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 application program, but records do not exist from the point of view of VSAM.

Like the ESDS and RRDS, an LDS contains a data component only.

An LDS can only be defined using ICF catalogs. IDCAMS is used to define an LDS but it is accessed using a Data-In-Virtual (DIV) macro. An LDS is sometimes referred to as a DIV object.



Figure 7 shows the format of an LDS.

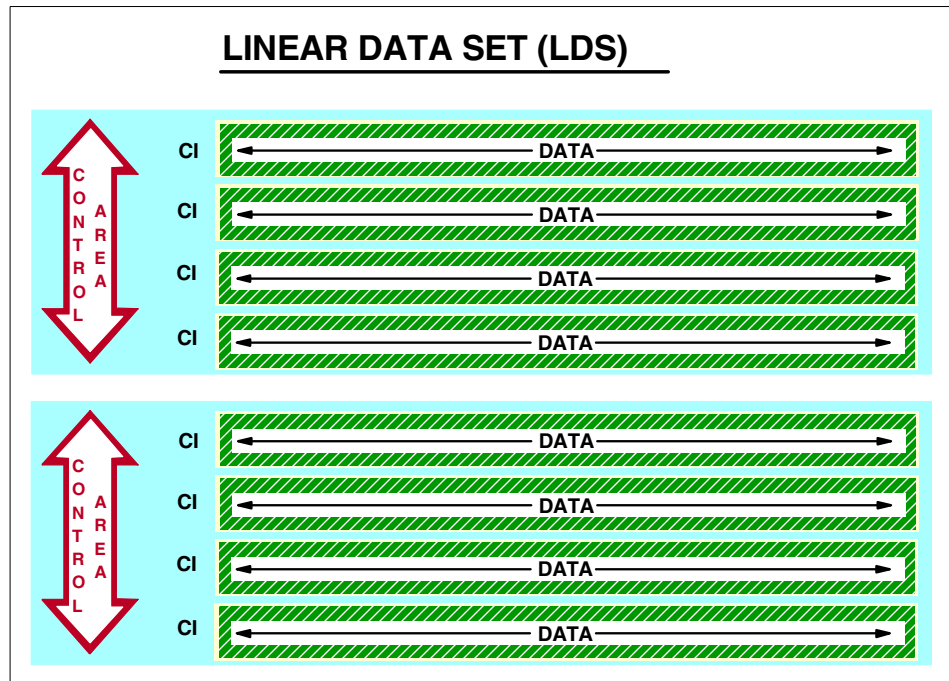


Figure 7. Linear data set (LDS)

You specify the LDS organization with the IDCAMS DEFINE command specifying the LINEAR parameter.

#### 1.4.5.1 Data-in-Virtual

Data-in-Virtual (DIV) is an optional and unique buffering technique used for LDS data sets. Application programs can use DIV to “map” an LDS data set or a portion of a data set into an address space, a data space, or a hiperspace.

Data is read into central storage through the paging algorithms only when that data block is actually referenced. During RSM page steal processing, only changed pages are written to auxiliary storage. Unchanged pages are discarded since they can be retrieved again from the permanent data set.

DIV is designed to improve the performance of applications that process large files 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 or table files.

#### 1.4.5.2 Mapping a linear data set

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

- *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 may 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 will result in a page fault which causes DIV services to read the data from the linear data set into the window.

- *DIV SAVE* can be used to write out changes to the data object.
- *DIV RESET* can be used to discard changes made in the window since the last SAVE operation.

---

### 1.5 Extended format data set

An extended format data set can be thought of as having the same characteristics as a physical sequential data set. Extended format is a technique that affects the way count key data is stored in a 3390/3380 logical track. It increases the performance and the reliability of an I/O operation. It is recommended that you convert your data sets to extended format to get better performance, reliability and functionality. A good time is when you next reorganize a VSAM data set.

An extended format data set for VSAM can be allocated for KSDSs, ESDSs, RRDSs, VRRDSs, and LDSs. The benefits available for extended format data sets include:

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

Extended format data sets must be system-managed. They 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 related to an extended format cluster are also extended format.

When in extended format, there is no support for the key-range VSAM option, MVS checkpoint restart, or hiperbatch. Key-range is not recommended in an SMS environment with the new RAID controllers.

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

- Catalogs
- System data sets
- Temporary data sets

If a data set is allocated as an extended format data set, 32 bytes (x'20') are added to each physical block. When the control interval size is calculated or explicitly specified, this physical blockoverhead may increase the amount of space actually needed for the data set. The 32-byte suffix of each data record contains:

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

The data records in an extended format data set are the same length (a sort of fixed block architecture). The block format and the suffix are transparent to the application, that is, the application does not require internal or external modifications to create and use the new data set format.

All the VSAM organizations can be either extended format or non-extended format. To convert a non-extended format data set to extended format, or to allocate an extended format data set, you need to create an SMS data class (DC) with the DATASETNAME TYPE field equal to EXT and assign the data sets to that data class.

---

## 1.6 Extended addressability (EA)

Extended addressability (EA) was introduced in DFSMS/MVS 1.3, for KSDS data sets. Since DFSMS/MVS 1.4, EA is supported in record level sharing (RLS). With DFSMS/MVS 1.5, support for extended addressability is extended to all other VSAM record organizations.

With EA, the 4G architectural limit for data set size imposed by using the 4-byte field for the relative byte address (RBA) was eliminated.

It is important to state up-front that extended addressability and extended format are not the same concept. Extended format is a way of storing data in a 3390/3380 logical volume. Extended addressability is the ability of allowing larger VSAM data sets. However, extended format is a pre-prerequisite for extended addressability.

Using EA, the size limit for a VSAM data set is determined by either:

- CI size multiplied by 4 GB
- The volume size multiplied by 59

A 4 K CI size yields a maximum data set size of 16 TB, while a 32 KB CI size yields a maximum data set size of 128 TB. A 4K CI size is preferred by many applications for performance reasons. No increase in processing time is expected for extended format data sets that grow beyond 4 GB.

To use EA, the data set must be:

- SMS-managed
- Defined as extended format

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

DSNTYPE=EXT and EXTENDED ADDRESSABILITY=Y

Figure 8 shows the ISMF panel where you specify EF and EA when you define or alter a data class.

```

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 . . . . . B (P=Preferred, R=Required or blank)
Extended Addressability . . . Y (Y or N)
Record Access Bias . . . . S (S=System, U=User or blank)
Reuse . . . . . N (Y or N)
Initial Load . . . . . S (S=Speed, R=Recovery or blank)
Spanned / Nonspanned . . . . S (S=Spanned, N=Nonspanned or blank)
BWO . . . . . _ (TC=TYPECICS, TI=TYPEIMS, NO or blank)
Log . . . . . _ (N=NONE, U=UNDO, A=ALL or blank)
Logstream Id . . . . . _____

```

Figure 8. DATA CLASS DEFINE ISMF panel

After creating a data class with the attributes above, users can code the DATACLAS value on their DD statements or let the ACS routines assign the appropriate class for their eligible data. The only other method in which JCL can be used to create a KSDS with extended addressability, is through the DD statement keyword LIKE.

Applications that access a VSAM extended format KSDS through a user-specified key can take advantage of this support without making JCL or code changes.

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

- RPL macro, added the XRBA subparameter to the OPTCD parameter to indicate that extended addressability will be used. It must be used when 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 high-used RBA (HURBA)
  - XHALCRBA, to obtain the high-allocated RBA (HARBA)

The fields above 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 and the type of access uses the RBA specified. Special provisions are allowed for certain types of requests (for example GET SEQ,ADR or GET ADR,DIR,LRD).

A major idea in the EA design, is to make applications, such as backup, transparent to the function. That is, we do not require an extended field (8-bytes) XRBA, unless it is a positioning request. In the case of a backup when the data set is to be just read sequentially from the beginning, requiring 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.

With DB2 V1.6 and later EA can be used for table space for.

A VSAM EF KSDS defined for EA must not be shared with any system running a release prior to DFSMS/MVS 1.3. Toleration maintenance is available for DFSMS/MVS 1.2 systems which support VSAM extended format data sets without EA. A DFSMSdftp toleration PTF allows systems at this level to issue an error message if any attempt is made to open a VSAM KSDS with extended addressability. DFSMSdss provides a toleration PTF so that an error message is issued when a logical DUMP, RESTORE, or COPY operation is attempted on a KSDS with extended addressability.

IMS supports EF KSDS data sets only when EA is not permitted.

CICS, which provides an interface that allows users to access data by RBA, is not restricted from using VSAM EA. Contact your IBM representative for more information on the product level that exploits this support.

Data warehousing projects that require table spaces larger than 1 terabyte can use the EA support for linear data sets provided in DFSMS/MVS 1.5. To do this, assign a data class with the extended addressability attribute to the data set when it is defined. The data class should have the following attributes specified for it:

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

Then make sure your data class ACS routine for DB2 permits the use of the data class created.

The message IDC3351I \*\* VSAM {OPENICLOSEII/O} RETURN CODE IS 116 is produced when you try to go beyond 4 GB in a VSAM data set without EA.

## 1.7 Comparing VSAM data set organizations

Table 1 provides a summary of the characteristics of VSAM data set types described in this chapter.

Table 1. 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	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 have fixed length	Records have 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)
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 given up by a deleted or shortened record becomes free space	A slot given up by a deleted record can be reused	Space 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

---

## 1.8 A brief history of VSAM

In the early 1970s, IBM introduced a collection of three data set organizations — sequential, indexed, and direct-access, together with the access methods and utilities to be used on the mainframe operating systems.

This collection of data set organizations is called *Virtual Storage Access Method (VSAM)*. 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.

VSAM was developed to replace the Indexed Sequential Access Method (ISAM) which is a much older technology. ISAM has major processing overheads which IBM wanted to improve.

Throughout the years, there have been many improvements to VSAM. Among these are: extended format and extended addressability which allows a data set to go beyond the 4 GB limitation, system managed buffering (SMB) for improved performance, data compression, record level sharing (RLS), and most recently, data striping and transactional VSAM.

A brief history of the VSAM enhancements and the level of DFSMS/MVS the were introduced in follows:

### **DFSMS 2.10**

- Data striping and multi-layering for all VSAM data sets
- Transactional VSAM

### **DFSMS/MVS 1.5**

- Data striping for LDS through an SPE
- Extended addressability and SMB support for ESDS, RRDS, VRRDS and LDS

### **DFSMS/MVS 1.4**

- SMB for KSDS
- RLS support of extended addressability for KSDS

### **DFSMS/MVS 1.3**

- VSAM RLS
- Extended addressability supporting more than 4 gigabytes in a VSAM cluster for KSDS



## DFSMS/MVS 1.2

- Data compression for extended format VSAM KSDS (as well as BSAM and QSAM)
- Partial space release for extended format VSAM KSDS
- More secondary allocation options for extended format VSAM KSDS
- Removal of Backup While Open restrictions

---

### 1.9 Choosing a VSAM data set type

During the application development process you need to make decisions about your data model. Among them are the data organization and the type of access, performance, and recovery tools you need. VSAM has several organizations and different ways for accessing your data. How to choose the one for your application is discussed here.

The major issues are functionality, performance, and recovery capabilities. In respect to functionality, refer to 1.7, “Comparing VSAM data set organizations” on page 23. Regarding performance, refer to 2.7.2.7, “Use of VSAM data sets” on page 110. Refer to Chapter 3, “Recovery of VSAM data sets” on page 127, to get more information on recovery aspects.

Before you select a particular VSAM organization, you need to have answers for the following questions:

- What is the main purpose of your data set? Does it look more like a log, a data base, 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?
- Does the record length change?
- Do you need to have insertions and deletions?
- Is the data set going to be extended ?
- How often will you need to delete records?
- Will 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 explore DIV?

- Do you want to use data compression?
- Do you need the utility functions provided by IDCAMS?

When you have answered these questions, we can use them to choose the best organization for your data set. Remember that VSAM data sets cannot be processed using non-VSAM applications. Similarly, non-VSAM data sets cannot be processed using the VSAM access method.

Following are our recommendations.

- Use QSAM or BSAM if:
  - 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 KSDS if:
  - The data access will be sequential, skip sequential, and direct access by a key field.
  - You would prefer easy programming for direct data processing.
  - There will be many record insertions, deletions, and logical record length variations.
  - You may optionally access records by an alternate index.
  - Complex recovery (due to index and data components) is not a issue.
  - You want to use data compression.
  - The data is suitable for a sort of mini-database in an online application environment like CICS/VSAM.
- Use VRRDS if:
  - You have the same requirements as for a KSDS, but will use record number instead of a key field as argument.
- Use RRDS if:
  - The record processing will be 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 records identified by a continuous and dense pattern of numbers.
- All records are fixed length.
- There are a small number of 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 if:
  - You need sequential and direct record processing (not by a key field, but by an RBA).
  - You are using only logical record insertions or deletions (in the application control).
  - You are using a batch processing application.
- Use LDS if:
  - You want to exploit DIV.
  - Your application manages logical records.
  - Performance is an issue.

A final comment is that many times, you will be using a specific VSAM organization depending on the software product you are running — for example, DB2 uses LDS data sets. In such a case, this will be your only option.

---

## 1.10 Accessing VSAM data

VSAM data sets can be accessed using several methods, for example, IDCAMS, DITTO, batch and CICS application programs, and DB2.

Batch and CICS application programs can be written using languages that support VSAM, such as COBOL and Assembler. To obtain VSAM services, these application programs use VSAM macros. For details, refer to *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913.

### 1.10.1 IDCAMS

Access method services (AMS) is a service program used with VSAM to establish and maintain catalogs and data sets invoked by JCL or TSO.

The IDCAMS program can be run with the AMS command and its parameters as input to the IDCAMS program. You can call the IDCAMS program from within another program and pass the AMS command and parameters to the IDCAMS program.

We can create and process the various types of VSAM data sets using IDCAMS, which is included with DFSMS/MVS. The majority of these functions are covered in this book.

There are two types of AMS commands for catalogs and user data sets, functional commands and modal commands (see Figure 9). Functional commands are used to request the actual work (for example, defining a data set or listing a catalog).

Modal commands allow the conditional execution of functional commands. Time sharing option (TSO) users can use functional commands only. See *DFSMS/MVS Access Method Services for the Integrated Catalog Facility*, SC26-4906 for more information.

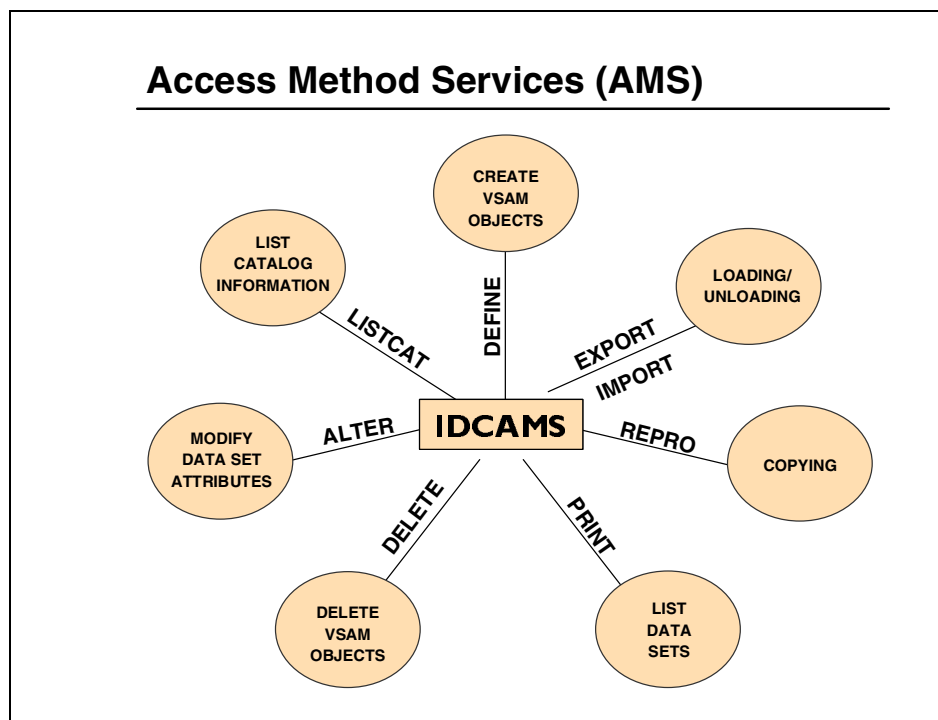


Figure 9. AMS commands

IDCAMS can be invoked:

- As a job or jobstep by specifying PGM=IDCAMS on the EXEC card
- From a TSO terminal
- From an application program

### 1.10.2 Accessing HFS files through VSAM

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

- JCL DD statement specifying PATH=pathname
- SVC99
- TSO ALLOCATE command

HFS files are simulated as an ESDS. However, since HFS files are not actually stored as ESDSs, VSAM cannot simulate all the characteristics of a sequential data set. As a result, there are certain macros and services which have incompatibilities or restrictions when dealing with HFS files. Refer to *DFSMS/MVS Using Data Sets*, SC26-4922 for more information.

### 1.10.3 DITTO/ESA

DITTO is a very powerful utility 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 invoke DITTO as start procedures may vary with the installation.

In full-screen mode, you can use menus, online help, and interactive browse and update functions. You will probably find full-screen mode 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. Refer to *DITTO/ESA V1R3 User's Guide*, SH19-8221 for more information on using DITTO.

Figure 10 shows the DITTO Task Selection Menu. Choose option 1 then option 3 to browse a VSAM data set.

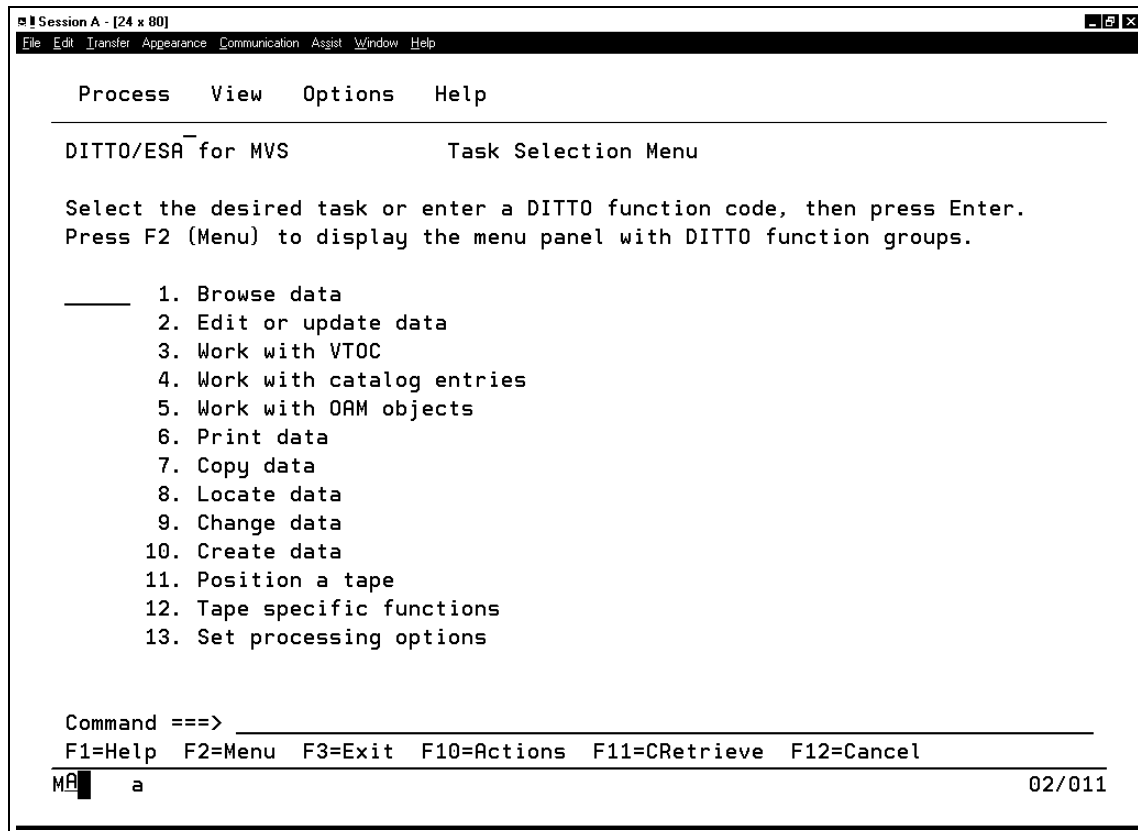


Figure 10. DITTO selection panel

Choose option 2 on the Task Selection Menu, then option 1 to edit a VSAM data set.

Figure 11 shows the DITTO edit menu.

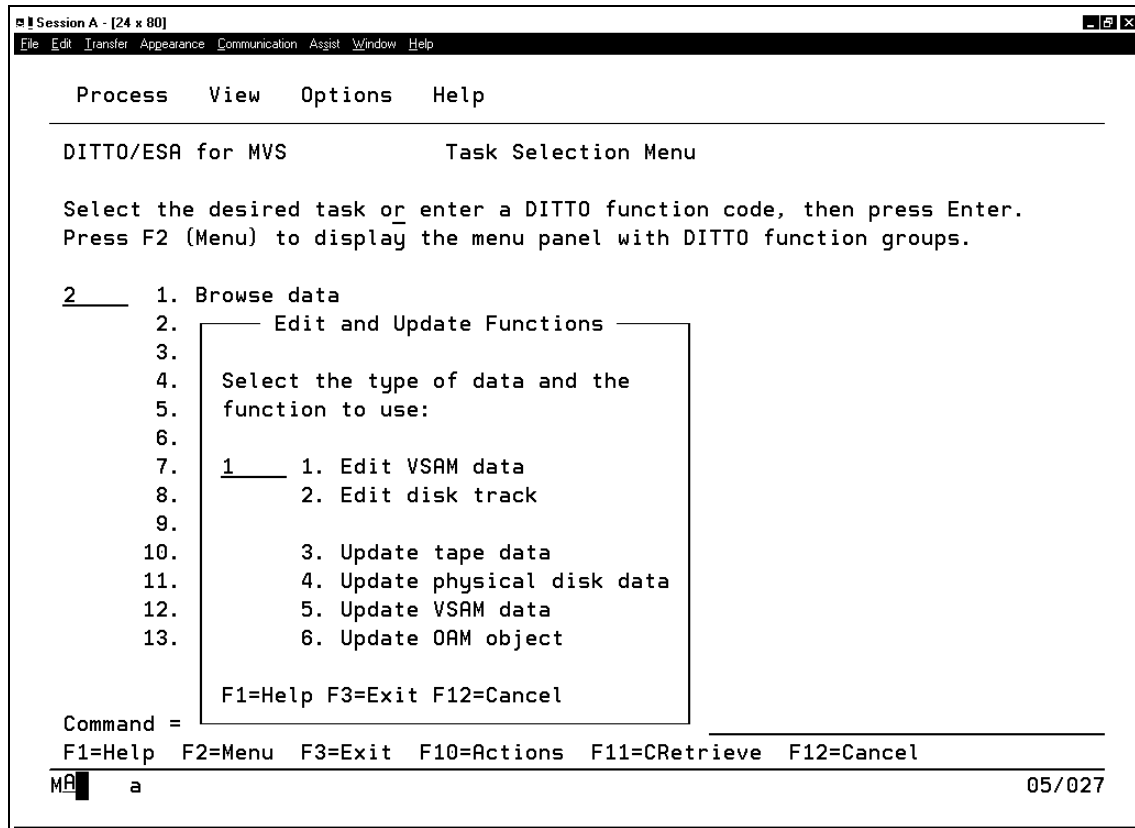


Figure 11. DITTO edit function

Figure 12 shows an example of editing a record using DITTO.

The screenshot displays a terminal window titled "Session A - [24 x 80]". The menu bar includes "File", "Edit", "Transfer", "Appearance", "Communication", "Assist", "Window", and "Help". Below the menu bar, the main title bar reads "Process View Options Help". The main content area shows the "DITTO/ESA for MVS" interface for "VE - VSAM Edit", indicating "1 string(s) changed".

Metadata fields include:

- DSNAME VALERIA.KSDS.K4
- VOLSER TSMS16 Type KSDS
- Col 1 (underlined) Format CHAR
- Insert length 80

A data line shows: <==5>..10....5...20....5...30....5...40....5...50....5...60....5...70...

The data section is titled "\*\*\*\* Top of data \*\*\*\*" and lists 12 records:

Record Number	Key	Description
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.....

At the bottom, the command line shows: Command ==> change /VSAM/DAWN/ (underlined) Scroll PAGE. Below this, function key definitions are listed: F1=Help, F2=Zoom, F3=Exit, F4=Left, F5=Right, F6=RFind, F7=Bkwd, F8=Fwd, F10=Actions, F11=CRetrieve, F12=Cancel. The status bar at the bottom left shows "MA" and "a", the center shows "A", and the right shows "22/033".

Figure 12. VSAM edit panel



---

## 1.11 Defining VSAM data sets

You can define a VSAM data set using any of the following methods:

- **IDCAMS** DEFINE or ALLOCATE commands.
- **ALLOCATE command** from a TSO terminal. The TSO commands are described in *OS/390 TSO/E Command Reference*, SC28-1969.
- **JCL DD statements**. All data sets can be defined directly through JCL. For information on using JCL, see *OS/390 MVS JCL Reference*, GC28-1757 and *OS/390 MVS JCL User's Guide*, GC28-1758.
- **Dynamic allocation**. The DYNALLOC macro is described in *OS/390 MVS Authorized Assembler Services Guide*, GC28-1763.

### 1.11.1 Using IDCAMS

When using IDCAMS to define a VSAM data set, you specify the following:

- Data set name or cluster name. Component names are optional.
- Data set type. The default is INDEXED (KSDS).

INDEXED	KSDS
LINEAR	LDS
NONINDEXED	ESDS
NUMBERED	RRDS
- Space allocation, both primary and secondary allocations, and the volumes on which the cluster's components are to have space.
- Data set attributes, such as recordsize and CI size. For a KSDS, you specify key information and free space.

Figure 13 shows the syntax of the DEFINE command and 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 13. DEFINE command required parameters

### 1.11.2 System-managed data sets

For SMS-managed VSAM data sets, only the NAME is required. DFSMS must be active to define SMS-managed data sets. You can specify data class, management class and storage class parameters and take advantage of the attributes defined by your system administrator.

This is an example of a JCL to define an SMS-managed VSAM data set.

```
***** Top of Data *****  
//DEFVSAM JOB 'DEF SMS-MANAGED VSAM DS',MSGCLASS=X,NOTIFY=&SYSUID  
//STEP1 EXEC PGM=IDCAMS  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
        DEFINE CLUSTER -  
            (NAME (DAWN.KSDS) -  
            STORAGECLASS (FAST) -  
            DATACLASS (KEYED) )  
/*  
***** Bottom of Data *****
```

### 1.11.3 Parameters of interest

Here we discuss different DEFINE parameters. However, we will focus on parameters that affect performance.

#### 1.11.3.1 CA size

The CA size for the data component is determined by the smaller value specified for *primary* and *secondary allocation* in the DEFINE command. However, CA size cannot be greater than one cylinder.

- Space allocations can be specified in cylinders, kilobytes, megabytes, records, and tracks. A kilobyte or megabyte will resolve to either tracks or cylinders. A record allocation resolves to tracks.

The following are the results when allocating a VSAM data set:

- If value < 1 cylinder, CA size equals track allocation.
- If value  $\geq$  1 cylinder, CA size equals one cylinder.

The CA size for index component is set to one DASD track.

#### 1.11.3.2 Free space options

Free space is the reserved space to be held free when the cluster is initially loaded or when a mass insert is done. The purpose is to allow data records to be inserted or expanded in length when updated. The FREESPACE parameter applies only to the data component of KSDS and VRRDS.

You can specify the amount of CI and CA free space when you define the data set using the IDCAMS DEFINE command. You can change the amount of free space using the IDCAMS ALTER command. Free space is specified as a percentage.

For a CI, VSAM takes the specified percentage times the actual CI size rounded down to a full byte. VSAM does not care about the record length.

The CA percentage specifies the amount of CIs to be held per CA. The number of empty CIs per CA is the number of CIs per CA times the CA percentage. The track required for the sequence set is not included in the calculation of CIs per CA, and the calculated figure is rounded down to the next integer. If the figure is less than one, it is the same as specifying 100% for the CA percent value.

If you specify 100 (the maximum value) for both the CI and CA percentage, each control interval will contain one record and each control area will contain one used control interval.

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 honored.

The syntax of the FREESPACE parameter is

```
FREESPACE(CI-percent CA-percent)
```

We recommend that you specify:

```
FREESPACE(20 20)
```

Specify CA free space to avoid CA splits. CI splits are not as time-consuming as CA splits.

#### **1.11.3.3 Share options**

The SHAREOPTIONS parameter specifies how the component or cluster can be shared among users within one system or across systems. Refer to 4.2.4, “Cross-region options” on page 188 and 4.2.5, “Cross-system options” on page 189 for more details.

SMS-managed volumes and catalogs containing SMS-managed data sets, must not be shared with non-SMS systems.

The syntax for the SHAREOPTIONS parameter is

```
SHAREOPTIONS(crossregion crosssystem)
```

#### **1.11.3.4 IMBED and REPLICATE**

The IMBED parameter requires that the sequence set is written as many times as it can fit in the first track of each data CA.

With the REPLICATE parameter, each index record at all levels is written as many times as it can fit in a track.

#### **NOTE:**

Support for IMBED parameter has been dropped with DFSMS/MVS V1R5. If specified, it will be ignored and no message will be issued.

---

## Chapter 2. Performance

How can you get the most out of your VSAM data sets? How can you improve data storage and retrieval? What parameters can be used when you define a VSAM data set to enhance data access?

This chapter describes the VSAM functions that can enhance performance. Hints and tips are provided to help you implement these VSAM functions.

Before we discuss VSAM performance, we will go over some performance basics.

---

### 2.1 Service level agreement (SLA)

The human perception of the performance of a system is subjective, emotional, not precise, and related to throughput (transactions per second), response time, and transaction distribution.

In order to make it more objective and related to business needs, the concept of SLA was introduced.

SLA is a type of contract between Information Systems and user departments that objectively describes:

- Average transaction *response time* (Tr) for:
  - Network
  - Input/output (I/O)
  - CPU
  - Total resources
- The distribution of the response times (a measurement about how erratic they are)
- The throughput, also called external throughput rate (ETR), measured in ended transactions per second of elapsed time (not CPU time)
- System availability; the percentage of time that the system is available to the end user

## 2.2 Transaction performance

A transaction is a business unit of work produced by an online or batch interaction with an end user. It can be a CICS, a TSO, a WEB, an APPC, a DRDA, or even a batch interaction. If you are in workload manager (WLM) goal mode, all these transactions are monitored and accounted by OS/390.

To characterize the performance of a transaction, we need to understand its different response time components. Figure 14 shows general response time components, where:

- $ETR = Ne / T$

ETR is the External Throughput Rate, Ne is the number of ended transactions, T is the elapsed time

- $Tr = Ts + Tw$

Ts is Service Time, Tw is Waiting Time, and Tr is Response Time

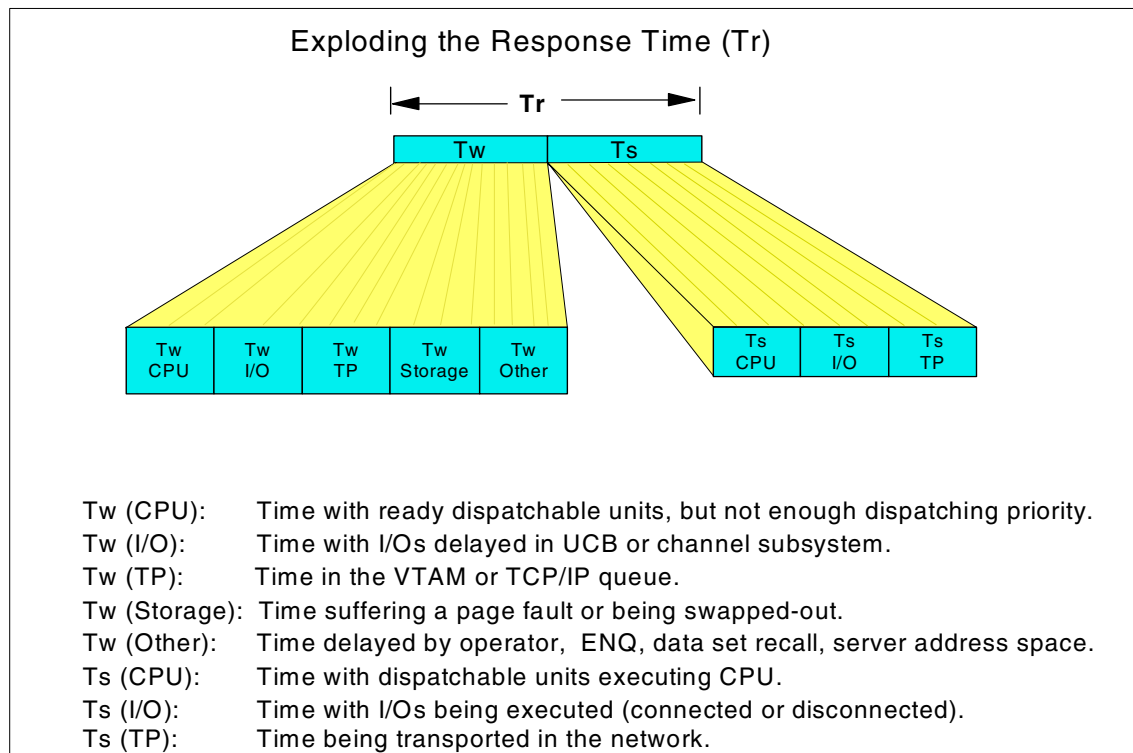


Figure 14. Response time components

- Exploding the above formula, we have:

$$T_s = T_s(\text{CPU}) + T_s(\text{IO}) + T_s(\text{TP})$$

$$T_w = T_w(\text{CPU}) + T_w(\text{IO}) + T_w(\text{TP}) + T_w(\text{Storage}) + T_w(\text{Other})$$

- There is also one formula relating the  $T_r$  with ETR, derived by Little's law:

$$\text{ETR} = N / (T_t + T_r)$$

In this formula,  $N$  is the average number of users sending transactions, and  $T_t$  is the average thinking time of these users. Following are some considerations regarding this formula:

- The variables that more intensively affect the ETR are  $N$  and  $T_t$ . Therefore, you should never accept an SLA specifying ETR, because the only variable that the IS department can directly control is  $T_r$ .
- However, experiences show that when  $T_r$  is in a sub-second value, the value of  $T_t$  drops dramatically. This fact has to do with the human behavior in front of a terminal. If the machine responds fast (in a sub-second value), we also respond fast. There is some work done by Arvind J. Thadhani showing this experience.

Transaction response time is the best way to characterize the performance of a transaction. Here, the target is to reduce its value and consequently to increase the ETR figures (when the value is below one second). Remember that in RMF reports the  $T_s(\text{TP})$  and  $T_w(\text{TP})$  are not included.

---

## 2.3 Performance management

Performance management is the activity in an installation that monitors and allocates data processing resources to applications according to a service level agreement (SLA).

There are three main ways to solve performance problems:

- **Buy:** You can simply buy more resources.
- **Tune:** You can create the *illusion* you bought more resources. This is known as *tuning*. Capability to do this implies that previously, you had been wasting resources, and you are now making full use of them. As with purchasing, there is a cost in people resources to tune. While it may be higher, this cost is always less visible than when purchasing.
- **Steal:** You can “steal” the resources from a less critical application. Here the cost is in lower service to the application from which the resources were stolen.

### 2.3.1 I/O performance

As CPU speeds increase, the I/O time is the determinant factor in the response time. So, you can get excellent response time returns by reducing I/O wait time and I/O service time. However, experience has shown that when you increase the I/O rate (for example, for data striping), suddenly the bottleneck is moved from the I/O subsystem to the processor.

Generally speaking, you can reduce I/O service time by software or hardware techniques. The software techniques are:

- Virtual address space and data space buffers
- Hiperspace buffers
- Data compression

The hardware techniques are:

- Faster channels (FICON, ESCON) to the processor
- Faster device paths (adapters) to the controllers
- Larger controller cache
- More DASD subsystem concurrency, for example, parallel access volumes (PAV) in Enterprise Storage Server (ESS).

---

## 2.4 VSAM performance management

The goal of VSAM performance management is to decrease the values of I/O wait time ( $T_w(\text{IO})$ ) and I/O service time ( $T_s(\text{IO})$ ) — that is, the I/O response time ( $T_r(\text{IO})$ ), for the transactions accessing VSAM data. The concentration is on transactions most closely linked to your business needs.

In this performance chapter, we use two approaches in order to help you to improve VSAM performance. First, we discuss all the VSAM external parameters whose values may affect performance. We give recommendations based on experience, on how those parameters should be set. This will be known as “rule-of-thumb” (ROT) mode, more oriented to help an end user.

In the second approach, we simulate a scenario (as real as possible), where one or several VSAM data sets are causing performance problems to key transactions in a real installation. It is more oriented to help a system programmer. We follow a methodology to fix the situation. Here some of the recommendations covered in the ROTs are presented again, now connected with a specific system behavior.



Also, other factors not connected to VSAM parameters — such as I/O configuration delays, SMS storage class attributes, use of Hiperbatch, and heavy CPU usages — are introduced and discussed.

Throughout this chapter, you see a set of screens containing real RMF reports covering VSAM I/O measurements related to the theories presented. These measurements were obtained in our lab setup described in B.1.1, “General lab description” on page 223. Have a look, when you can.

---

## 2.5 VSAM rule-of-thumb (ROT) mode

Before we start, let us make a general statement about VSAM cluster parameters affecting performance and their defaults. You should note that these defaults were established a long time ago, when virtual storage was all below the 16-MB line; central storage restricted; the DASD was slow, removable, and expensive; and channels were a scarce resource.

Therefore, the defaults are outdated in many cases. The reason IBM does not change the defaults is to maintain compatibility with your existing workload. Remember that compatibility is an important feature protecting your investment. You do not need to throw out your code and hardware when OS/390 changes a release. However, keep in mind that these defaults can be easily updated through JCL, the ACB macro, data class constructs, ACS routines, or IDCAMS.

### 2.5.1 Invalid rules-of-thumb (IROTs)

Another important point to mention is that you may have heard many recommendations about data set placement and the effect on I/O performance. We will call these recommendations “invalid rules-of-thumb” (IROTs). Generally speaking, they are out-of-date due to the introduction of the RAID controllers and enhancements to channel programs, such as these:

- The 3390/3380 volume concept no longer exists, and we do not know where their logical tracks are located on the disks, so there are no seek arm movement considerations.

Also, in the past we had several changes in the device geometry as new products were introduced. Now the logical 3390/3380 geometry will still be valid for some time because it is not affected by enhancements to the disk controllers. The device independence recommendation (for units of allocation, for example), due to changes in future track or cylinder geometry, is not a consideration.

- In the case of a cache miss, the channel always reads (or writes) in cache, asynchronously in relation to the disk, so there are no extra revolutions.
- The existence of the Define Extent CCW and the Format CCW for the Enterprise Storage Server (ESS).

The following recommendations (ILOTS), are *no longer valid*:

- Place your VTOC around the middle of the volume.
- If you need to place two active data sets in the same 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 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 3390/3380 geometry changes.
- Reorganize your KSDS data set after some CA splits in order to avoid long embedded seeks. (For information on VSAM KSDS data set reorganization refer to 4.1, "Reorganization considerations" on page 181.)
- Avoid channel utilization above 30% to inhibit RPS misses and another DASD revolution.
- Use VSAM keyrange to have control over the allocation of the key ranges of your data set.

---

## 2.6 Parameters affecting performance

The following VSAM parameters and options can affect performance.

### 2.6.1 Allocation units

When you define your new data set, either the end user or SMS must specify the amount of space to be allocated for it. One of the allocation functions, the creation of the DSCB in the VTOC, is done by the DADSM routine. If SMS is active, you can specify a data class and take advantage of the space allocation set by your storage administrator. If you choose to specify space

explicitly, you can specify it for VSAM data sets in units of records, kilobytes, megabytes, tracks, or cylinders.

If you specify records as the allocation unit, the number of records you declare is multiplied by RECORDSIZE(AVERAGE) value, to derive the space in bytes. This keyword can indicate the maximum value allowed for the record length. If the maximum record length is exceeded, VSAM rejects the new record.

Note that VSAM data sets cataloged in an integrated catalog facility (ICF) catalog are allocated with the CONTIG attribute. If the allocation unit is TRACKS, the primary and secondary allocations are in contiguous tracks. This may cause unexpected allocation abends.

DADSM only accepts allocation requests in tracks or cylinders. The units specified (records, kilobytes, or megabytes) are converted by IDCAMS before the DADSM request. The amount of space you allocate should depend on the size of your data set and the index options you have chosen.

Usually you declare a primary and a secondary amount of space for allocation purposes. Do not define too small an amount of secondary space allocation value, especially for a KSDS or a VRRDS data set. There are a large number of I/O operations involved when the secondary allocation takes place.

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. Refer to 4.5.3, “End-of-Volume (EOV) macro” on page 202, for more details. VSAM acquires space in increments of control areas (CAs). Each time a new record does not fit in the allocated space, EOVS allocates more space in the secondary space amount. This can be repeated until the volume is out of space or the extent limit is reached. Depending on the type of data set allocation request, a new volume may be used.

#### **2.6.1.1 Guaranteed Space**

The allocation function depends on whether the data set has the Guaranteed Space attribute.

The Guaranteed Space storage class SMS attribute lets you reserve space on specific volumes for data sets that require special placement to meet performance or availability requirements. For example, IMS logs that are duplexed by IMS to improve availability should be on separate volumes.

Typically, you use storage class performance and availability attributes and storage group assignments to determine where to place your data set. You assign storage groups in your storage group ACS routine. SMS then selects volumes by evaluating each candidate's ability to satisfy the performance, availability, and space requirements for the data set. To place data sets on specific volumes, 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 is unsuccessful.

For non-guaranteed space data set allocation, when you allocate space, it is possible for the user to specify whether to use primary or secondary allocation amounts when extending to a new volume. This is done with an SMS DATACLASS parameter for VSAM attributes, as pictured in the following screen of the ISMF application.

```
ADDITIONAL  The ADDITIONAL VOLUME AMT field shows the type of allocation
VOLUME AMT  amount when a VSAM data set in extended format begins allocation
---(15)---  on subsequent new volumes.
```

Possible values:

```
PRIMARY     Primary allocation amount has been requested.

SECONDARY   Secondary allocation amount has been requested.

-----     If the value has not been specified.  The system will use
              the default value of Primary.
```

For a Guaranteed Space data set allocation, the following conditions must met:

- All volumes specified belong to the same storage group.
- The storage group to which these volumes belong is in the list of storage groups selected by the ACS routines for this allocation.

It is recommended to allocate space at the data component level, because this is the component that you are able to size. VSAM allocates space as follows:

- If allocation is specified at the cluster or alternate index level only, the amount needed for the index is subtracted from the specified amount. The remainder of the specified amount is assigned to data.

- If allocation is specified at the data component level only, the specified amount is assigned to data. The amount 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 and the specified index amount is assigned to the index.
- If secondary allocation is specified at the data level, secondary allocation must be specified at the index level or the cluster level.

### 2.6.1.2 Optimizing control area (CA) size

Before we discuss this topic, we want to clarify that when we say cylinder and track, we are referring to the logical 3390/3380 cylinder and track, not the cylinder and track of the disks in the RAID DASD controllers.

Regarding optimizing the control area (CA) size, there is no way to explicitly specify this. Generally, the primary and secondary space allocation amounts determine the CA size, as follows:

- If either the primary or secondary allocation is smaller than one cylinder, the smaller value 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. An exception is for data striping, where the CA size can be 16 tracks (one more than the cylinder) — the maximum size for a CA.
  - CYLINDERS(5,10): This results in a 1-cylinder CA size.

The index CI size and buffer space can also affect the CA size. The previous examples assume the index CI is large enough to handle all the data CIs in the CA, and the buffer space is large enough 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 which is not large enough to contain the largest spanned record.

**Note:** If space is allocated in kilobytes, megabytes, or records, VSAM sets the CA size equal to multiples of the minimum number of tracks or cylinders required to contain the specified kilobytes, megabytes, or records. Space is not assigned in units of bytes or records.

If the CA is smaller than a cylinder, its size is an integral multiple of tracks, and it can span cylinders. However, a CA can never span an extent of a data set, which is always composed of a whole number of CAs.

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 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 records. The sequence set record for a CA is always read for you. Fewer records means less time spent reading them.
- If you have allocated 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 overlap between I/O and CPU for sequential processing is done in a CA boundary. When reached the application must wait until the last input/output to the CA is done before proceeding to the next CA. The I/O operations are always scheduled within CA boundaries.

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 ties up more real storage and more buffers.

#### **2.6.1.3 Partial release**

Partial release is used to release data (not index) unused space from the end of an *extended format* data set. Refer to 1.5, “Extended format data set” on page 18. Partial release is specified through the SMS management class or by the JCL RLSE subparameter.

All space after the high used RBA (HURBA) is released on a CA boundary up to the high allocated RBA (HARBA). Refer to 3.9, “IDCAMS LISTCAT output fields” on page 166, to get more information on HURBA and 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 restrictions are:

- Alternate indexes (AIXs) opened for path or upgrade processing are not eligible for partial release. The data component of an AIX when opened as cluster could be eligible for partial release.
- Partial release processing is not supported for temporary close.
- Partial release processing is not supported for data sets defined with guaranteed space.
- Extended format is a requirement.

#### **2.6.1.4 Allocation constraint relief**

Users occasionally encounter data set allocation or extension failures (the X37 equivalent type of abends) because there is not enough space available on a volume to satisfy the request. SMS alleviates this situation to some extent by performing volume selection, 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 retried, if it fails due to space constraints. Refer to 3.4, “Space Constraint Relief parameter (fewer X'037' abends)” on page 131.

Recommendations:

- Because cylinders and tracks are units of space not direct linked to your application planned DASD capacity, you should specify records, kilobytes, or megabytes.
- Do not define too small an amount of secondary allocation for a KSDS/VRRDS data set. There are a large number of I/O operations involved when the secondary allocation takes place.
- It is recommended to allocate space at the cluster or data levels.
- Do not use tracks as an allocation unit; this forces the CONTIG attribute for secondary allocations.
- Avoid either primary or secondary allocation smaller than one cylinder, it makes the CA size less than one cylinder.
- Consider partial release, if applicable.
- Consider constraint relief, if applicable.

## 2.6.2 Buffer space

You specify the BUFFERSPACE value at IDCAMS DEFINE time. The BUFFERSPACE value in the catalog entry for the data set is the *minimum* amount of buffer space, and it applies to the whole cluster. It helps VSAM to determine the CI size of the data components and index components. For more details; refer to 2.6.9.1, “How BUFFERSPACE and BUFSP affect buffering” on page 60.

Recommendations:

- Do not specify BUFFERSPACE. Let it default.

## 2.6.3 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 only have data CIs. Refer to 1.3.3, “Control interval” on page 3. The data CI and index CI sizes are declared in the IDCAMS DEFINE command and kept in the catalog. In the following sections we offer recommendations about how to size both.

### 2.6.3.1 Data control interval

There are arguments favoring both large and small CI sizes. Let us look at each of these possibilities:

- For sequential processing, larger data CI sizes are desirable. For example, given a 16 KB data buffer space, it is better to read two 8 KB CIs with one I/O operation than four 4 KB CIs with two I/O operations.

Another good point about a large CI is a more centralized management of the free space CI, and consequently fewer CI splits.

- For direct processing, smaller data CIs are desirable because you are only retrieving one logical record at a time, then avoiding useless data transfer.
- For data set shared access within an address space refer to 4.2.3, “Intra-address space sharing” on page 185, the size of the CI affects the amount of data locked by VSAM; the smaller, the better.
- If you intend to use hiperspace ESO (in LSR) for a second level of buffering, if the CI size of the data component is not a multiple of 4k, both virtual space and hiperspace is wasted.

In conclusion, for data sets accessed both randomly and sequentially, a small data CI with multiple buffers for sequential processing can be a good choice.



### 2.6.3.2 Index control interval

For KSDS and VRRDS, specify CI size with the data component only; in this case VSAM calculates the index CI size value. If you define CI size for the cluster, this value is used for both data CI and index CI. It is better to let VSAM calculate the index CI size (which is generally smaller than data CIs). The size of the index CI affects:

- The number of levels in the index set. However, the number of indexes affects the performance only for very large clusters.
- The usable capacity of a data CA. Sometimes a small index CI size may cause some loss in the usable capacity of the associated data CA. For the calculation of the CI index size, VSAM assumes a compressed key of 5 bytes. Sometimes the key does not compress well and this assumption is underestimated. To determine if your CI size is too small and results in losing data CA space; refer to 4.1, “Reorganization considerations” on page 181.

Use your automation console to detect the following message, generating an alert:

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

There is no external parameter to use to define the CA size. However, you may influence the size of the CA when you allocate a new data set. Refer to 2.6.1, “Allocation units” on page 42 for more details. Also refer to 2.8, “VSAM and SmartBatch” on page 121, to see how the CI size can be determined when you use SmartBatch.

Recommendations:

- For sequential access, define data CIs with 16 KB or larger.
- For random access, define data CIs with 4 KB.
- For mixed random and sequential access, define data CIs with 4 KB (for random access) and plenty of buffer space in the buffer pool (allowing CCW chaining for sequential access).
- Let VSAM derive the size of the index CI. Define it yourself only when you are losing data CA capacity.

## 2.6.4 Free space

Free space specifies, through IDCAMS DEFINE (in the catalog), the percentage of each data CI and data CA is to be set aside as free space when the cluster KSDS or VRRDS is initially loaded or when a mass insertion is done. There is no free space external specification for index CIs or CAs.

Free space is used to reduce the number of CI and CA splits along insertions or updating in-place records with a length increase. Refer to 1.3.5, “Splits” on page 4. A CA split causes considerable overhead (during the split), since approximately half of the CIs from the CA are moved to the end of the data set. Performance is enhanced when CA splits are reduced. However, there is no concern (with the exception of some sequential processing remarks) about any negative aspect of the spreading of data caused by the split. The performance problem has to do with the split itself.

When you specify free space, ensure that the CI free space percentage results in enough free space to hold at least one logical record. You can ensure this by taking into account the logical record length, the size of the CI, and the length of CIDE and RDEs. Following is a numeric example:

- Supposing a fixed length record data set, where each CI is 4096 bytes, 10 bytes are reserved for control information (2 RDEs and 1 CIDE). Each logical record has 1000 bytes and you want to reserve room for 10% inclusion.
- If you specify 10% of CI free space VSAM (record management) puts aside 410 bytes ( $4096 * 0.10$ ) for free space. The logical record space threshold is 3676 ( $4096 - 420$ ) bytes. The space between the threshold and the control information is reserved as free space.
- 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. There are 86 bytes of unused space. All this calculation is done for a non-compressed data set. For compression information refer to 2.6.10, “Data compression” on page 84.

For CA free space, VSAM ensures that at least one CI per CA remains empty during loading when you declare a CA FREESPACE amount other than zero.

Too much free space in a CI or CA can result in:

- Increased number of index levels, which affects run times for direct processing slightly.
- More DASD storage required to contain the data set.

- More I/O operations required to sequentially process the same number of records. Note that 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. Direct access data transfer for the data component is not affected by CI free space.

Too little free space can result in an excessive number of CI and CA splits, with consequences such as:

- The CA splits are time consuming, due to the overhead (during the split), since approximately half of the CIs from the CA are moved to the end of the data set.
- CI and CA splits may also affect the sequential processing because the DASD controller only detects logical 3390/3380 physical sequence. Splits make the logical sequence different from the physical sequence, and the controller only detects the physical sequence pattern.

Determine the amount of CI free space based on the percentage of record additions expected, and their distribution.

VSAM offers two insert record techniques for a split:

- Split CIs and CAs at the insert point (SIS).
- Split at the midpoint (NIS, default) when doing direct PUTs.

If you know in advance the pattern of the keys being inserted, you can take advantage of it by choosing the technique best suited for the application.

Recommendations:

- Determine the amount of CI free space based on the percentage of record additions expected, 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 additional insertions.

If the few records to be added are fairly evenly distributed, CI free space should be equal to the percentage of records to be added (FSPC (nn 0), where nn equals the percentage of records to be added.)

*Evenly distributed additions.* If new records will be evenly distributed throughout the data set, CA free space should equal the percentage of records to be added to the data set after the data set is loaded. (FSPC (0 nn), where nn equals the percentage of records to be added.)

*Unevenly distributed additions.* If new records will be unevenly distributed throughout the data set, specify a small amount of free space. 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, you can take full advantage of mass insertion by using the ALTER command to change free space to (0 0) after the data set is loaded.

*Additions to a specific part of the data set.* If new records will be added to only a specific part of the data set, load those parts where additions will not occur with a free space of (0 0). Then, alter the specification to (n n) and load those specific parts of the data set.

- Choose, if possible between insert point or midpoint techniques.

### 2.6.5 Index options

There are two index options associated with a cluster defined in the IDCAMS DEFINE command and stored in the catalog:

- REPLICATE, where each index record is replicated (written on a track of a 3390/3380 DASD volume) as many times as it fits aims to reduce rotational delay in retrieving the index record.

Due to the fact that the 3390/3380 volume does not exist in today's DASD controllers and we do not know where their logical tracks are located on the disks, there are no seek arm movement considerations. The use of the REPLICATE option is not recommended. Refer to 2.5, "VSAM rule-of-thumb (ROT) mode" on page 41, for more information about the changes in the performance aspects due to the implementation of the new RAID controllers.

- IMBED, where the index sequence set (the lower level index) is placed adjacent to the corresponding data CA.

Recommendations:

- Do not use REPLICATE, take the default of NOREPLICATE. Beginning with DFSMS 1.5 this parameter is no longer valid. No warning message is issued.
- Do not use IMBED. Beginning with DFSMS/MVS V1.5 this parameter is no longer valid. No warning message is issued.

### 2.6.6 Share options

There are four important mechanisms to implement VSAM data set integrity, they are:

- VSAM SHAREOPTIONS play an important role in VSAM integrity, specifying whether and to what extent data is to be shared among tasks in one or multiple OS/390 address spaces.
- ENQ/reserve serialization functions.
- JCL disposition (OLD or SHR), which also issues an ENQ.
- Record level sharing (RLS) locking mechanism, which is not covered in this document.

The reason that we are covering these mechanisms in the performance discussion is because usually integrity and performance vary inversely. Total integrity may mean bad performance.

When you define VSAM data sets, you can specify how the data is to be shared within a single system or among multiple systems that can have access to your data. Before you define the level of sharing for a data set, you must evaluate the consequences of reading incorrect data (a loss of read integrity) and writing incorrect data (a loss of write integrity), situations that can result when one or more of the data set's users do not adhere to guidelines recommended for accessing shared data sets. On the other hand, it is important to avoid the unnecessary use of certain serialization functions which may cause a performance degradation.

Refer to 4.2, "Sharing VSAM data sets" on page 183, for more information.

Recommendations:

- If you are sure that no application updates or deletes the VSAM data set, then do not use SHAREOPTIONS 4. It causes bufferpool refresh for direct reads or writes. The bufferpool 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-system environment, avoid the use of the RESERVE macro, which locks the full 3390/3380 logical volume. Use instead the ENQ macro for a GRS global ENQ resource.

### 2.6.7 Initial load option

Initial load mode happens when you load your data sequentially in a VSAM data set. This data set is already IDCAMS defined (cataloged and described through the DSCB in VTOC) and high used RBA (HURBA) equal to zero. This is the first step after the DEFINE or a when data set, defined with REUSE attribute, is open with MACRF=RST (reset) in the ACB.

Initial load of your data set means either that this is the first time you are writing its initial contents, or that you reusing the data set with new data. You can do this through IDCAMS REPRO, IDCAMS IMPORT, or by writing your own program.

Following are recommendations and remarks about initial loading of a VSAM data set after it is defined.

There are two options where performance is concerned, SPEED and RECOVERY.

- RECOVERY initially pre-formats (with Write format CCWs) each CA with zeroed CIs (VSAM end-of-file mark). In a second step it fills this data portion (with Write modified CCW), of the pre-formatted CA with real data. So, the I/O response time for the load almost doubles. Read 2.7.5.1, “Cache highlights” on page 113, to get more information on Write format and Write modified.

The RECOVERY option is suppose to allow the restart of the load operation from the last written record, however other difficulties such as tape repositioning (in the input file) have inhibited customers use of it.

- SPEED does not pre-format. It issues Write format CCWs with real data. As in the Write modified type of channel program (in RECOVERY), many writes are assembled in the same channel program to improve CPU and I/O usage. However free CIs are not written together with occupied CIs in the same channel program.

Be aware that during load mode processing, you cannot share data sets. Share options are overridden during load mode processing to (1 3). 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. If the data set is shared between systems, VSAM does nothing to ensure that another system is not accessing the data set concurrently. The user must ensure that another system is not accessing the data set.

There are other VSAM options affecting the initial load performance:

- Use system managed buffering (SMB) due to better buffer management. Refer to 2.6.9.7, “System managed buffering (SMB)” on page 77.
- Use extended format; refer to 1.5, “Extended format data set” on page 18.

Recommendations:

- Never use the RECOVERY option.
- Use SMB.
- Use extended format.
- Any specific SHARE options assignment in a load is defaulted to (1 3)

#### 2.6.7.1 Write checks

Due to the new RAID controllers, the write check option is useless and introduces performance degradation.

Recommendations:

- Never use the write check option.

### 2.6.8 Region size

The key to reducing the number of I/O operations is to keep more data in virtual storage. The recommendations given here go in that direction. So, before you implement them, you should review the region size parameter to avoid an S878 type of abend, as shown in the following VSAM message:

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

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* (located from the bottom to top of the private area). The *region size* is the amount of storage in the *user region* available to the job, started task, or TSO/E user. Figure 15 shows the virtual storage layout. For a description of each area; refer to the *OS/390 MVS Initialization and Tuning Guide*, SC28-1751.

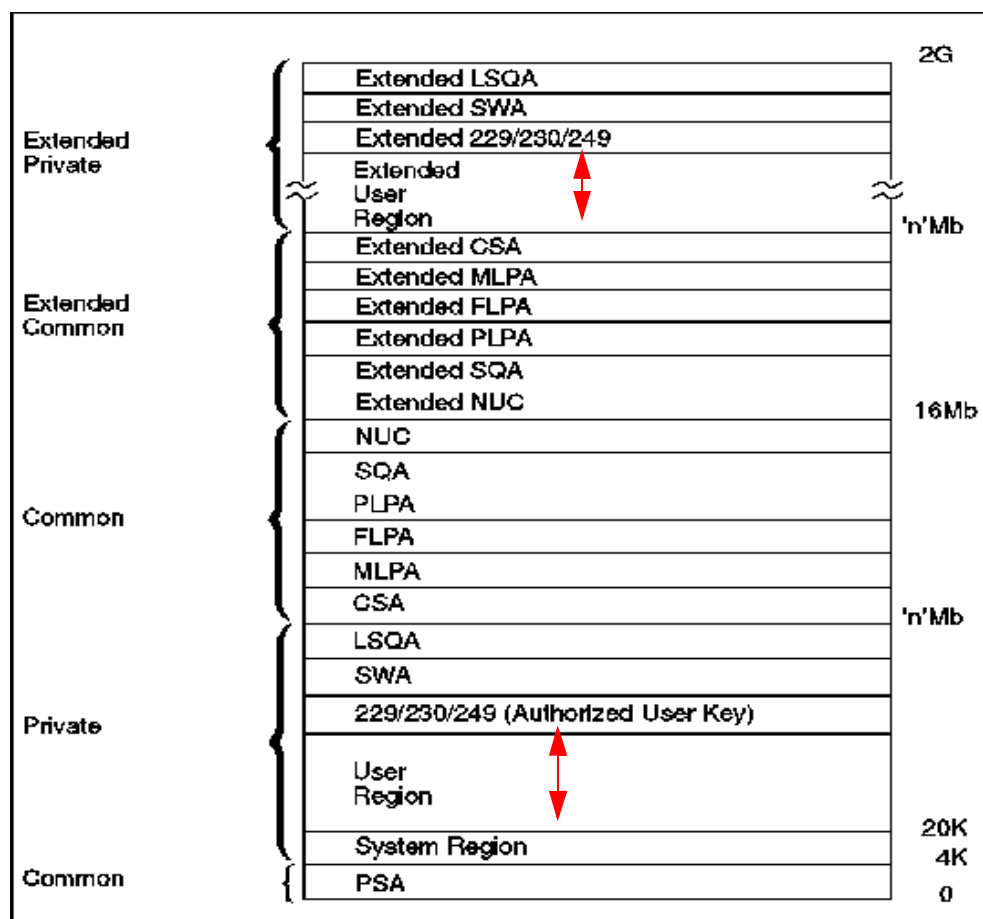


Figure 15. Address space layout

You specify a job's region size by coding the REGION parameter in the JOB or EXEC statement. The system rounds all region sizes to a 4K multiple. Some installations use the IEFUSI exit to modify or limit the region size.

Following is an explanation about the need for region size; refer to Figure 15. This explanation describes the private area below 16 MB. The same considerations apply to the area above the 16 MB line.

- The size of the *private area* is determined by the difference in 16 MB minus the common area below, rounded to a 1 MB boundary. The rounding is because a segment table (which covers 1 MB of virtual storage), cannot have one piece with common addresses and another piece with private addresses.



- Depending on the type (subpool number) of the GETMAIN, the virtual addresses GETMAINED can be taken from top to bottom (for system subpools like LSQA, SWA, 229, 230) or from bottom to top (subpools for loading programs and user virtual work areas).
- Some programs issue a GETMAIN requiring more private virtual storage (from bottom to top) than they really need. The surplus storage is used for performance gains. The programs would sometimes abend because there was no room for GETMAINS from top to bottom. To control abends and excessive paging the concept of region was introduced.
- Jobs run out of virtual space and abend when:
  - The REGION specified is greater than the available private area.
  - A private area GETMAIN reaches the limit determined by the IEFUSI exit (the default is the specified REGION plus 64k) even if there is virtual free space available.
  - A private area GETMAIN cannot be served because no contiguous virtual free space exists. This means a collision between the top-to-bottom and bottom-to-top subpools.

Table 2 shows how the JCL REGION parameter is interpreted and how much virtual storage is available, below and above 16 MB, for each step of a job.

Do not hesitate to increase your job region size. We will show how good buffering can reduce the number of I/Os, job elapsed time, CPU time and device connect, and disconnect time. If your job is I/O bound, then by giving it enough resources, it executes more quickly. That is good both for the user and for total system performance.

Refer to 2.8, “VSAM and SmartBatch” on page 121, to see how this product can help you to avoid virtual storage abends.

*Table 2. Region JCL parameter*

REGION value	Region available below 16 MB	Region available above 16 MB
<b>0k &gt; REGION &lt; 16 MB</b>	Establishes the private area size below	32 MB
<b>16 MB &gt; REGION =&lt; 32M</b>	All private area below is available	32 MB
<b>32 MB &gt; REGION =&lt; 2G</b>	All private area below is available	Establishes the private area size available above 16 MB
<b>0k or 0M</b>	All private area below is available	All private area above is available

If your job is experiencing constraints in storage below 16 MB when VSAM files are opened, you can relieve storage usage below 16 MB by specifying that VSAM allocate the buffers and/or control blocks above 16 MB. For details; refer to 2.6.9.6, “Locating VSAM buffers above 16 MB” on page 76.

### 2.6.9 Buffering options

Buffering in virtual storage is an important technique in VSAM in order to improve performance. You have the possibility of using this technique through the buffering options. Here we cover the aspects of buffering in virtual storage.

A VSAM resource pool is a set of VSAM control blocks plus a buffer pool. A buffer pool is a collection of same size I/O buffers plus control information describing the occupancy of such buffers. The objective of a buffer pool is:

- To avoid I/O operations in random access due to control intervals hits.
- To synchronize differences in speed from I/O to processor during sequential processing, and consequently to improve performance.

For more efficient use of virtual storage, buffer pools can be shared among *data sets* using locally or globally shared buffer pools. There are four types of resource pools, according to the technique used to manage them:

- Non-Shared Resource (NSR)
- Local Shared Resource (LSR)
- Global Shared Resource (GSR)
- Record Level Shared (RLS)

You will see details about each one in this chapter, except for RLS. The exploiter of RLS is CICS. For details, refer to *DFSMS/MVS Using Data Sets*, SC26-4922; and *CICS and VSAM Record Level Sharing: Implementation Guide*, SG24-4766.

Buffers are acquired dynamically, and only when the data set is opened. The amount of space for buffers is based on parameters in effect when the program opens the data set.

One of the major sources for determining how much space should be allocated in a buffer pool is the Access Control Block (ACB). The ACB is created in the program to:

- Identify the data set to be opened.
- Specify the type of processing to be done with the data set.
- Specify the basic options.

- Indicate whether user exit routines are to be used while the data set is being processed.

The system obtains ACB information for VSAM data sets as follows:

- Information in DD statements overrides SMS data class information.
- The program's ACB information is used.
- The catalog entry for the data set is used.
- For LSR buffering, the BUFND, BUFNI, BUFFERSPACE and STRNO parameters do not apply and cannot be overridden by JCL. For details, see 2.6.9.3, "Local Shared Resources (LSR)" on page 71.

Parameters that influence the buffer allocation are shown in Table 3.

You can see below how specifications in the MACRF, ACB's parameter, affects 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 technique to be used: NSR (default), LSR, GSR or RLS.
- The manner in which the records are intended to be accessed: Direct (DIR), sequential (SEQ), skip sequential (SKP).
- Type of argument used to access records: By key (KEY option), by RBA (ADR option), or access is to the entire contents of a control interval rather than to an individual data record (CNV). KEY is the default.
- What kind of processing is done in the data set: Input (IN, default) or output (OUT); this is just an intention.
- How writes are to be managed: defer writes (DFR) or not (NDF). For details about deferring writes; refer to 2.6.9.5, "Deferring write requests" on page 75.
- Using LSR, how VSAM deals with conflict for an exclusive control in buffers: VSAM defers the request until the resource becomes available (LEW, default) or VSAM returns the exclusive control return code X'14' to the application program (NLW). 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, default) when doing direct PUTs.

In MACRF, you 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, you use request parameter lists (RPL), where you specify only the processing options appropriate to 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, NSR buffering management.

Table 3. Parameters affecting buffer allocation

SOURCE	PARAMETER
DD Statement	AMP parameter: BUFSP, BUFND, and BUFNI
SMS data class (ACDS)	RECORD ACCESS BIAS
Program's ACB	MACRF=(INIOUT, SEQISKP, DIR)
	STRNO=n, BUFSP=n, BUFND=n, BUFNI=n (1)
	SHRPOOL=n (2)
The catalog entry for the data set	BUFFERSPACE
<b>Note:</b> (1) Applies only to NSR. (2) Only for LSR/GSR, connect the ACB to an existent resource pool	

#### 2.6.9.1 How BUFFERSPACE and BUFSP affect buffering

The BUFFERSPACE value in the catalog entry for the data set is the *minimum* amount of buffer space while the value assigned in 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:

- 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 used for buffers is the *largest* of these values:

- The amount specified in the catalog (BUFFERSPACE)
- The amount determined from BUFND and BUFNI
- The minimum storage required to process the data set with its specified processing options

The BUFSP value takes precedence over BUFNI and BUFND as follows:

- If the number of buffers specified in the BUFND and BUFNI subparameters exceed the virtual storage specified in the BUFSP space, the number of buffers is decreased to fit in the BUFSP space as follows:
  - If the ACB indicates direct access only, first the number of data buffers is decreased until it reaches the BUFSP value, but it never becomes less than the minimum required. If the BUFSP value is not reached, then 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 not reached, BUFND is decreased. When the minimum is reached, but BUFSP is not reached, then 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, additional index buffers are allocated.
  - For sequential access, one additional index is allocated 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 to process the data set, VSAM cannot open the data set.

Recommendation:

- Do not specify BUFSP. This avoids mistakes in calculation leading to different results from those expected.

When a buffer's contents are written, using direct access, the buffer's 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 if the desired control interval is in storage. This is not valid for share options 4, where buffers used for direct processing are refreshed for each request.

Buffer space is released when all data sets that are using the buffer pool are closed. If an abend occurs before closing VSAM data sets, the buffers are not flushed by VSAM or Recovery Termination Manager routines. It is left to the application decision throughout the use of an (E)SPIE/(E)STAE routine to close the data sets and consequently flush the buffers. For details about ESTAE and ESPIE macros, refer to *OS/390 MVS Programming: Authorized Assembler Services Reference, Volume 2 (ENFREQ-IXGWRITE)*, GC28-1765.

When processing a data set using a path, the number of needed buffers increase, since 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. For more details, refer to *DFSMS/MVS Using Data Sets*, SC26-4922.

Choosing the adequate VSAM buffer length 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 might cause excessive paging or excessive internal processing. There is an optimum point at which more buffers do not decrease the job elapsed time and device connect time. You can see that in Table 4 on page 66. You should attempt to have data available just before it is to be used. If data is read into buffers too far ahead of its use in the program, it can be paged out.

The default for the number of VSAM allocation buffers is as follows:

- For the index component: BUFNI=STRNO
- For the data buffers: BUFND=STRNO+1

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.

This default is also the *minimum* number of buffers required by VSAM.

We will now look at VSAM buffering techniques.

#### **2.6.9.2 Non-shared resources (NSR)**

NSR is the default VSAM buffering technique. It has the following characteristics:

- The buffers are *not* shared among VSAM data sets.
- The buffers are located in the private area.
- It is suited for sequential processing, because the buffers are managed via a sequential algorithm:

- For sequential processing, there is look-ahead, and CIs in the buffer pool are not managed by LRU (meaning after use, they are strong candidates to leave the buffer pool).
- For direct processing, there is no look-ahead, but CIs are not managed by LRU.
- It is used by high level languages.
- The resource pool (buffer pool plus control blocks) is built automatically by OPEN.
- The data insert buffer is used only for CI splits.
- Each string has its own buffer for the index component. Only the additional index buffers provided are:
  - Used to cache index set records
  - Shared among strings
- Each string has its own buffer for the data component. If additional data buffers are provided, they are for:
  - Sequential READS and WRITES
  - CA splits
  - Spanned records
- Dynamically extends the number of strings as they are needed by concurrent requests for the ACB.
- For subtask sharing, when a CI is not available for the type of task processing requested, VSAM under NSR buffering has a proper way of managing the contention. For details, refer to 2.6.6, "Share options" on page 53.

Figure 16 shows how buffers are searched.

# VSAM NSR

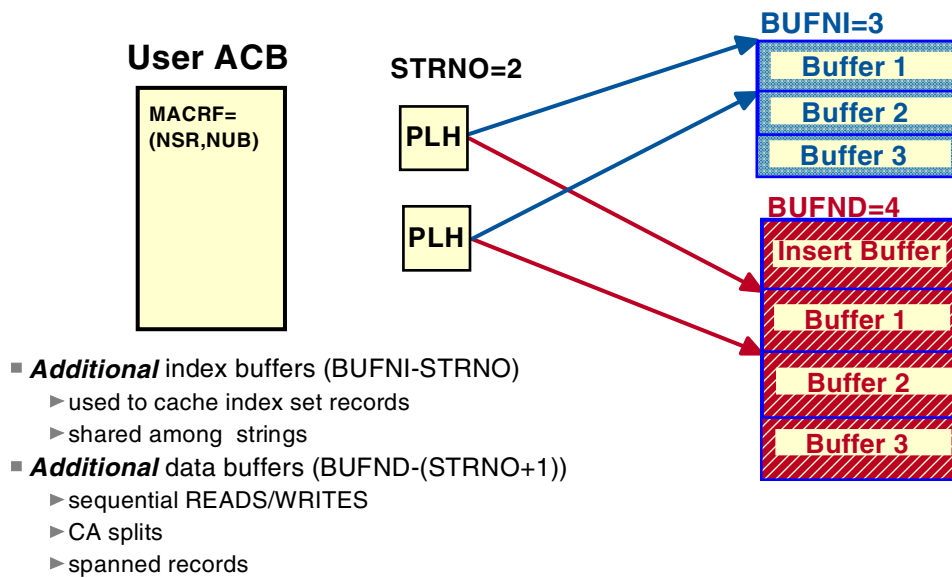


Figure 16. NSR buffering

## NSR with sequential access

NSR is best used for applications that use sequential or skip sequential as their primary access mode.

For the first loading of the buffers, VSAM uses the string's assigned buffer plus *all* additional buffers to a *maximum* of CIs/CA, since 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, and so forth. So, having more buffers than CI/CA plus one is useful only when having more than one string. When a string finishes using its buffers, additional buffers are available to other strings.

For overlap between CPU and I/O:

- For writes, once one-half of the buffers are filled by the application with records, an I/O operation is scheduled for the other half.



- For reads, once one-half of the buffers is being processed by the application, an I/O operation is scheduled for the other half.

This continues until a CA boundary is encountered and the application must wait until the last I/O to the CA is done before proceeding to the next CA. The I/O operations are always scheduled within CA boundaries.

When you are accessing data sequentially, you can increase performance by increasing the number of data buffers. When there are multiple data buffers, VSAM uses a read-ahead function to read the next data control intervals into buffers as buffers become available as described above.

Table 4 shows results of tests varying the number of buffers. Note that:

- As more buffers are used, the number of Execution Channel Program (EXCPs) is reduced. This means there are fewer I/O interrupts. That is why SRB time consumption drops so much compared with TCB time. For a discussion about what is a VSAM EXCP in numerical values, refer to Appendix B, “Miscellaneous performance items” on page 223. In general with VSAM, the number of EXCPs is equal to the number of SSCH instructions (and not the number of transferred CIs or physical blocks).
- The TCB time drops due to decrease in I/O preparation. After the optimum point, the TCB time increase due to excessive buffering management processing.

We recommend you read section B.1, “Our laboratory” on page 223, for a better understanding of the test results shown in this book.

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 writes. For details about defer write, refer to 2.6.9.5, “Deferring write requests” on page 75.

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 POINT macro does not cause read-ahead processing unless RPL OPTCD=SEQ is specified. POINT positions the data set for subsequent sequential retrieval.

Having only one index I/O buffer does not hinder performance, because VSAM gets to the next CI by using the horizontal pointers in sequence set records rather than the vertical pointers in the index set. Extra index buffers have little effect during sequential processing. You can see that in Table 4.

As 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, the amount of data buffers needed depends on how many records per data CI the data set has. In a data CI with many records, the number of data buffers is saturated before a data CI with few records. Saturated means you do not get better performance by increasing the number of buffers.

By specifying enough data buffers, you can access the same amount of data per I/O operation with small data CIs as with large data CIs.

Table 4 contains the results from our lab tests. Remember that, in our lab tests, the data set record processing is minimal (very I/O-bound). Also notice that we did not run in a controlled environment, so the elapsed time value can vary according to priority and system workload.

*Table 4. NSR — read sequential varying the number of buffers — STRNO=1*

Data buffers	Index Buffers	EXCPs	Device connect time	SRB time	TCB time	Elapsed Time
Default=2	Default=1	37735	28	0.95	2.7	51
10	1	7641	15.8	0.28	1.6	25
30	1	2549	13.4	0.15	1.4	19
50	1	1623	12.86	0.13	1.4	16
50	2	1623	12.85	0.13	1.4	16
90	1	927	12.5	0.11	1.5	16
181	1	466	12.4	0.1	1.7	14
<b>Note:</b> all times are in seconds						

As you can see, buffering can save I/O connect time. Refer to 2.7.6, “I/O service time (connect) for VSAM files” on page 117, to understand why.

If you experience a performance problem waiting for input from the device, you should specify more data buffers to improve your job's run time. More data buffers allow you to do more read-ahead processing.

For mixed processing situations (sequential and direct), you can improve performance in the following ways:

- Increase the number of index component buffers to the number of index levels to help direct access. Table 6 on page 70 shows how the number of index component buffers can improve performance for direct access.
- For sequential access, increase the number of data component buffers. Table 4 on page 66 shows how this can improve performance, based on tests in our lab. The best number of data component buffers varies according to the application processing done between each read to the data set. This means how fast the application requires new records to process. If the bottleneck is always the I/O operation, it is useless to increase the size of the buffer pool, as explained in 2.7.5.1, “Cache highlights” on page 113. The advantages of a large buffer pool only show up when there are fluctuations in the I/O rates in the application and in the I/O subsystem.

If your data set is SMS managed and has extended format, you do not need to worry about how many buffers to specify for the index or data component, you can take advantage of system managed buffering; for details, refer to 2.6.9.7, “System managed buffering (SMB)” on page 77.

#### ***Buffering in NSR initial load mode***

Initial load mode occurs when you load your data *sequentially* in a VSAM data set with a High Used RBA (HURBA) equal to zero. This is the first step after the DEFINE; or when a data set, defined with the REUSE attribute, is open with MACRF=RST (reset) in the ACB.

In this case, the RECOVERY or SPEED defined attributes of the data set are used.

When RECOVERY, the default option, is specified, all CAs are pre-formatted. All previous information from the direct access storage area are cleared, and end-of-file indicators are written.

With the SPEED attribute, data CAs are not performed, and an end-of-file indicator is written only after the last record is loaded. With SPEED you get better performance for initial load mode. With extended format data sets and SPEED, you can get much better performance using SMB, writing each CA with one I/O request. For details, refer to 2.6.9.7, “System managed buffering (SMB)” on page 77. Also refer to 2.6.7, “Initial load option” on page 54, to see our recommendations about using RECOVERY or SPEED.

Table 5 shows the results of loading an empty data set, with 450,002 records, using IDCAMS REPRO. The best result is obtained when:

Data buffers = 181 = CIs/CA + 1

Index Buffers = 3 (the number of index levels after data set loading)

For initial load mode processing, you get the best results using:

Index buffers = 3

Data buffers = CIs/CA + 1

Table 5. NSR - Initial Load mode varying the number of buffers

Data buffers	Index buffers	EXCPs	Device connect time	CPU Time	Elapsed time
Default	Default	39375	42.18	5.49	84
90	Default	2801	20.5	3.1	24
181	Default	2338	20.2	3.25	26
181	3	2109	20.1	3.16	25
360	Default	2338	20.32	3.53	41
All times shown are in seconds					

### **NSR with direct access**

NSR is not intended for direct access. However, many of your applications may use NSR for direct processing because it is simple. In this topic we will look at how NSR works for direct access. Remember that today you have solutions available to avoid direct processing without changing your code: system managed buffering, batch local share resources, and data acceleration in SmartBatch. We will cover their functions here.

In direct access, records are randomly accessed, and VSAM NSR does not use read-ahead buffers. Many types of data buffers do not increase performance, because VSAM reads only one data CI for each access. For output processing (PUT for update), VSAM defers write *only* if OPTCD=NSP is specified in the RPL macro; otherwise, VSAM immediately writes the updated CI.

With the NSR buffering technique, data buffers are not shared among strings. Each string is associated with one request parameter list (RPL). For example, 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.
2. VSAM then reads the data CI into storage and gives the user the requested record.
3. If the user then issues another direct GET for a record with a key of 6789, which is in a different CI from record 1234, the same data buffer gets used for this request, and the CI containing 6789 overlays the CI containing 1234.
4. If the user then issues another direct GET for a record with key 1235 (which is in the same CI as 1234), that CI must be read in again because the intervening GET for 6789 causes the previous buffer contents to be lost.

This problem cannot be solved by using multiple strings. If, for example, the requests for 1234, 6789, and 1235 use three different RPLs, both CIs will be in storage when the request for 1235 occurs, but VSAM will look *only* in the buffer assigned to the string related to RPL requesting 1235 and will not look in another string's buffers to satisfy the request. The CI will be read in again anyway.

These NSR characteristics lead to performance complaints in cases where the processing is random, but the same CI is requested multiple times with intervening requests to other CIs (revisiting).

When the number of I/O buffers provided for index records is greater than the number of strings:

- One buffer is used for the highest-level index record.
- Additional buffers are used, as required, for other index set index records.
- Buffers are shared among strings, as shown in Figure 16 on page 64.

With direct access, you should provide at least enough index buffers to be equal to the value of the STRNO parameter of the ACB plus one if you want VSAM to keep the highest-level index record always resident.

For a KSDS or VRRDS, you can increase performance for direct processing by increasing the number of index 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 to contain the entire high-level index set and one sequence set CI per string in virtual storage. Table 6 shows how adding index buffers improves performance. The elapsed

time can vary according to the system workload. Note that additional index buffers will not be used for more than one sequence set buffer per string unless shared resource pools are used.

VSAM reads index buffers one at a time. Index buffers are loaded when the index is referred to. When many index buffers are provided, index buffers are not reused until a requested index CI is not in storage.

VSAM 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. Because the characteristics of the data set cannot allow a small index, you should be aware of how index I/O buffers are used so you can determine how many to provide.

Many data buffers do not increase performance, because only one data buffer is used for each access, as you can see in Table 6. The elapsed time can vary according to the system workload.

*Table 6. NSR buffering with direct access — STRNO=1*

<b>Data Buffers</b>	<b>Index Buffers</b>	<b>EXCPs</b>	<b>CPU time (sec)</b>	<b>Elapsed time (min)</b>
Default	1	199000	13.62	4
30	1	199000	13.60	4
Default	3	118000	8.46	2.4
Default	4	99376	7.26	2.0
Default	5	99376	7.27	2.0
Default	10	99376	7.28	2.0
Default	50	99376	7.42	2.0

If your job is having performance problems randomly accessing VSAM data sets and your program application uses NSR buffering, you can improve performance with no changes in your applications, as follows:

- If your data set is SMS managed, has extended format, and your installation has DFSMS V1R4 or later, you can use system managed buffering. For details, refer to 2.6.9.7, “System managed buffering (SMB)” on page 77.
- For batch applications, you can use BLSR; for details, refer to 2.6.9.8, “Batch local shared resources (BLSR)” on page 81.

Refer to A.3, “Sample program to extract information from SMF record type 64” on page 214 for information that can help you find that jobs which are candidates to use BSLR. Remember that BLSR should be used only for direct access. Consider the use of SMB, where you can get better results.

Recommendations:

With NSR:

- For sequential processing, use SMB to get an optimum buffering, or:  
BUFNI = Number of 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 LSR and get an optimum buffering, or:  
BUFNI = STRNO plus number of levels  
BUFND = Let the default value (STRNO plus one)

### **2.6.9.3 Local Shared Resources (LSR)**

The buffers in a LSR pool:

- Are explicitly constructed via BLDVRP macro, before the OPEN for the first data set that will use it.
- Are shared among VSAM data sets accessed by tasks in the same address space.
- Are located in the private area and ESO hiperspace (if specified in BLDVRP macro)
- Are replaced based on the least recently used (LRU) algorithm.
- Have no look-ahead for sequential processing.

Buffering is also affected by the following:

- For subtask sharing, when a CI is not available for the type of task processing requested, VSAM under LSR and GSR buffering has a proper way of managing the contention. For details, refer to 2.6.6, “Share options” on page 53.
- Programs using LSR can invalidate BP contents through MRKBFR macro and forced them to be written immediately through 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 or with multiple references to the same data. Subsequent access to data does not have to access DASD.

With LSR and GSR, the number and size of buffers are specified in BLDVRP macro (see Figure 17) and are not overridden by ACB or JCL. The buffer pool is identified by a number and a data set is connected to it through the ACB macro, where the buffer pool ID is specified. After all data sets using a resource pool are closed, the resource pool can be delete issuing the DLVRP (delete VSAM resource pool) macro. For more details about macros, refer to *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913.

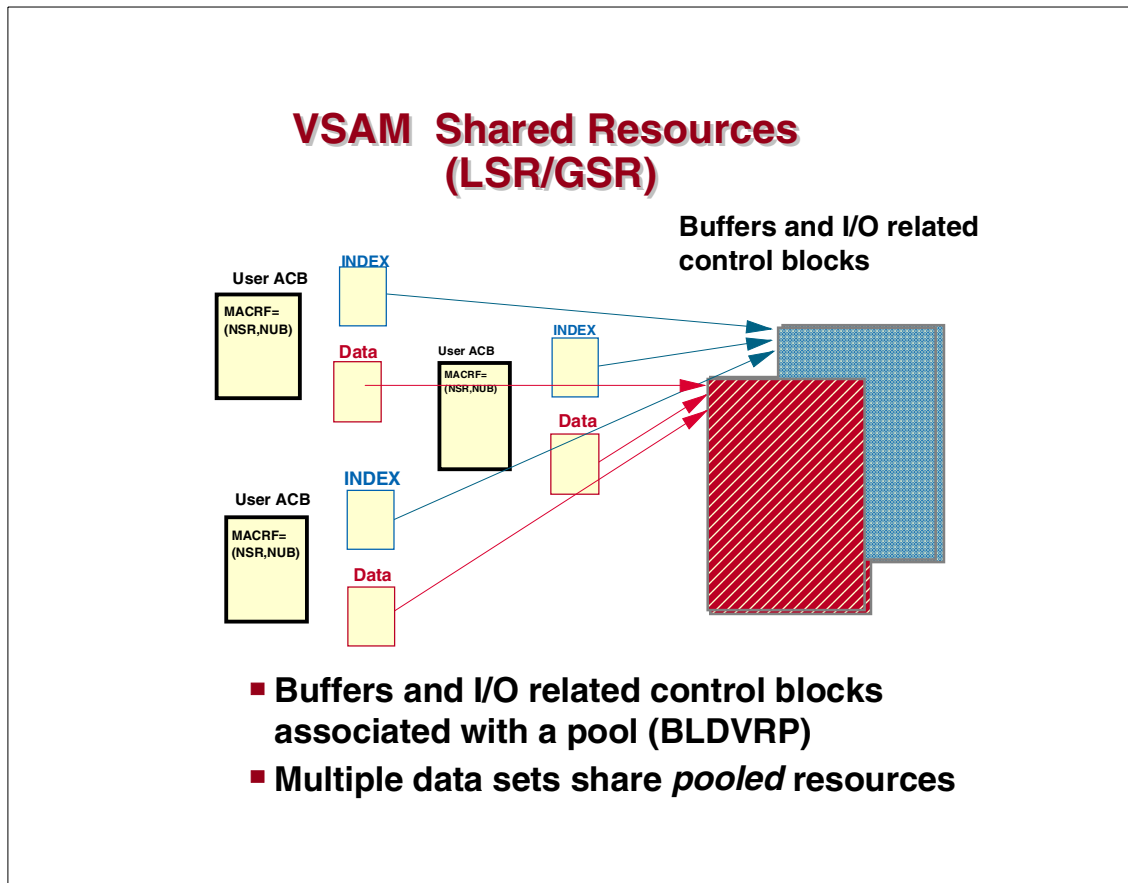


Figure 17. VSAM shared resources (LSR/GSR)

Online transaction program (OLTP) applications like CICS and IMS are the biggest users of LSR shared resources because they typically need to have hundreds of data sets open in one address space at any given 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 them, not directly by the user application. Information for building the shared buffer pool is specified in the product FCT table. Refer to 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 requests in cases where subsequent requests can be satisfied by the data already in the buffer pool. For more details, refer to 2.6.9.5, “Deferring write requests” on page 75.

The search in the buffer pool is not affected by the pool size once the search is by hashing, so there is no overhead. Also, with LSR buffering, your application can use hiperspace as a second level of buffering. For details, refer to “Using buffers in hiperspace” on page 74.

If you intend to build an application using LSR buffering techniques:

- Note that LSR is suited for direct access; for sequential access, use NSR.
- Build resource pools before any open to the data sets that will use them.
- Build a separate resource pool for indexes to avoid index data being flushed by data being read.
- Use defer write if possible.

For details how to build an LSR pool, refer to *DFSMS/MVS Using Data Sets*, SC26-4922.

#### ***LSR with direct access***

LSR buffering is suited for direct access. If your applications use NSR buffering and direct access, and you are having performance problems, you can take advantage of LSR buffering. Refer to 2.6.9.7, “System managed buffering (SMB)” on page 77 and 2.6.9.8, “Batch local shared resources (BLSR)” on page 81.

#### ***LSR with sequential access***

The LSR buffer management technique is for random access, not sequential processing. If you are relying on sequential read-ahead for good performance LSR buffering could degrade, rather than improve performance. LSR does not do read-ahead. Your sequential request is not read in more than one data CI per I/O operation. You should use NSR buffering, where VSAM can read up to a CA's worth of data CIs in a single I/O, conditions permitting. For details about NSR, refer to “NSR with sequential access” on page 64.

Recommendations:

If you intend for your batch application to use the LSR buffering technique:

1. Write the application 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 when processing or BLSR if there is no SMS.

### ***Using buffers in hiperspace***

VSAM LSR can use a second level of buffering, that is, the hiperspace expanded storage only (HS ESO). In the BLDVRP macro, the requester can ask for a certain amount of the buffer pool (BP) in the address space and another in an HS ESO. At OPEN time, both are allocated.

The reasons for the HS ESO buffer pool are:

- It saves virtual storage addresses in the address space.
- If you have 2 GB of central storage (the architectural limit) and plenty of expanded storage, HS ESO is a good way to use the expanded storage.

VSAM use HS ESO in a *store-through* mode. That is, all the buffers in the HS ESO have the most current CI content. Then, if real storage manager (RSM) steals a page from the expanded storage frame backing up the HS ESO CI, integrity is not affected.

However, HS ESO buffering performance is not as good when compared to an address space. The reason is that the CPU cannot process a page in expanded storage. The page must be moved to central storage in order to be processed. In VSAM terms, if a CI is not in the address space buffer but rather in an HS ESO buffer, the following operations must be executed:

1. Move a CI from the address space buffer to the HS ESO buffer to make room.
2. Bring the referred CI from the hiperspace ESO buffer to the address space buffer.

If you need 500 MB of buffer pool space, is better to have all of it in an address space (if you are not short of virtual or central storage) than 250 MB in the address space and 250 MB in the hiperspace ESO.

For LSR, hiperspace buffers are created with the resource pool, using the BLDVRSP macro, where you specify the size and number of hiperspaces buffers. The size of the hiperspace buffer *must* be a multiple of 4096 and *must* match the CI size of the data component. The default is zero; this means, do not use hiperspace.

If your application uses NSR buffering with direct access, you can use SMB to convert to LSR. You can also ask for hiperspace buffers; for details, refer to 2.6.9.7, “System managed buffering (SMB)” on page 77.

#### 2.6.9.4 Global Shared Resources (GSR)

GSR is similar to LSR buffering technique. The GSR characteristics that differ from LSR are:

- The buffer pool is shared among VSAM data sets accessed by tasks in *multiple* address spaces.
- Buffers are located in CSA.
- Buffers cannot use hiperspace.
- The separate index resource pools are not supported for GSR.

With these differences in mind, refer to 2.6.9.3, “Local Shared Resources (LSR)” on page 71 for details on how GSR buffering technique works.

#### 2.6.9.5 Deferring write requests

Use defer write (if possible). With defer write, VSAM uses the buffer pool in a *store in* mode. That is, the updates are not propagated immediately to DASD.

The performance advantages of deferring writes are these:

- If the same record is updated  $n$  times and is then written asynchronously to DASD, you save  $n-1$  I/O operations. It is clear that, if there are no further updates, there will be no saves.
- The application doing the write does not wait for the I/O operation.

The negative aspect of deferring writes is that VSAM does not have a log (at present). If the system fails before the buffer destage, you lose some updates in your file. So, is up to you to make the decision based on the type of data you are processing.

Defer write option is only valid for LSR/GSR. The defer write option is bypassed in LSR/GSR if SHAREOPTIONS 4 is specified. Refer to 4.2, “Sharing VSAM data sets” on page 183.

When NSR defer write is used, with the exception of SHAREOPTIONS 4, the buffers are immediately refreshed.

You specify that writes are to be deferred by coding MACRF=DFR in the ACB, along with MACRF=LSR or GSR:

```
ACB      MACRF=( { LSR | GSR } , { DFR | NDF } , ... ) , ...
```

You can also specify defer write through the use of SmartBatch; refer to 2.8, “VSAM and SmartBatch” on page 121.

#### 2.6.9.6 Locating VSAM buffers above 16 MB

The *default* VSAM allocation for a resource pool is below 16 MB. The location of the buffers and I/O control blocks can be controlled in two ways using the RMODE31 parameter:

1. In an assembler application program:
  - For NSR buffering, in the ACB of the data set
  - For LSR buffering, in BLDVRP macro
2. In the JCL, via AMP parameter

The values you may specify for RMODE31 are:

- ALL is used to allocate I/O relate control blocks and buffers above 16 MB.
- BUFF is used only to allocate buffers above 16 MB.
- CB is used only for I/O relate control blocks above 16 MB.
- NONE is the default. VSAM allocates I/O related control blocks and buffers below 16 MB.

With the flexibility of JCL, you can avoid the work of changing an existing application. But be aware that programs with AMODE=24 can abend with SOC4 when *locate* mode is used and RMODE31=BUFF is specified.

Locate mode is used when OPTCD=LOC is specified in the RPL, and indicates to VSAM to return to the application the address of the record, which is located in the buffer. Using locate mode, programs addressing 24 bits cannot access data above 16 MB (addressing 31 bits).

There is also a *move* mode, where VSAM moves your logical record from the buffer to an area you specify.

COBOL for OS/390 always:

- Uses MVE mode; VSAM moves the record to an area whose address is indicated in the RPL, built by COBOL.
- Requires VSAM buffers above 16 MB.

You can relieve the storage constraint below 16 MB 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. Refer to 2.6.8, “Region size” on page 55.

Recommendations:

- Use VSAM buffers above 16 MB.

#### **2.6.9.7 System managed buffering (SMB)**

SMB was introduced in DFSMS V1R4 and enables VSAM to determine the optimum number of index and data buffers, as well as the type of buffer management (LSR or NSR).

SMB does not perform miracles; it just allocates the optimum number of data and index buffers based on the access used. Remember that the region size should support the increase in virtual storage.

Performance improvements can be dramatic, particularly where defaults for buffering are used and a switch from NSR processing to LSR is made (for direct processing). This can be seen in Table 7 on page 79.

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 (SMS) and use a data class defined with DSNTYPE=EXT. For details about extended format; refer to “Use the ECKD extended format:” on page 117.
- In the application program, ACB *must* be NSR and MACRF cannot contain:
  - ICI — Improved control interval processing
  - AIX — Processing the data set through the alternate index of the path specified in the DDname
  - UBF — Management of I/O buffers left up to the VSAM user, as in DB2

When the conditions above are not satisfied, the job does notabend, the SMB services are not used and no messages are issued.

You can invoke SMB services through one of the following methods:

1. Using a data class defined with RECORD\_ACCESS\_BIAS=SYSTEM
2. Specifying ACCBIAS in the AMP parameter of JCL.

The order of precedence for specifying values are shown in the SOURCE column in Table 3 on page 60.

The AMP JCL parameter can be used to specify access bias with a subparameter ACCBIAS, where you specify the type of buffering management technique. You can specify one of the following values:

- DO — SMB optimizes buffers management to direct access. SMB changes the buffering management to LSR and allocates the adequate number of buffers for direct processing.
- DW — To indicate to SMB that the processing is mixed direct and sequential, but with dominance of direct access. So, more index buffers are allocated to support direct processing but some buffers will be reserved for data to help any sequential processing that might occur. The buffering management technique is changed to LSR.
- SO — Indicates to SMB to optimize buffer allocations for sequential processing. More data buffers are allocated to support sequential access.
- SW — Specifies mixed processing, but with dominance of sequential access. SMB optimizes buffer handling for sequential processing allocating more data buffers, but buffers will be reserved for index to help direct access. SMB is faster than NSR for sequential process, as indicated in Table 9 on page 80.
- SYSTEM — Let the system determine the buffering technique. Based on the ACB's MACRF and storage class specifications one of the four techniques specified above will be used.
- USER — Bypass SMB. This 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), SMB converts NSR buffering to LSR. In this case, you can use the following AMP parameters to tell the LSR buffer manager how to handle the processing of the buffers:

- SMBVSP — Specifies the amount of virtual storage to obtain for buffers when opening the data set. Used to override the default buffer space to be obtained, which is calculated assuming that 20% of the data will account for 80% of the accesses. The buffer space acquired is split across two LSR pools: one for the index and one for the data. The specification can be done in either of two formats:
  - SMBVSP=nnK
  - SMBVSP=nnM
- 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. The value specified is not a direct multiple

of the number of virtual buffers that are allocated to the resource pool, but act as a weighting factor for the number of hiperspace buffers to be established. The hiperspace size buffer will be a multiple of 4k. These buffers may be allocated for the base data component of the sphere. If the CI size of the data component is not a multiple of 4k, both virtual space and hiperspace is wasted. Before specifying this parameter, see details about the use of hiperspace in “Using buffers in hiperspace” on page 74. The default is 0 and means that hiperspace is not used.

- **SMBDFR** — Allows the user to specify whether writing the data from a buffer to DASD can be deferred until the buffer is required for another request or the data set is closed. A CLOSE macro invoked with TYPE=T (temporary, does not need an OPEN to restart processing) option does not write the buffers to DASD 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)

Table 7 shows the benefits of using SMB, compared with the use of BLSR or buffering default. When we ran our test, we had 61 CA splits. Some CPU time was due to managing these splits; allocating extents, and data set catalog entry update. For considerations on CA and CI splits, refer to 2.6.4, “Free space” on page 50.

If you specify RECORD\_ACCESS\_BIAS=SYSTEM in the data class, you do not have to worry about what kind of access is done, or bother with JCL modifications. You can also see that for extended format data sets, you can get performance enhancements, even with default buffering. In our lab tests (refer to Table 7), the number of EXCPs, CPU time, and connect time is less than non-extended format data sets. For details on why this happens; refer to 2.7.7.2, “Use of extended format” on page 121 and 2.7.7.2, “Use of extended format” on page 121.

*Table 7. Direct access: benefits of using SMB — updates and insertions*

Ext format	Buffering	EXCPs	connect time (sec)	CPU time(sec)	Elapsed time (min)
No	Default	772057	549	52.54	16.2
Yes	Default	771265	541	45.67	19.3
No	BLSR	293960	255	24.83	6.8
Yes	SMB	79741	75	11.04	2.5

Ext format	Buffering	EXCPs	connect time (sec)	CPU time(sec)	Elapsed time (min)
Gain using SMB (%)		90	86	79	85

When SYSTEM is specified, the parameters used to determine how buffers are allocated and managed are:

- ACB's MACRF values of SEQ, DIR, and SKP
- The storage class values for BIAS and MSR

Table 8 shows how ACB's MACRF and storage class BIAS parameters affect the buffer allocation and management when SYSTEM is specified.

*Table 8. Some effects of ACB's 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
<b>Note:</b> DO=Direct Optimized; DW=Direct Weighted; SO=Sequential Optimized; SW = Sequential Weighted				

The advantage of using SYSTEM is that you do not need to be concerned about what kind of access is done. Table 9 shows the results when loading our laboratory data set using default buffering and SMB. You can see, the results are the same using sequential optimization (SO) or SYSTEM. The advantage of SYSTEM is that all you need to do is specify in the data class. For details about our lab; refer to B.1.1, "General lab description" on page 223.

*Table 9. Initial load mode comparing SMB with no-SMB buffering*

Extended format	Buffering	EXCPs	Connect time	CPU time	Elapsed time
No	NSR - Default	39375	42.18	5.49	84
Yes	ACCBIAS=SO	2341	21.5	2.7	27



Extended format	Buffering	EXCPs	Connect time	CPU time	Elapsed time
Yes	ACCBIAS=SYSTEM	2341	21.5	2.7	28
<b>Gain using SMB (%)</b>		<b>94</b>	<b>49</b>	<b>51</b>	<b>67</b>
<b>Note:</b> All times are shown in seconds					

### ***SMB message IEC161I 001(8,36)***

When processing a VSAM data set using SMB, you can receive the message IEC161I 001(8,36)-087. It is issued by the BLDVRP macro and indicates that there was not enough virtual storage to satisfy the request done by SMB. SMB gets the available storage and processing goes on. To get optimum SMB buffering, you should provide enough virtual storage. Refer to 2.6.8, “Region size” on page 55 and how you can relieve the use of storage below 16 MB by specifying that VSAM allocates buffers above 16 MB. Refer to 2.6.9.6, “Locating VSAM buffers above 16 MB” on page 76.

Also refer to 2.8, “VSAM and SmartBatch” on page 121, to learn how you can dynamically define your buffer pool through the SmartBatch.

### **2.6.9.8 Batch local shared resources (BLSR)**

BLSR is a subsystem that provides advantages in an application using VSAM NSR buffering techniques to switch to LSR without changing the application source code or link-editing the application again. Only a JCL change is required.

Your application performance will improve using BLSR when direct access is used and the same CI is referenced more than once in the processing. Using the BLSR subsystem with sequential access could degrade performance rather than improve it. For information about how LSR works, refer to 2.6.9.3, “Local Shared Resources (LSR)” on page 71.

For mixed processing (some direct, some sequential), you may benefit from using BLSR. If the amount of data to be processed sequentially is not very large, you can compensate for the lack of read-ahead by using a large data CI size.

BLSR supports the VSAM 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. You can also use hiperspace, and you can restrict its use with RACF or an equivalent security software.

The BLSR subsystem has the following restrictions:

- The ACB cannot be above 16 megabytes. 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, and 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, then it is reopened for LSR. Similarly, if the data set was not eligible for LSR processing the first time, then it is reopened for NSR processing, even if LSR is now applicable.

High-level languages refresh the DDNAME for each open. Consequently, the subsystem is always called for the following:

- COBOL/VS programs using the ISAM interface to access VSAM files.
- PL/1 programs written for ISAM accessing VSAM files.
- Other programs using the RDJFCB macro to try to identify the file type, for example, IDCAMS.

Before using BSLR, the subsystem must be installed. Contact your installation system programmer, or refer to the manual *MVS Batch Local Shared Resources*, GC28-1469.

When the subsystem is active, to invoke its services, add the SUBSYS parameter to the JCL. The following example illustrates how to do this.

If, for example, the application opens the following data set for NSR processing:

```
//VSAMDD DD DISP=SHR,DSN=VSAMDSN
```

You can convert to the BLSR subsystem as follows:

1. Change the DDNAME on the previous JCL command statement, for example, from VSAMDD to NEWBUFF:

```
//NEWBUFF DD DISP=SHR,DSN=VSAMDSN
```

2. Add the following DD statement, where the SUBSYS subsystem-name subparameter is BLSR and the SUBSYS DDNAME subparameter is the DD name selected in step 1:

```
//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 correct DDNAME (`NEWBUFF` in the example).

When the application opens the `VSAMDD` ACB, the BLSR subsystem completes the conversion to LSR processing.

The following system parameters are not allowed with the `SUBSYS` parameter:

`*`, `AMP`, `BURST`, `CHARS`, `COPIES`, `DATA`, `DDNAME`, `DYNAM`, `FLASH`, `MODIFY`, `QNAME`, `SPACE`, `SYSOUT`.

You must nullify any of these parameters if they are specified on a DD statement you are overriding.

You specify the `SUBSYS` parameter as:

```
SUBSYS=(subsys-name, 'DDNAME=value', 'subparm1=value', ..., 'subparmn=value')
```

Where:

- `subsys-name` is the name given to BLSR subsystem, usually, `BLSR`
- `DDNAME` value is the name of the DDNAME to be open by the application program.

The following subparameters are allowed on the `BLSR SUBSYS` parameter: `BUFND`, `BUFNI`, `HBUFND`, `HBUFNI`, `RMODE31`, `STRNO`, `DEFERW`, `SHRPOOL`, `BUFSD`, `BUFSI` and `MSG`. For the meaning, default and the use of them; refer to the manual *MVS Batch Local Shared Resources*, GC28-1469.

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 will get better performance using SMB. For details, refer to 2.6.9.7, “System managed buffering (SMB)” on page 77.

BLSR can also be implemented by SmartBatch; refer to 2.8, “VSAM and SmartBatch” on page 121 for more information.

Recommendations:

- If possible, use SMB, the results are better, as you can see in Table 7 on page 79.

- If possible, locate the buffers above 16 MB.

### **2.6.10 Data compression**

To compress means to store data in a format that requires less space than the original data. There are quite a few methods (algorithms) to compress data, such as the following:

- Character-based methods: Huffman, Run-length encoding, Ziv-Lempel
- Bit-level methods: Image data compression, IDRC (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:

- Compression dictionary
- Expansion dictionary

The same method may use many dictionaries.

S/390 uses the Ziv-Lempel (ZL) method in software and hardware options.

ZL-based schemes work by entering phrases into a dictionary and then, when a repeat occurrence of that particular phrase is found, outputting the dictionary index 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. S/390 compression uses the ZL1 version of the ZL implementation, and the RVA family of products uses ZL2.

#### **2.6.10.1 Where to use compression**

Compression can be executed in the processor or outbound in an I/O controller (for example, tape). With CPU compression, you save:

- I/O buffer pool space
- Channel cycles
- Controller cache space
- Controller internal data path cycles
- Media space (disk and tape)
- Transmission line cycles (for TP)

The advantages of compressing in the controller are to save:

- Controller cache space
- Controller internal data path cycles
- Media space (disk and tape)
- CPU cycles

### **2.6.10.2 OS/390 compression interface**

The interface to data compression in OS/390 is through the Compression Management Facility (CMF), which consists of two parts.

#### ***Compression service activation***

Compression service activation covers checking and setting up the required compression environment.

- It determines if the Compression Call instruction is available; if not, the compression is done by software. It verifies if the SYS1.DBBLIB data set is available. It contains dictionary building blocks (DBB) used for compression.

#### ***Compression management services (CMS)***

Compression management services covers the actual process a data set goes through when compression is requested. CMS provides and carries out the following services used by VSAM.

#### **Candidate data set verification**

Candidate data set verification has the following requirements:

- KSDS organization only. Only data is compressed, indexes are not compressed (AIX are not compressed).
- SMS functions must be active and the data set must be SMS managed and in extended format.
- Data class assigned has to specify the DSNTYPE=EXT with a required COMPACTION=Y (blank defaults to N).
- Have a primary allocation of at least 5 MB (data component only) due to the amount of sampling needed to develop a dictionary token. If no secondary allocation is specified, then the primary allocation must be at least 8 MB.
- Have a minimum record length of 40 bytes (not including key length).
- IMBED, CI Mode processing and key range are 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/MVS Using Data Sets, SC26-4922-01.

## Dictionary selection

There are two forms of compression: generic and tailored (refer to Figure 18 on page 87.)

- Generic compression:

This involves the sampling and interrogation of the compression eligible data set by CMF. Next, 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 using DBBs during the sampling and interrogation, the dictionary does not exist when the first bytes of the data set are written. Once 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 (not for VSAM data sets). With tailored compression, the system attempts to derive a compression dictionary that is tailored specifically to the initial data written to a data set. Once a tailored dictionary is derived, it is imbedded in the compressed data set. This technique is expected to provide improved compression ratios, thereby reducing DASD usage and channel traffic.

Because the dictionary is tailored to the user data, significantly more data is sampled than was required by generic compression. The process of sampling the data and building the dictionary during creation of a new data set takes more CPU cycles and is, therefore, 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 installation has the option of either using the new tailored compression or continuing to use generic DBB-based compression first introduced with DFSMS/MVS V1R2. Tailored compression allows for more types of data to be compressed, for example, where non-English languages are used.

Because the original generic dictionary was developed with standard American-English scripts, files containing sequences of characters that do not appear in the American scripts do not compress very well. The tailored dictionary avoids this problem, as it is generated dynamically from the data itself, for each data set.

Dynamically generating a tailored dictionary from the data itself should make tailored compression more useful to the non-English-speaking world. Tailored compression support does not apply to VSAM KSDSs, which can continue to be compressed with generic DBB dictionary compression.

VSAM KSDSs can only use generic compression.

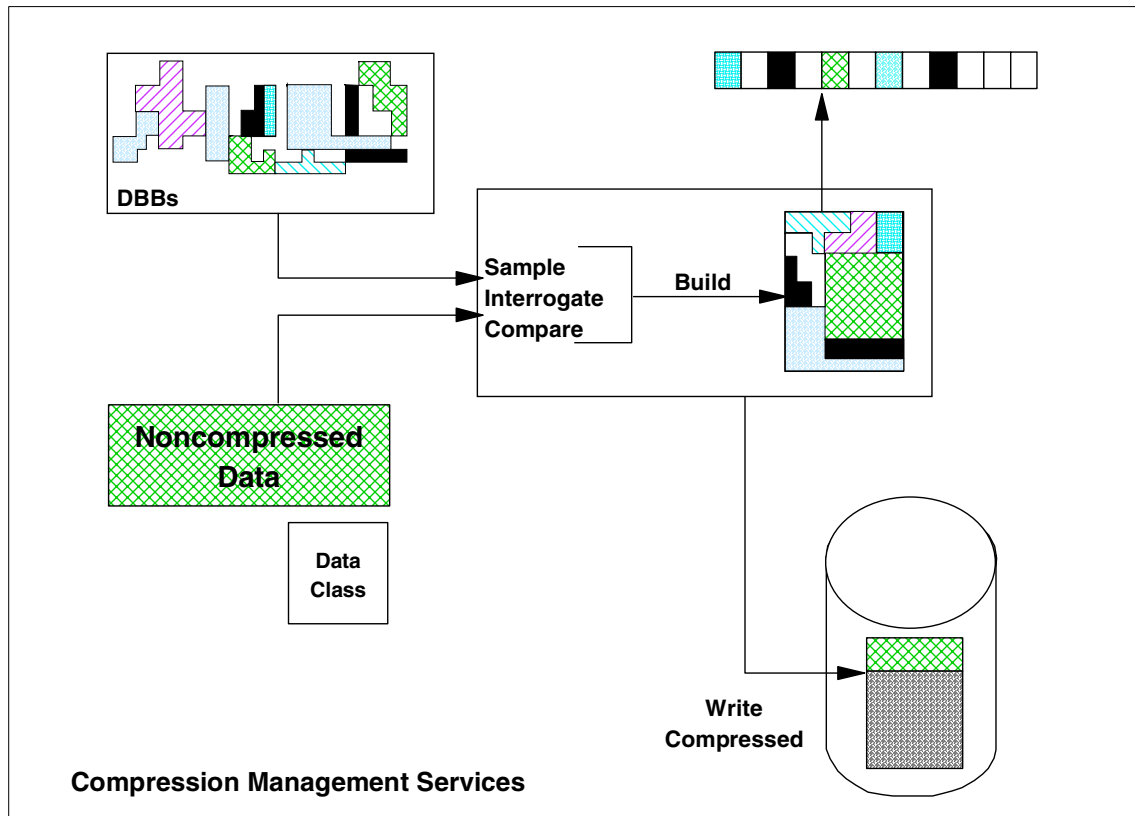


Figure 18. CMS dictionary selection

#### 2.6.10.3 Forms of candidate data sets

A compression-eligible data set can exist in one of three forms, depending on the moment:

- *Dictionary selection:* The data set is eligible, but its makeup is yet to be determined. The dictionary tokens are compared with the data set contents during interrogation and sampling. Interrogation maps bytes into alpha, numeric, upper- or lower-case, and sampling evaluates DBB compression efficiency.
- *Mated:* The data set is mated with the appropriate dictionary; this concludes the sampling and interrogation processes.
- *Rejected:* A suitable dictionary match was not found during sampling and interrogation, so 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 ZL compression, resulting in rejection, such as these:

- Small data sets.
- Data sets in which the pattern of the data does not produce a good compression rate, such as image data.
- Small records. Compression is performed on a logical record basis. When logical records are very short, the cost of additional work may become excessive compared to the reduction in size that compression achieves.

#### 2.6.10.4 Data compression and decompression

Data compression and decompression are invoked whenever read and/or write or get and/or put is done on 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, and their combination constitutes the dictionary token associated with the data set and held in the catalog entry as well. Compression and decompression 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, because the dictionary is a table dynamically built in storage from the token information that enables the compression and decompression of a data set when it is opened.



Using this approach, DFSMS/MVS retains responsibility for dictionary management and shields the user and application from the physical representation of compressed data on disk.

#### **2.6.10.5 Compression concepts**

Following are some basic concepts regarding compression:

- Import into an empty data set does not propagate extended format and compression information.
- When data is compressed, the length of a stored record may change after an update without any logical record length change.
- Locate mode 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.
- IDCAMS REPRO copies data sets without decompressing:
  - If the target data set is eligible for compression, and
  - If the target device is a like device, or CI sizes are equal for VSAM
- Otherwise, REPRO decompress the data when reading and optionally compresses the data when writing.
- Our SMF sample report shows the compression factor.
- LISTCAT lists the compression related information.
- IEHLIST indicates that the data set is compressed.
- DFSMSdss can be used for:
  - Logical dump; this does not change format from compressed to non-compressed or vice versa; and includes cataloging information with dump.
  - Logical restore; this does not change format from non-compressed to compressed.
  - Logical copy; this never changes the format from non-compressed to compressed.
  - DFSMSHsm avoids double compression.

#### **2.6.10.6 VSAM compression**

VSAM compression is done transparently to the application, through the data class (DC) parameter in SMS data sets. This DC assigned to the data set has to specify the following DSNTYPE=EXT with a required COMPACTION=Y (blank defaults to N). The following screen pictures the ISMF list of the DC:

DATA CLAS			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

VSAM compression only applies to KSDS in extended format. All the fields to the left of the key in the logical record are not compressed. In your next data model, you can define the key field with offset equal to zero.

Compression affects the catalog, VTOC, and SMF information about VSAM data sets. Refer to 2.6.10.8, “Compression information sources” on page 91 for information on how to get this data.

### 2.6.10.7 Compression performance

The relative CPU compression cost depends on:

- Dictionary size:  
Usually, you do not have much control over this.
- Ratio of reads to writes of compressed records:  
Ziv-Lempel expands faster (less CPU cycles) than compresses. This means that it is better suited for data sets with a high read-to-write ratio.

Table 10 shows the amount of processing per compressed read or write operation that we found in our lab environment.

Table 10. Comparing compression

Type of access	Compression	EXCPs	Device connect time (sec)	Step elapsed time (sec)
Initial load	No	2341	21.5	28
Initial load	Yes	396	3.6	20
Direct access	No	79,741	75	11.04
Direct access	Yes	8278	7.1	12

Recommendations:

- Do not use compression for data sets with low read-to-write ratio (less than 60%).
- Make key offset equal to zero.

#### **2.6.10.8 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 additional information about compressed format 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:

- 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)
- Whether user data sizes are valid (SIZES-VALID)
- Compression characteristic record.

- SMF:

Compression information in the type 14 and 15 records is updated when CLOSE is performed on a sequential compressed format data set. Information on the dictionary token selected and the compressed and non-compressed data set size is added to the SMF type 14 and 15 records.

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 used for compression. You can find more detailed information on the SMF record layout in the MVS/ESA SP V5 System Management Facilities (SMF), GC28-1457.

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'

### 2.6.11 Data striping

Usually, in a multi-extent, multi-volume VSAM data set processed in sequential mode, processing does not present any type of parallelism for I/O operations among the volumes. This means that when an I/O operation is executed for an extent in a volume, no other I/O activity from the same task is scheduled to the other volumes. In a situation where I/O is 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* along the volumes; instead they are organized in stripes.
- Parallel I/O operations are scheduled 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 concurrent I/O requests to multiple 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 in order to satisfy a single application request will be referred to as an *I/O packet*.

### 2.6.11.1 VSAM data striping

Data striping support for VSAM is initially released with DFSMS 2.10. Support is provided for all VSAM organization, including KSDS, ESDS, RRDS, VRRDS, and LDS. Support for LDS was made generally available via PTF to DFSMS/MVS V1.5.

Following are the characteristics of a VSAM striped data set:

- Striping is done by CI, as opposed to track for SAM.
- A stripe is associated with a single volume or a set of volumes.
- Only the data component of a base cluster may be striped.
- A data set may have up to a maximum of 16 stripes.
- A stripe on a single volume has a maximum of 123 extents per stripe.
- A multi-layered stripe has a maximum of 255 extents per stripe.
- The following features are supported:
  - Data in extended addressability (>4GB):
  - Data in compressed format
  - Partial release
- A data set is system-managed.
- A data set is allocated in extended format.
- A data set may be assigned either GUARANTEED SPACE or NON-GUARANTEED SPACE.
- 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, up to 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) in the storage class.

Figure 19 is an example of a four-stripe VSAM data set.

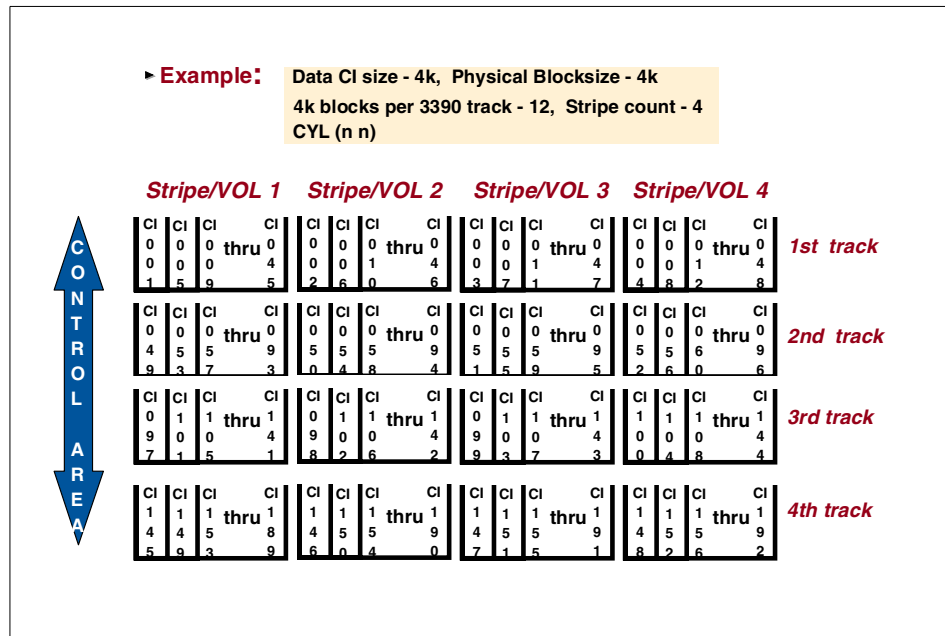


Figure 19. Striped VSAM data set

### 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.

Once 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 20 shows an example of the concept of layering with a three-stripe data set.

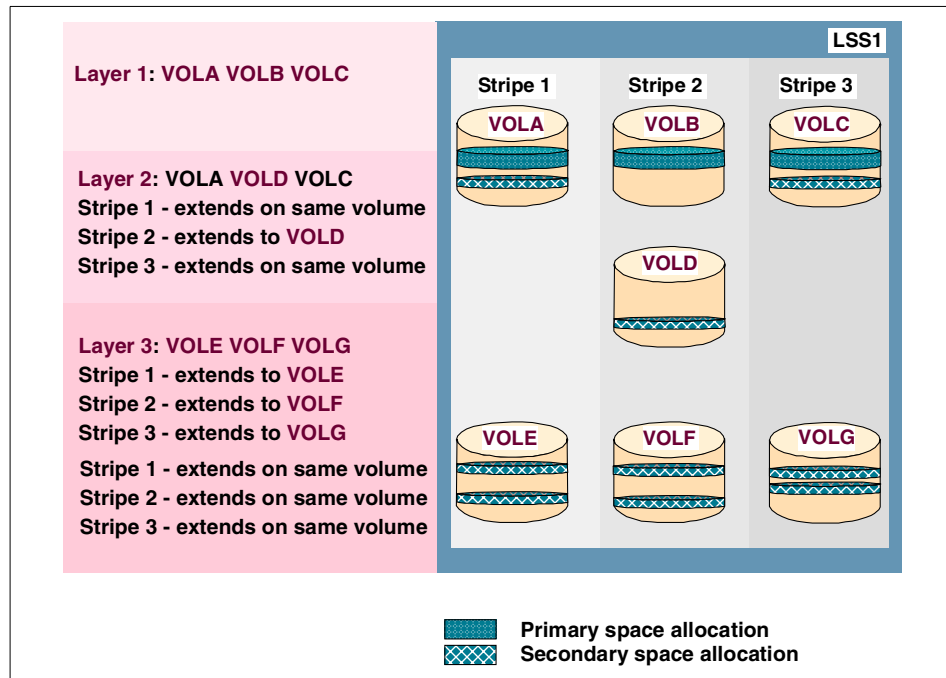


Figure 20. Layering in VSAM data set striping

### Implementing VSAM data striping

In order to create a striped VSAM data set, you must do the following:

- Define an SMS-managed VSAM data set in extended format.  
 You should specify *Data Set Name Type* in the data class as EXT Required. *Data Set Name Type* may be set to EXT Preferred but the data set is not defined as striped if it is not allocated in extended format.
- Specify an SDR value in the storage class for either guaranteed or non-guaranteed space. This will tell SMS that you want to implement striping.
- For *guaranteed space*, specify the VOLUME COUNT in the data class or the VOLUMES parameter in JCL. The JCL specification overrides the data class.

If you do not specify either the volume count or the volume serial numbers, you only get a single stripe.

Refer to 2.6.1.1, “Guaranteed Space” on page 43 for additional information on guaranteed space.

- For *non-guaranteed space*, just specify the SDR value in the storage class. SMS computes the number of stripes based on approximately 4MB/sec rate. The volume count is ignored.

As a simple example: if you specified a SDR of 17MB/sec, your data set will be defined with 4 stripes.

### **Examples of defining striped data sets**

In this example, we create a striped VSAM data set using guaranteed space, and a specified volume count of 4. The data class name is KEYEDEXG, 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)
/*
```

Following is the content of the KEYEDEXG data class:

```
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 . . . . . :
CIsze 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 (%) :

```

Following is the content of the STRIPE storage class.

```

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 . . . . . :

```

This is the job output. The data set is explicitly defined with 4 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

```

In this example, we create a striped VSAM data set using non-guaranteed space and SDR 17MB/sec:

```

//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 (KEYEDEXG) -
              STORAGECLASS (STRIPE))
          LISTC ALL ENTRIES (DAWN.KSDSEXT)
/*

```

Following is the content of KEYEDEXG data class:

```

CDS Name      . . . : SYS1.SMS.SCDS
Data Class Name : KEYEDEXG

Description : TO BE USED FOR STRIPING TEST

Recorg . . . . . : KS
Recfm . . . . . :
Lrecl . . . . . : 300
Keylen . . . . . : 8
Keyoff . . . . . : 0
Space Avgrec . . . . . : K
      Avg Value . . . . . : 300

```

```

        Primary . . . . . : 600
        Secondary . . . . . : 100
        Directory . . . . . :
Retpdt Or Expdt . . . . . :
Volume Count . . . . . : 8
    Add'l Volume Amount . . :
Imbed . . . . . :
Replicate . . . . . :
CIsze 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 (%) :

```

Following is the content of the STRIPE storage class using non-guaranteed space:

```

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 . . . . . :

```

This is the job output of our second example. The data set is defined with 5 stripes. SMS allocated on 5 volumes out of the 8 volumes defined in the storage group. Only the data component is striped:

```

DEFINE CLUSTER -
      (NAME (DAWN.KSDSEXT) -
      DATACLASS (KEYEEXT) -
      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
IDC0508I DATA ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0508I DATA ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SBOX31 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME *      IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME *      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 KEYEEXT
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

```

After loading the data set, the LISTCAT output will show the HURBA only on the first volume. For the other volumes, HURBA=0:

```

DATA ----- DAWN.KSDSEXTG.DATA
IN-CAT --- MCAT.SANDBOX.R9.VSBOX11
HISTORY
DATASET-OWNER----- (NULL)  CREATION-----2000.104
RELEASE-----2 EXPIRATION-----0000.000

```

```

ACCOUNT-INFO----- (NULL)
PROTECTION-PSWD----- (NULL)  RACF----- (NO)
ASSOCIATIONS
CLUSTER--DAWN.KSDSEXG
ATTRIBUTES
KEYLEN-----8      AVGLRECL-----300      BUFSPACE-----10240
CISIZE-----4096
RKP-----0      MAXLRECL-----300      EXCPEXIT----- (NULL)
CI/CA-----192
STRIPE-COUNT-----4
SHROPTNS (1,3)      SPEED      UNIQUE      NOERASE      INDEXED      NOWRITECHK
NOIMBED      NOREPLICAT
UNORDERED      NOREUSE      NONSPANNED      EXTENDED
STATISTICS
REC-TOTAL-----450002 SPLITS-CI-----0 EXCPS-----435
REC-DELETED-----0 SPLITS-CA-----0 EXTENTS-----4
REC-INSERTED-----0 FREESPACE-%CI-----10 SYSTEM-TIMESTAMP:
IDCAMS  SYSTEM SERVICES                                TIME: 13:17:48
04/13/00      PAGE      2
REC-UPDATED-----0 FREESPACE-%CA-----10 X'B3E38064F2AD17C3'
REC-RETRIEVED-----0 FREESPC-----98054144
ALLOCATION
SPACE-TYPE-----TRACK HI-A-RBA-----251658240
SPACE-PRI-----1280 HI-U-RBA-----170655744
SPACE-SEC-----0
VOLUME
VOLSER-----SBOX28      PHYREC-SIZE-----4096      HI-A-RBA-----251658240
EXTENT-NUMBER-----1
DEVTYPE-----X'3010200F'      PHYRECS/TRK-----12      HI-U-RBA-----170655744
EXTENT-TYPE-----X'40'
VOLFLAG-----PRIME TRACKS/CA-----4
STRIPE-NUMBER-----1
EXTENTS:
LOW-CCHH-----X'003B0000' LOW-RBA-----0 TRACKS-----1280
HIGH-CCHH-----X'00900004' HIGH-RBA-----251658239
VOLUME
VOLSER-----SBOX29      PHYREC-SIZE-----4096      HI-A-RBA-----251658240
EXTENT-NUMBER-----1
DEVTYPE-----X'3010200F'      PHYRECS/TRK-----12      HI-U-RBA-----0
EXTENT-TYPE-----X'40'
VOLFLAG-----PRIME TRACKS/CA-----4
STRIPE-NUMBER-----2
EXTENTS:
LOW-CCHH-----X'003B000A' LOW-RBA-----0 TRACKS-----1280
HIGH-CCHH-----X'0090000E' HIGH-RBA-----251658239
VOLUME
VOLSER-----SBOX31      PHYREC-SIZE-----4096      HI-A-RBA-----251658240
EXTENT-NUMBER-----1
DEVTYPE-----X'3010200F'      PHYRECS/TRK-----12      HI-U-RBA-----0
EXTENT-TYPE-----X'40'
VOLFLAG-----PRIME TRACKS/CA-----4
STRIPE-NUMBER-----3
EXTENTS:
LOW-CCHH-----X'000A000A' LOW-RBA-----0 TRACKS-----1280
HIGH-CCHH-----X'005F000E' HIGH-RBA-----251658239
VOLUME
VOLSER-----MHLV15      PHYREC-SIZE-----4096      HI-A-RBA-----251658240
EXTENT-NUMBER-----1
DEVTYPE-----X'3010200F'      PHYRECS/TRK-----12      HI-U-RBA-----0
EXTENT-TYPE-----X'40'
VOLFLAG-----PRIME TRACKS/CA-----4
STRIPE-NUMBER-----4
EXTENTS:

```

```
LOW-CCHH-----X'00090000' LOW-RBA-----0 TRACKS-----1280
HIGH-CCHH---X'005E0004' HIGH-RBA-----251658239
```

### **CA size considerations**

CA size calculations is affected by striping. Normally, the CA size is the lesser of the primary or secondary value and 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 may have to be rounded to the next integral of the stripe count. Also, since the maximum stripe count is 16, a CA size of 16 tracks must be allowed to accommodate 16 stripes. Note that CA size maximum is increased to 16 tracks from previous 1 cylinder (15 tracks) allocation.

For striped data sets, all computations for CA size are performed using the equivalent amount of tracks. For example:

- A data set has 7 stripes.
- The equivalent allocation in tracks is (45 30), so 15 tracks are used.
- 15 tracks rounded to the next integral of stripe count is 21. But 21 is greater than maximum of 16 tracks, so CA size is rounded down to 14 tracks.

### **Performance considerations**

Consider the following when using VSAM striped data sets:

- The I/O for a striped data set is as long as the longest I/O in the stripe.
- If you implement  $n$  stripes, it does not mean that your aggregate data rate will be  $n$  times bigger.
- The I/O post to the application is done when all the I/O to the stripes ends. Likewise, if an I/O error occurs in one of the stripes/volumes, the I/O operation ends with an error.
- For striped data sets, you should use SMB to determine the number of buffers, or you should 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. Refer to 2.6.9, "Buffering options" on page 58 for more information.

Using the default BUFND eliminates some of the benefits of striping.

- Using LSR in VSAM striped data set is not rejected. However, you may not see a performance improvement in the way that NSR does.
- The way VSAM striping is done now, you may get better performance than what you specified in the SDR in the storage class. As we have mentioned, SMS computes the number of stripes based on a 4MB/sec rate. But in reality, your DASD performs better.
- For KSDS, it has been experienced that after the fourth stripe, the processing improvement curve flattens. You may not get performance improvements if you define more than 4 stripes.
- Striping does not give a great deal of benefit for data sets accessed directly. However, these data sets will be accessed sequentially during backup, report generating, and so on. Therefore, you may wish to consider striping all your VSAM data sets.

#### Recommendations:

- For striped data sets, you should 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.
- As a ROT for VSAM, do not go beyond 4 stripes — if you do, you will overload the processor.
- Define striping only for sequential processing.

---

## 2.7 VSAM performance management

Here, we cover all the aspects that may affect the VSAM I/O delay time,  $T_w(\text{IO})$ ; and the I/O service time,  $T_s(\text{IO})$ . Also, because VSAM CPU service time,  $T_s(\text{CPU})$ , which is used by VSAM, has some effect on the total response time, we offer suggestions on how to decrease it.

### 2.7.1 Performance scenario using RMF reports

In VSAM performance management, we use an approach in steps that are based on RMF or an equivalent product, assuming you are in WLM goal mode:

1. Look for your most important service class period that is not reaching the goal in the RMF monitor III SYSSUM report:

```

----- 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

```

HTTPS1 is a service class associated with the transactions of an HTTP scalable server. They run under in dispatchable units under enclaves. Its importance (I) is one (the maximum importance), its goal is average response time of 0.80 seconds., its actual response time is 1.6 seconds and the performance index is 2.0 meaning 100% out of the target.

2. Let us look at the Enclave report 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

```

In this report, you see two enclaves belonging to the service class HTTPS1 suffering 88% and 89% of delay, respectively. Some of the delays are measured by WLM in order to be minimized, depending on the performance index of the service class. They are:

- CPU
- I/O (if you are in WLM I/O Management option)
- Storage
- Delay for HTTP Queue Server

In addition to these delays, the RMF Monitor III tracks other delays, such as:



- ENQ delays
  - Operator delays (mount and messages)
  - Subsystem delays (JES, HSM and XCF)
3. Let us now zoom in to these delays in the report, Enclave Classification Data:

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
```

As you can see, detailing the enclave ENC00003, 77% of the delays are caused by I/O delays, that is, being delayed in the UCB or in the channel subsystem (pending time). The Using I/O value is 27%, meaning the application, is executing a channel program (connected or disconnected).

4. Next we will zoom a little more, going to the Monitor III Device Delay report (DEV):

DEV Report

Jobname	Service C Class	DLY %	USG %	CON %	Main Delay Volume(s) -					
					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			

To date, there is no RMF report showing details (at a volume level) about the I/O using and I/O delay of an enclave. Then, you must know the name of the address space where the HTTP transactions are doing I/O. In this case it is the HTTPS000. Here we can see that the volume VSMS15 is responsible for 70% of the delays experienced by the enclave the HTTP transactions.

5. Now, we need to go to the Monitor III DEVN report to see details about the 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

In the DEVN report for volume VSM15, we have 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 percentile distribution of connect, disconnect and pending. If you want to derive how much of connect per I/O operation (AVG CONN TIME) follow this line of thought: "If the range is 100 seconds, the total connect time was 8 seconds. In 100 seconds it was executed (42.1 \* 100) I/O instructions. So, dividing 8 by 4210, we have 2 milliseconds of AVG CONN TIME".

6. Now, the final step using RMF reports is to discover the data set involved in the performance problem. We can do this through the Monitor III DSNV report:

----- Volume VSM15 Device Data -----					
Number:	006C	Active:	68%	Pending:	2%
Device:	33903	Connect:	8%	Delay DB:	1%
Shared:	Yes	Disconnect:	60%	Delay CU:	1%
				Delay DP:	0%
				Jobname	ASID
				DUSG%	DDL%
----- Data Set Name -----					
PROD.KSDS.MARCH.U12		ENC00003		0026	20
		BATQMF2		0089	50

Finally we have all the information — the volume and the data set name (which, incidentally, is a VSAM data set). You should determine the largest figure between IOSQ, pending, connected, and disconnected in your installation (in our example it is disconnect time). Depending on the one you pick, go to the corresponding topic in this chapter, where you will 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, please refer to 2.7.2, “Reduce the number of I/Os” on page 107, where you will find suggestions to eliminate this problem.

## **2.7.2 Reduce the number of I/Os**

The general rule, that the best I/O is the one that is not executed, still applies to VSAM. Therefore, before trying to improve the I/O operation, let us focus on how to avoid these I/Os. The general techniques for doing that are explained in the following sections.

### **2.7.2.1 Buffering**

Buffering is one of the most important VSAM features that can be used to avoid I/Os, and consequently to improve performance. There are two types of buffering: VSAM controlled buffer pools, and Hiperbatch.

#### ***VSAM controlled buffer pools***

VSAM controlled buffer pools are controlled by LSR, NSR, RLS, and GSR. Some of them allocated in the application address space and some in the hiperspace. Refer to 2.6.9, “Buffering options” on page 58.

#### ***Hiperbatch***

Hiperbatch is designed to eliminate the problems caused by:

- Multiple jobs in one OS/390 image requesting data from the same QSAM/VSAM NSR data set simultaneously. Each job causes I/O operations to the device holding 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 passing temporary or short-lived QSAM or VSAM NSR data sets to subsequent jobs. As a job completes, it puts the data 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 (like a data set in the HS ESO). A DLF object contains data from a single data set managed by Hiperbatch. The user (an application program) is connected to the DLF object, and 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 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, 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. As long as 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, and the system would delete the DLF object. To prevent this automatic deletion of an object, 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.

Refer to Figure 21 for the following example:

We have a non-retained data set (here called Master), that is a DLF object. All the reading jobs should be started in parallel (as usual without Hiperbatch). The first job (J1) reads sequentially the record one (R1) and suffers the I/O delay. After the 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, we have only M accesses to DASD, instead of  $M * N$ .

The figure highlights the I/O operations not executed (saved).

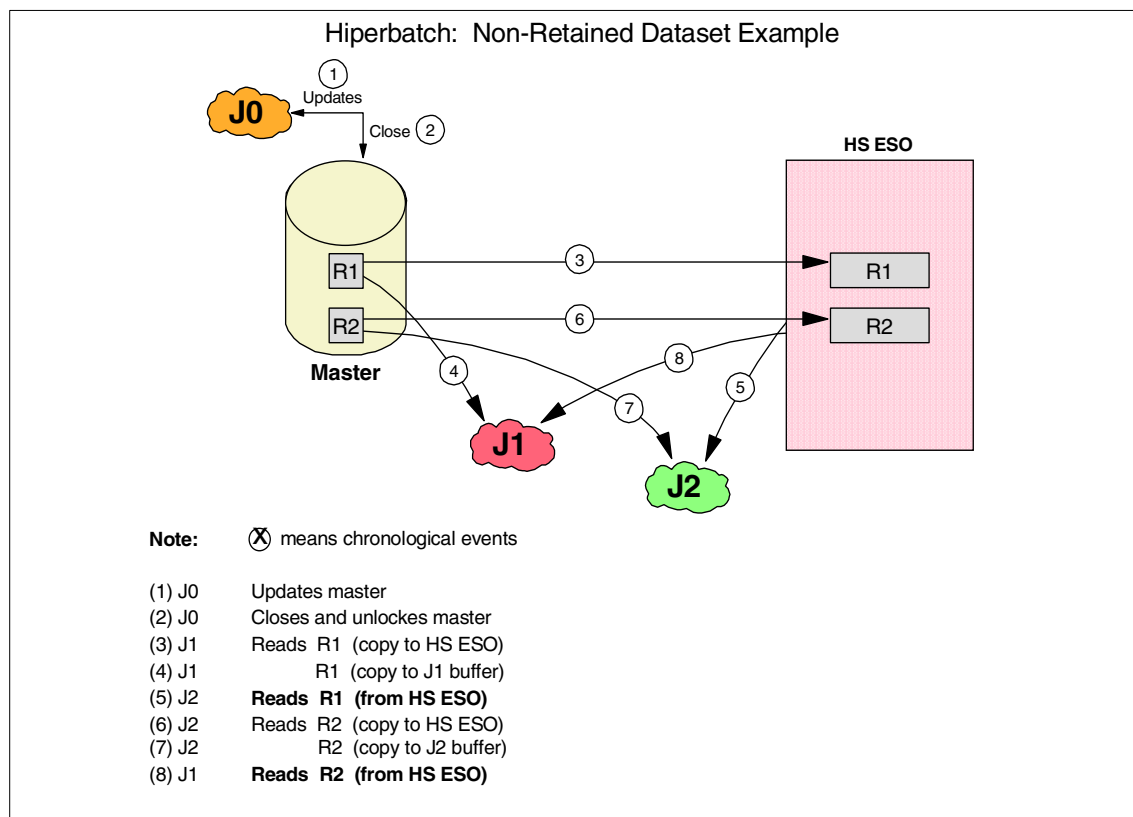


Figure 21. Hiperbatch example

Following are some comments about using Hiperbatch with VSAM:

- VSAM non-shared resource (NSR) access is a requirement.
- Hiperbatch does not support extended format data sets.
- 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.
- If the data set is a VSAM key sequenced data set (KSDS), the DLF object contains only the data component, not the index component.

- To avoid data integrity exposures, Hiperbatch does not process a VSAM data set with shareoptions 3 or 4, even if that data set has been 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 method is best suited for sequential processing.

#### **2.7.2.2 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 and increase the logical record size during an update. CA splits may be minimized by the use of CA free space. Refer to 2.6.4, “Free space” on page 50. CI splits is not a problem because it needs less than five I/Os.

#### **2.7.2.3 Secondary space allocations**

Every secondary allocation implies going through End-of-Volume processing with numerous I/Os in the catalog and VTOC. Refer to 2.6.1, “Allocation units” on page 42. You should require a consistent amount of secondary allocation.

#### **2.7.2.4 Write checks**

Write checks needlessly increase the number of I/O operations. Refer to 2.6.7.1, “Write checks” on page 55. These should be avoided.

#### **2.7.2.5 RECOVERY option**

Use the SPEED option instead, when loading a VSAM file. Refer to 2.6.7, “Initial load option” on page 54.

#### **2.7.2.6 Defer write**

Whenever you can, ask for defer write to save I/O operations. Refer to 2.6.9.5, “Deferring write requests” on page 75, for more information.

#### **2.7.2.7 Use of VSAM data sets**

Is the VSAM data set you are using the right one to address the type of access and organization required by your application? For example, if you are accessing your VSAM ESDS data set only sequentially, maybe replacing it by SAM could be a good idea where performance is concerned. You can save numerous I/O and CPU cycles. Often, you may be paying for a function that you are not using. Following is a list of the data set organizations, in

ascending order of complexity and resource consumption (CPU storage and I/O), where the increased complexity increases I/O and CPU processing:

- VSAM LDS — no indexes, no logical record concept, no physical inclusions or deletions.
- SAM — no indexes, no physical inclusions, logical record concept. Fixed length records perform better than variable length records.
- BPAM (PDS/PDSE) — no indexes, no physical inclusions, logical record concept (fixed or variable), some directory processing (could be very heavy for large non-buffered directory).
- RRDS — no indexes, fixed length records, no physical insertions (just use pre-defined free slots).
- ESDS — no indexes, variable length record, no physical insertions.
- KSDS/VRRDS — indexes, variable length record, physical insertions.

### 2.7.3 I/O wait time (IOSQ) for VSAM files

$T_w(\text{IO})$  has two components respectively: IOS Queue Time and Pending Time. Here, we cover the first one. Let us suppose that you are reading this because the IOSQ time is the dominant factor in the I/O of your VSAM data set.

*IOS Queue Time* is the time waiting for the device availability in the OS/390 operating system. For a non-ESS device, Input Output Supervisor (IOS) does not start an I/O operation to a device if there is a previous one in execution. In this case, the I/O operation is queued in the UCB (a control block representing the device to IOS).

A queue starts to build up when the unique server (device) is utilized above 35%. This rule applies to the IOSQ Time. This utilization can be caused by activity coming from this MVS or from another MVS (in a shared DASD case), or both.

To decrease the IOSQ Time, you can:

- Buy a faster device/channel to decrease the I/O Service Time ( $T_s(\text{IO})$ ) and consequently the utilization (for the same I/O load).
- Decrease the  $T_s(\text{IO})$  by tuning; refer to 2.7.5, “I/O service time (disconnect) for VSAM files” on page 112 and 2.7.6, “I/O service time (connect) for VSAM files” on page 117.
- Increase the importance of the goal or change its numerical value to make it more difficult to be obtained. In consequence, the I/O priority is raised by

the WLM goal mode. This happens when the transaction is not reaching its goal and the major delay is the I/O delay. Be aware that, in this case, you are not improving the I/O in general, but just improving the response time of your favorite transactions (“stealing”).

- Avoid placing 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 guaranteed space to force the index in a specific volume.

#### **2.7.4 I/O wait time (PEND) for VSAM files**

Let us suppose you are concerned because the pending time is the dominant factor in the I/O of your VSAM data set.

*Pending time* is the time waiting in the channel subsystem, after the Start Subchannel instruction, and before the starting of the channel program execution. There is a Statement of Direction for implementing the I/O priority concept within channel subsystem queues. This I/O priority will be the same as defined by WLM goal mode. The pending time is formed by the following delay times:

- All channels reaching the device are busy at same time. Verify the utilization of the non-EMIF channel in your logical partition and the EMIF ones in all logical partitions they serve. To fix this, you may need to avoid EMIF, to have more or faster channels.
- The ESCD port is busy. Possibly you do not have enough ports, or this case is masking a control unit contention.
- The control unit is busy; this happens when the control units have more channel interfaces than the number of concurrent internal data paths. You should decrease the load in the control unit.
- The device is busy because of shared DASD contention caused by I/O activity from the other system or a Reserve CCW. GRS global ENQs can offer a way to decrease such a delay (mainly the GRS start topology in a Parallel Sysplex).

#### **2.7.5 I/O service time (disconnect) for VSAM files**

The I/O Service Time (Ts(IO)) corresponds to the execution of the channel program. There are cases where just increasing the I/O priority does not pay back, for instance, when the queue length is small, or when all the requests in the queue belong to the same transaction. In this case, we must look at the I/O Service Time, which can be divided into two parts: the disconnect time,



and the connect time. Here we cover the disconnect. Let us suppose that the disconnect time is the dominant factor in the I/Os of your VSAM data set.

*Disconnect time* occurs when the channel is not performing activities related to the execution of the channel program. It is disconnected. This means that the target record for a read is not in the cache, and the disk access is a requirement.

The best solution to decrease the disconnect time is to force the use of cache (if the I/O workload is cache-friendly). For a write in the modern control unit, almost all writes are cache hits, so the disconnect time trends to be zero. An exception to that is when a heavy sequential write load causes DASD Fast Write (DFW) bypass the non-volatile cache (NVS) and synchronously store data in the disks. DFW bypass is covered later in this chapter.

Certain controllers can produce a different type of disconnect time, which occurs when the device is busy because shared DASD busy. Here, the normal behavior is to inform the channel, which informs SAP, and the delay time is accounted for under pending time. However, the controller accepts the I/O request from the channel and disconnects it. So, the shared DASD device delay is reported under disconnect time instead of pending, as it should be.

#### **2.7.5.1 Cache highlights**

A cache is a fast storage (no mechanical movement) located in the controller with two functions: to minimize access to disks (by having cache hits) and to serve as a *speed matching buffer* to synchronize elements with different speeds (like channels and disks) in a cache miss. In this discussion the term disk does not have the same meaning of DASD. Disk implies the RAID media, where data is stored in fixed block architecture (FBA) blocks through an SSA or an SCSI protocol as used by modern controllers. DASD is the logical 3390/3380 device as perceived by you, your application, and your MVS operating system.

In order to have random cache hits (saving disk access) for reads and writes, the I/O workload must access the same data. Typically there are two types of hits, when the application revisits data:

- Exactly the same logical record in a CI is already in cache.
- The same data CI is already in cache because another logical record was previously accessed.

For sequential access, it is important to say that 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 utilized.

Random reads and random writes have a completely different stories.

Refer to 2.7.5.3, “Decrease disconnect time VSAM access” on page 116. There we discuss what to do with this type of access. Before we introduce some specific recommendations about VSAM and the cache, it may be a good idea to review the basic cache concepts, in B.1.3, “DASD cache concepts” on page 226.

#### **2.7.5.2 VSAM hints to decrease disconnect time by using cache**

If the disconnect time of your key VSAM data sets is above two milliseconds, read this topic. In our example, refer to item 6 of 2.7, “VSAM performance management” on page 103.

RMF is an IBM program product which measures the OS/390 system. RMF has three Monitors together with a Postprocessor.

Each Monitor has a Data Gatherer and a Data Reporter. 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, measuring system variables.
- Monitor II is interactive, showing data mainly about address spaces.
- Monitor III is interactive, showing the major delays suffered by the transactions. In the explanations given in this chapter, we use Monitor III data.

For this performance scenario, let us look at the RMF Monitor III Volume Cache report.

## Volume Cache report

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
CFW Bypass :	0.0	Bypass:	0.0	Hits :	0.0	Write Prom:	0.0
DFW Inhibit:	0.0						

There are no cache reports per data set in RMF. You can use the volume report to reach your conclusions.

Look at the ICL value. If it is consistently non-zero, this may mean:

1. It is an SMS data set, and its cache attribute is never-cache. Then, change its MSR value to always-cache or may-cache.
2. It is an SMS data set, its cache attribute is may-cache. Then change it to always-cache.
3. It is a non-SMS data set, and IDCAMS does not properly set the cache attributes in the volume (check whether CACHE and DFW are active in the report header). Then, change it to normal and DFW.

If, after making the modifications listed above, the disconnect time does not get better (smaller); this means that the data set is cache-unfriendly. In this case, undo modifications 1 and 2 above, and read 2.7.5.3, "Decrease disconnect time VSAM access" on page 116. If ICL is very close to zero, read the next topic.

- If the access is dominantly for writes, with READ% consistently below 50%, take a look in the WRITE HIT% column. Pick up the value corresponding to the highest value in Write RATE (between NORM and SEQ). If NORM is the larger, you have a random access. Compare the FAST value with RATE. If they are not the same, this implies that SMS is not always using DFW cache mode. If the numbers are the same (or very close) check the HIT% value. It must be higher than 95% (for a non-RVA device, almost 100% of the random writes should be hits).

- If SEQ is the larger value (you are writing sequential) and HIT% less than 95%, take a look in the DFW Bypass figure. If consistently non-zero, this means:
  - The write sequential load saturated the NVS and DFW bypass occurred. That is, the records are sent synchronously from volatile cache to disks, and this time is included in disconnect time.
  - In this case your workload is the cache-unfriendly type. Refer to 2.7.5.3, “Decrease disconnect time VSAM access” on page 116.
- If the access is dominant in reads, with READ% consistently above 50%, look in the READ HIT% column. Pick up the value corresponding to the highest value in READ RATE (between NORM and SEQ). If NORM is the larger, you have a random access (if not is sequential). If the corresponding READ HIT% is less than 80%, you have found your problem, or you may have one of the following problems:
  - A low sequential Read hit. This means that the speed of the channel (pushed by the application) is higher than the disk speed. In this case refer to 2.7.5.3, “Decrease disconnect time VSAM access” on page 116.
  - A low random Read hit. This means that your reads are candidates for using cache, but they are not currently doing so. Let us investigate this in more detail:
    - Possibly your access is sequential, and the KSDS has many CI splits, impeding the controller in recognizing the sequential pattern; then there is no look-ahead and the cache is treated in LRU mode. Reorganization may be an answer. Refer to 4.1, “Reorganization considerations” on page 181.
    - If you are accessing data randomly, be sure that SMS is using record level cache (RLC) for reads. It avoids polluting the cache with the rest of the logical 3390/3380 physical track.
    - If you are accessing data randomly, try to use a smaller CI for read data or less free space in the CIs. It avoids polluting the cache with other non-referenced logical records or free space.

### **2.7.5.3 Decrease disconnect time VSAM access**

Before we start a logic flow to guide you along the cache-unfriendly performance problem, let us say that before the appearance of the RAID disks, the data set placement was key to avoid high disconnect time. Refer to 2.5, “VSAM rule-of-thumb (ROT) mode” on page 41 for the list of out of date performance recommendations due to the implementation of new disk technology.

If cache cannot help your VSAM data set, you need to reduce the contention on the disks. Here are some suggestions.

- Reducing I/O rate (demand) against the controller. This can be done via:
  - Either compression or use of smaller CIs for random processing.  
For compression, refer to 2.6.10, “Data compression” on page 84. For use of smaller CIs, refer to 2.6.3, “Control interval size” on page 48.
  - Reorganization of the data set, if there is plenty of free space in the CIs. Refer to 2.7.6, “I/O service time (connect) for VSAM files” on page 117.
  - Reducing the number of active data sets in the controller.
- Decrease the number of writes in the controller by moving data sets, to avoid the effect of write penalty.

### 2.7.6 I/O service time (connect) for VSAM files

*Connect time* means the period of time when the channel is transferring data from or to the cache or exchanging control information with the controller.

Often, a high connect time is good, meaning that a great deal of data is being transferred in just one I/O operation, through one CCW or many CCWs. If this is your experience, examine a more detailed report (generally based in GTF with CCW analysis) looking for the channel program.

However, if this is not the case, here are some recommendations:

- Use plenty of buffer space in the buffer pool for sequential processing:  
By using many buffers in a sequential processing, you are decreasing the number of SSCHs instructions. For every I/O operation starting, there is a standard conversation between the channel and the controller. If you decrease the number of SSCH, but transfer the same amount of data, you save connect time. Refer to Table 5 on page 68 and Table 10 on page 90.
- Use data compression in CPU:  
There is a difference between compaction and compression. Compaction has to do with the simple task of extracting repetitive characters from a text. Compression is a much more elaborate task, where dictionaries may be used to obtain the best result (like the Ziv-Lempel algorithm). These dictionaries contain the most repeated set of characters found in the text and their respective compressed substitution. Refer to 2.6.10, “Data compression” on page 84.
- Use the ECKD extended format:

ECKD extended format (also called extended format) is a technique that affects the way count key data is stored in a 3390/3380 logical track. It improves performance of an I/O operation, by decreasing Ts(I/O) with a better channel program. It is recommended that, if time permits, you convert your data sets to extended format to get this better performance. Refer to Table 11, showing the following results from our tests of random processing: extended format versus non-extended format data sets.

Table 11. Random processing: extended format vs. non-extended format data sets

Ext format	Buffering	EXCPs	Connect time (sec)
No	Default	772057	549
Yes	Default	771265	541

Here, for random processing, you can see some decrease in the number of EXCPs and in the connect time. However, the big performance appeal of extended format is that it allows the use of strong performance capabilities such as striping, compressing, and SMB.

- Small CIs for random access:

This avoids bringing unneeded logical records into memory. Here there is a recommendation. If the cluster is going to be accessed sequentially and randomly, the CI size should be made smaller.

To solve the performance problem in sequential access, you can define many data buffers, thus causing VSAM to chain CIs in just one channel program. Refer to 2.6.3, "Control interval size" on page 48.

- Less free space in the CIs:

The existence of a significant amount of free space in a CI, mainly along a sequential process, may increase the I/O connect time. This free space may be caused by an excess in the definition of the data set, or by CI and CA splits. Remember that any CI with a logical record is moved to storage in a sequential read. This is not true with totally free CIs.

**Note:** We are *not* saying that all insertions increase the average free space per byte; just the ones which cause splits do that.

The solution here is a reorganization. Refer to 4.1, "Reorganization considerations" on page 181.

- Parallel VSAM I/O operations:

Sometimes, if you cannot decrease the Ts, you may increase the ETR by introducing parallelism. There are two types of I/O parallel processing:

- Within a transaction:

The VSAM option which allows parallelism within a transaction or task (when accessing a VSAM cluster) is data striping. Refer to 2.6.11, “Data striping” on page 92.

- Between transactions:

There are VSAM options which allow parallelism between transactions or tasks, when accessing a VSAM cluster, such as:

- STRNO: In multiple string processing, there can be multiple independent Request Parameter Lists (RPL) within an address space for the same data set. The data set can have multiple tasks that share a common control block structure. There are several ACB and RPL arrangements to indicate that multiple string processing will occur:
- 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 using asynchronous requests.
- Multiple RPLs are active against the same ACB using synchronous processing with each requiring positioning to be held.

Refer to DFSMS/MVS Using Data Sets, SC26-4922 for more information on multiple string processing.

For more information on Share options refer to 2.6.6, “Share options” on page 53.

For more information on SmartBatch options; refer to 2.8, “VSAM and SmartBatch” on page 121.

In ESS, the PAV and Multiple Allegiance features implement parallelism within and between transactions.

### **2.7.7 How to decrease VSAM CPU time**

The major theme of this chapter is to decrease the Tr(I/O) for VSAM I/O operations, in order to decrease the response time of key transactions using VSAM data sets. However, chances are that the enhancements introduced by your changes may shift the bottleneck to the CPU side. It is known that:

- 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 CPU is extremely busy, causing huge CPU delays.

#### 2.7.7.1 VSAM buffering

The adequate use of buffering (mainly for direct access) may result in savings in the CPU usage, as you can see in Table 12.

Increasing the number of buffers decreases the number of EXCPs, that is, the number of executed I/O operations. For the same amount of logic in your code, the CPU time that you spend is a direct function of the number of EXCPs. Because in our lab, the read data is not processed, we can say that:

- SRB time is caused by I/O interrupts (back end) processing.
- TCB time is the I/O operation preparation and buffer pool management.

It is clear that increasing the number of buffers means the management cycles dominate the savings in the I/O operations. Refer to B.1, “Our laboratory” on page 223.

Table 12. NSR — read sequential varying the number of buffers

Data buffers	Index Buffers	EXCPs	SRB time (Secs)	TCB time (secs)
Default=2	Default=1	37735	0.95	2.7
10	1	7641	0.28	1.6
30	1	2549	0.15	1.4
181	1	466	0.10	1.71

Also, the use of better techniques to manage your buffer pool, such as BLSR or system management buffers (SMB), can save enormous CPU cycles. See Table 13, which shows data obtained from our lab.

Table 13. Direct access: benefits of using SMB — updates and insertions

Ext format	Buffering	EXCPs	connect time (sec)	CPU time(sec)
No	Default	772057	549	52.54
No	BLSR	293960	255	24.83
Yes	SMB	79741	75	11.04



Ext format	Buffering	EXCPs	connect time (sec)	CPU time(sec)
Gain using SMB (%)		90	86	79

#### 2.7.7.2 Use of extended format

The use of extended format data sets can save a consistent amount of CPU time (TCB plus SRB). Consider the values shown in Table 14.

Table 14. Direct access: benefits of using SMB — updates and insertions

Ext format	Buffering	EXCPs	CPU time(sec)
No	Default	772057	52.54
Yes	Default	771265	45.67

Refer to 1.5, “Extended format data set” on page 18, to get more information.

#### 2.7.7.3 Compression

Compression of data can really increase the CPU time in your program.

#### 2.7.7.4 Track versus cylinder allocation

It is not true anymore that, when allocating VSAM data sets in cylinders, you consume less CPU than using track allocation. The validation if the channel program is crossing an extent boundary is done by the channel. It is informed about the correct extents through the Define Extent CCW.

---

## 2.8 VSAM and SmartBatch

IBM's SmartBatch for OS/390 addresses the major problems facing businesses with batch workloads: lengthy processing time, contention for resources, and unbalanced workloads. SmartBatch provides an integrated suite of functions to solve these problems.

### 2.8.1 SmartBatch highlights

SmartBatch uses parallelism, workload distribution, and I/O optimization to reduce your company's batch processing time and balance the workload across your system or Parallel Sysplex. By running jobs and job steps in parallel, SmartBatch can dramatically reduce the elapsed time of your job streams. Further elapsed time reductions result when SmartBatch is used to minimize I/O resource usage and automatically route job steps to available OS/390 images in the Parallel Sysplex.

In summary, SmartBatch:

- Allows two or more jobs that formerly ran serially to run in parallel.
- Allows individual job steps in a multi-step job to run in parallel.
- Routes job steps to the sysplex image that is best suited to execute the step; analyzes the steps in a batch job and schedules each step to run on an available image based on image attributes and current capacity.
- Reduces the number of physical I/O operations, when possible, by transferring data through image storage rather than DASD or tape.
- Optimizes I/O for tape drives and DASD.

## **2.8.2 SmartBatch components and VSAM**

SmartBatch has four components, some of them can be exploited in order to improve performance of batch jobs accessing VSAM data sets:

### **2.8.2.1 Batch Accelerator**

The Batch Accelerator component of SmartBatch continuously monitors your installation's batch workload for jobs that meet particular selection criteria. Batch Accelerator can run a job's individual steps in parallel (job step splitting), routing each step to the image in which the capacity satisfies the execution requirements of the step (job step targeting).

Then, any batch job using VSAM can get improvements in the elapsed time due to this enlarged parallelism

### **2.8.2.2 Data Accelerator**

The Data Accelerator component of SmartBatch allows you to tune the processing of your application's I/O requests. Data Accelerator chooses data set I/O requests for optimization based on user-defined selection criteria. If a data set I/O request matches the characteristics specified in a selection definition, Data Accelerator uses the performance options established in the action definition to provide I/O performance improvements.

Data Accelerator intercepts data set I/O based on criteria specified in data policy definition. A data policy defines when (in selection definitions) and how (in action definitions) I/O performance processing occurs.

Data policy definitions associate one or more data set characteristics with a Data Accelerator response. When a data set meets the criteria specified in a selection definition, Data Accelerator uses the response specified in the action definition to provide I/O performance improvements.

Data Accelerator provides selection criteria that allow you to control when I/O performance processing occurs. These criteria include the following items:

- Type of file to be used (VSAM, non-VSAM, or both)
- Data definition name (ddname)
- Data set name (dsname)
- Job name
- Program name
- Step name
- Procedure step name
- SMS data class name
- SMS storage class name
- SMS management class name
- Security product group ID
- Security product user ID
- Execution job class
- Time of day

A more complex selection definition would intercept for I/O performance processing all VSAM data sets that have a specific high-level qualifier and that are being processed during a specific time range by a specific job. Following are the options that you can apply to the data set:

### ***Action option***

When specified in an action definition, the action option includes or excludes specific jobs or data sets from I/O performance processing.

- The action option has the following possible values:
  - INCLUDE: Provide I/O performance processing for this data set.
  - EXCLUDE: Don't provide I/O performance processing for this data set.

- VSAM deferred writes option

The VSAM deferred writes option allows you to specify whether Data Accelerator writes requests as they occur, or places the write requests in a buffer to reduce physical I/O. Refer to 2.6.9.5, "Deferring write requests" on page 75 for more information on the subject

- VSAM SHAREOPTIONS(3 3) support option

The VSAM SHAREOPTIONS(3 3) support option allows you to specify whether Data Accelerator provides I/O performance improvements to VSAM data sets that are allocated as SHAREOPTIONS(3 3). Because Data Accelerator converts NSR to LSR when possible, the index buffer invalidation process of SHAREOPTIONS(3 3) data sets might not occur as expected.

Allow Data Accelerator to provide I/O performance improvements only if you know how your application is using the data set allocated with SHAREOPTIONS(3 3).

- VSAM LSR eligible option:

By default, Data Accelerator switches to LSR processing for VSAM data sets. This is one of the advanced techniques that Data Accelerator uses to provide performance benefits. Except for the performance benefits that are realized, an application is seldom aware of the switch to LSR processing. If your application is unable to tolerate the switch to LSR processing, However you can change the default so that Data Accelerator does not switch to LSR processing.

- Actual I/O count option:

The actual I/O count option allows you to specify whether Data Accelerator reports to SMF the number of I/O requests made by an application or the actual number of I/O requests made by Data Accelerator. The actual I/O count option is available for access to non-VSAM data sets only.

- Dynamic region adjustment option:

The dynamic region adjustment option allows Data Accelerator to modify region specifications automatically to compensate for additional I/O-related storage areas obtained on an application's behalf.

For example, if you have set the REGION parameter to 800 KB and Data Accelerator requires 100K for I/O performance processing, the REGION parameter for the job is dynamically adjusted to 900 KB when you specify Y for this option. The dynamic region adjustment option provides the following benefits:

- Eliminates the need to increase the region value on the JOB or EXEC JCL statements to prevent S878 and S80A abends when Data Accelerator is used. Without the dynamic region adjustment option, attempts to obtain additional storage on the application's behalf can result in storage-related abends if a small region value is specified in the JCL or SMF IEFUSI region control exit. With the dynamic region adjustment option, storage-related abends are minimized.
- Prevents the application from experiencing a decrease for storage obtainable with GETMAIN when Data Accelerator is used. *Without* the dynamic region adjustment option, the increased amount of I/O-related storage obtained on the application's behalf decreases the amount of storage that the application can obtain for its own use.

- Resources usage bias option:

The resource usage bias (RUB) option allows you to specify how Data Accelerator is to obtain and use virtual storage buffer resources when it provides I/O performance processing. The option determines the type of I/O performance techniques used and the relative amount of user and Data Accelerator buffers allocated. You specify the resource usage bias option as a numeric value.

A special RUB value, 99, is provided to indicate that Data Accelerator is to dynamically select the optimum RUB value for the:

- Type of processing being performed
- System resource availability at the time of processing.

For most situations, allowing Data Accelerator to select the RUB value provides the best performance benefits without over committing system resources. Under some circumstances, however, you might want to override Data Accelerator's choice to favor CPU or virtual storage for a given application or environment.

You might need to experiment with the RUB values to determine the optimum value for certain special applications or environmental conditions

For access to VSAM data sets, Data Accelerator manipulates buffer pools to obtain I/O performance improvements. This section explains how to select an appropriate RUB value for use with VSAM data set access. The RUB value for VSAM access adjusts the size of the buffer pool. Your available resources will determine your selection for the global setting.

- RUB=01-10 — As you move from 01 to 10, the size of the buffer pool is increased. You might need to experiment with several values before you determine the optimum value for your applications.
- RUB=99 — Data Accelerator dynamically selects an appropriate RUB value based on the following factors:
  - Open intent
  - CI size
  - Buffer location
  - Whether the addition of buffers poses a storage-shortage risk
  - Whether the CPU is currently constrained
  - Whether the paging subsystem is currently constrained.

- Write Statistics Report Option:

The write statistics report option allows you to specify whether Data Accelerator generates I/O performance statistics reports when SAM and non-VSAM data sets are selected for performance processing. The reports are written to the JES system message log.

- Message Level Option:

The message level option allows you to indicate what types of diagnostic messages you prefer.

---

## Chapter 3. Recovery of VSAM data sets

In the complex environment of computing today, it is almost a miracle that everything runs as expected. The possibilities for error are enormous, covering both hardware and software. S/390 is a very stable platform, which is capable of performing automatic recovery at all levels to hide the problem while attempting to solve it for the end user. Together with the recovery action, numerous records are captured describing the error.

Most likely, you have heard user complaints like these:

- Everything was working yesterday; we have changed nothing since then.
- We are starting to get strange error messages, and the explanations in the manual are meaningless.
- We cannot access the data, and do not know where to turn for help.

Your data is one of the most precious assets of your company. The challenge, in order to save your data, is to use all the error information to answer these three questions:

- What happened to cause this problem?
- What must you do in order to recover your data now?
- What must you do to avoid these problems in the future?

In this chapter, we describe how to handle common VSAM recovery situations, and point out where to get more information. Note that we are not covering VSAM component problem determination; this is beyond the scope of our book. For example, we do *not* discuss the recovery of:

- ICF catalogs. Refer to Integrated Catalog Facility Backup and Recovery, SG24-5644-00
- VSAM data sets in Record Level Sharing (RLS) mode. Refer to CICS/VSAM Record Level Sharing: Recovery Considerations, SG24-4768.
- VSAM data sets accessed by subsystems as CICS, DB2
- Abends in VSAM (record management, catalog management), OPEN, CLOSE and EOVS or any OS/390 component.
- VSAM data sets in non-RAID controllers.

---

### 3.1 Basic recommendations

Before starting, we would like to offer some basic recommendations:

- Back up all your critical data by using methods such as:
  - SMS management class with ABARS for backups, to allow restore of your data in the case of a hardware error and/or application error, and for use in disaster recovery.
  - Remote copy for disaster recovery.
- Familiarize yourself with getting the required documentation, such as logs, dumps, traces, and messages associated with errors.
- Keep your system at a current maintenance level. Apply PTF selective service in your OS/390 and DFSMS/MVS, especially the ones associated with VSAM. To find fixes associated with broken VSAM data sets, use the search word 'dsbreaker' in either `retain` or through `ibmlink (askq)`

---

### 3.2 VSAM recovery information sources

Following is a list of documents and Internet sites where you can find assistance in the VSAM recovery arena:

- VSAM Knowledge Database, which is an interactive diagnostic tool. It is a question-and-answer driven knowledge data base that resides on the DFSMS/MVS Technical Support Web site under "Technical Database" at the following URL:  
  
`http://knowledge.storage.ibm.com/`
- APAR, II08859 which has a methodology to assist you in fixing broken VSAM clusters
- DFSMS/MVS DFSMSdfp Diagnosis Reference, LY27-9606-05

---

### 3.3 How to back up VSAM data sets

Here, we discuss the various methods of backing up VSAM data sets, both while open and while closed.

#### 3.3.1 IDCAMS EXPORT and IMPORT

First, we would like to explain some of the unique characteristics of the EXPORT and IMPORT commands:

- The EXPORT command either exports a cluster or an alternate index or creates a backup copy of an integrated catalog facility catalog.



Exporting means to store 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.

Following 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, and 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
/*
```

- **IMPORT** is the opposite operation to **EXPORT**. Here, you reload the cluster or AIX data and recatalog its catalog information in an active catalog.

Following 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, 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)
/*

```

### 3.3.2 Backup-while-open concepts

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), or 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. The options are:

- TYPECICS

Use the TYPECICS parameter to specify BWO in a CICS environment. For RLS processing, this 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.

**Note:** 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 have been 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 performing defrag, dump, or restore operations.
- DFSMSHsm, used for incremental backups and aggregate backups of a data set, invokes DFSMSdss to perform BWO.

In the catalog are the following fields referring to BWO:

- BWO — Data set is enabled for backup-while-open.
- BWO STATUS — Indicates the status of the data set. Status can be:
  - 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 timestamp that indicates the time from which forward recovery logs have to be applied to a restored copy of the data set

---

### 3.4 Space Constraint Relief parameter (fewer X'037' abends)

Before we start explaining more complicated recovery situations, let us address a common abend situation prior to DFSMS/MVS 1.4.

Users occasionally encounter data set allocation or extension failures (the X37 type of abends) because there is not enough space available on a volume to satisfy the request. Incidentally, VSAM does not externalize an X37 abend. You recognize the out-of-space condition by the message IEC070I 203-204, where 203 is the reason code. You find the explanation in the message IEC161I, return code 203, when no secondary space allocation quantity was specified. Return code 204 is issued when a new extend was attempted, but the maximum number of extents was reached.

SMS alleviates this out-of-space situation to some extent by performing volume selection, checking all candidate volumes before failing an allocation.

With DFSMS/MVS 1.4, you can also use the Space Constraint Relief and Reduce Space Up To (%) attributes in the data class to request that an

allocation be retried if it fails due to space constraints. SMS retries the allocation by combining any of the following:

- Spreading the requested quantity over multiple volumes
- Allocating a percentage of the requested quantity
- Using more than 5 extents

Space Constraint Relief specifies whether or not to retry an allocation that was unsuccessful due to space constraints on the volume. Note that allocation is attempted on all candidate volumes before it is retried. This attribute applies only to system-managed data sets, and is limited to new data set allocations, and while extending the data set on new volumes.

Out-of-space conditions are now further reduced for new volume processing of SMS-managed data sets. VSAM and non-VSAM data sets can now acquire up to 123 extents instead of just 5 extents on a volume. Multivolume VSAM data sets can now have a maximum of 255 extents across volumes for each component, but no more than 123 extents per volume.

Two new parameters, Space Constraint Relief and Reduce Space Up To (%), are added to the SMS data class definition for this support.

Reduce Space Up to (%) specifies the amount by which you want to reduce the requested space quantity when the allocation is retried. You must specify Y for the Space Constraint Relief attribute. Valid values are 0 to 99.

If you specify Y for Space Constraint Relief, SMS begins the retry process. This is a one or two-step process, depending on the volume count you specified. For JCL allocations, SMS determines the volume count by taking the maximum of the unit, volume, or volser count. If these are not specified, SMS picks up a volume count from the data class. If there is no data class, SMS defaults the volume count to one:

- If the volume count is one (one-step process):

SMS retries the allocation after reducing the requested space quantity based on the Reduce Space Up To attribute. SMS simultaneously removes the 5-extent limit, so that SMS can use as many extents as the data set type allows

- If the volume count is greater than one (two-step process):

First, SMS uses a best-fit volume selection method to spread the primary quantity over more than one volume (up to the volume count). If this fails, SMS continues with the best fit method after reducing the primary quantity and removing the 5-extent limit.

If you request Space Constraint Relief but do not specify a percentage value (either 0 or blank), SMS does not reduce the requested space quantity. This implies your application cannot tolerate a reduction in the space to be allocated, so you want to remove the 5-extent limit, thereby allowing SMS to use more than 5 extents.

For extends to new volumes, Space Constraint Relief is strictly a one-step process. If regular volume selection has failed to allocate space, SMS reduces space or removes the 5-extent limit, but does not try the best-fit method.

The number of extents vary depending on data set type, as follows:

- Non-VSAM, non-extended format data sets, up to 16 extents on the volume
- Non-VSAM, extended format data sets, up to 123 extents
- PDSE, up to 123 extents on the volume
- VSAM data sets, up to 255 extents per component but only up to 123 extents per volume per component

When you request Space Constraint Relief in one or more data classes, you might notice any of the following:

- Very large allocations might succeed if a sufficiently large volume count is specified in the data class or through JCL.
- Existing data sets might end up with less space than initially requested on extents.
- The space allocated for new data sets might be less than requested.
- The number of extents used during initial allocation might result in fewer extents being subsequently available. For example, if the primary space allocation uses 10 extents when allocating a physical sequential data set, then only 6 extents are left for allocation of the secondary quantity.
- You might observe fewer X37 abends.

---

### **3.5 IDCAMS recovery commands**

IDCAMS has three important commands used to recover VSAM clusters and catalogs. They are: EXAMINE, DIAGNOSE, and VERIFY.

### 3.5.1 EXAMINE command

EXAMINE is an IDCAMS command that allows the user to analyze and collect information on 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 basic catalog structure (BCS) of an ICF catalog. This function cannot share a cluster opened for output or update by other task.

The EXAMINE function executes two possible tests:

- Test the Index portion (INDEXTEST), the default.

Evaluates the full index component of the KSDS/VVRDS cluster by cross-checking vertical and horizontal pointers contained within the index control intervals, and by performing analysis of the index information. Usually is a medium to small resource consuming task.

- 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 record and control interval integrity, free space conditions, spanned record update capacity, and the integrity of various internal VSAM pointers contained within the control interval. Usually this is a long resource consuming task.

You can also limit the number of error messages generated (ERRORLIMIT) by EXAMINE. Refer to 3.7, “Broken data sets” on page 139, for information about what to do whether EXAMINE is reporting errors in the Index or Data portion. Refer also to B.5, “IDCAMS Examine messages” on page 251.

### 3.5.2 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, or invalid relationships between entries.

Because the DIAGNOSE command checks the content of the catalog records, and the records might, for example, contain damaged length field values, there is a possibility that the DIAGNOSE job will abend. For detailed information on using DIAGNOSE, see DFSMS/MVS Managing Catalogs, SC26-4914.

Also refer to 3.7, “Broken data sets” on page 139, to get information about what to do whether DIAGNOSE is reporting errors in the BCS or VVDS.

### 3.5.3 VERIFY command

Some clarification of the word VERIFY is necessary in many cases. 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.

What record management does on receiving this macro is to start reading the data set CI by CI starting with the current high used RBA value stored in the ARDB. If the data set is a KSDS, then both the index and the data will be VERIFY'd.

VERIFY is also an IDCAMS command. In this case, IDCAMS will open the requested data set for output, issue the record management VERIFY macro and then close the data set. When the data set is closed, VSAM Close processing will use its catalog interface to call Catalog to update the VVR information from the new information in the VSAM control blocks (AMDSB and ARDB mostly). It is important to understand that IDCAMS is neither updating VSAM control blocks nor 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, the same with RLS data sets. Paths 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 (located in the VVR AMDSB cell). Among other fields, we have:

- 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

Refer to 3.9, “IDCAMS LISTCAT output fields” on page 166, for more information.

#### 3.5.3.1 Implicit VERIFY versus explicit VERIFY

Another area that is confusing when talking about VERIFY involves the terms “explicit VERIFY” and “implicit VERIFY”. An explicit VERIFY refers to the

user initiating the VERIFY himself by issuing an IDCAMS VERIFY job against the data set. An implicit VERIFY refers to VSAM Open processing internally issuing the VERIFY macro against the data set when it determines that the data set was not previously closed properly; this means that a previous job which had the data set open for output has either abended or failed to close the data set for some reason.

Open processing uses a bit in the catalog to determine this. 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 (enqueueing against the data set name) to determine if the data set is currently open for output. If not, then it concludes that the last close was abnormal and issues the VERIFY before completing the Open process. It also issues IEC161I messages (rc56 and rc62) to indicate that it has done this. All Open jobs against the data set will get this implicit VERIFY and the associated messages until the data set is opened for Output.

#### **3.5.3.2 Implicit VERIFY problems**

Here we discuss a couple of problems with the implicit VERIFY that are not inherently obvious.

Normally, a VERIFY does not take much overhead 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), then the VERIFY could take many minutes, as it basically reads the entire file. This could also be a severe impact 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), then all TCBs are abended, and all files open for output at the time will have the 'open for output' bit on in the VVR. This will mean they will all need to be implicitly VERIFY'd as the region is brought back up. If this is the case, you need to have a procedure in place to explicitly VERIFY the important files in the case of an abend situation so that the system can come up and the important applications can start running immediately.

Another situation is where you are sharing DASD, but do not have all systems in a GRS ring. If you open the data set for Input, on the system that does not have the data set open for Output, then Open processing will find the 'open for output' bit on, but not the ENQ. Therefore, it will conclude (incorrectly) that the data set was not closed properly and do an implicit VERIFY. Most customers actually like this because it means that the HURBA will be current



on their input job and they will be able to read all the records. The problem happens when they put this system into the ring and then no longer get the implicit VERIFY and cannot read all the records in the file. Unfortunately, this is working as designed, and they must either add a VERIFY to their read program or do a CLOSE or TCLOSE on the Output job.

There is also a common misconception that VSAM will always do an implicit VERIFY on the file if it was abnormally closed. This is true for files that have SHR(1 3) or SHR(2 3) and SPEED has not been specified when the cluster was defined (if, of course, the user has all his shared systems set up with GRS correctly). But in the case of SHR(3 x) and SHR(4 x) this is not true. With cross system shared data sets the "open for output bit" is not reliable. This is because if two systems are open for output at the same time and one closes, the bit gets turned off. If the second one then abends, the bit is still off and there is nothing to tell VSAM Open that the last job that had the data set open for output abended. Therefore, VSAM Open cannot reliably VERIFY the data set. That is why the user is responsible for issuing the VERIFY macro after Open on data sets with SHR(3 x) and SHR(4 x).

#### ***ACTION=REFRESH***

One addition which was made to VSAM quite some time ago was the ACTION=REFRESH parameter of the VERIFY macro. Without this parameter, the VERIFY macro will only read up to the current HARBA of the data set (as set in the ARDB control block). Since this value could have changed since the last time the data set was opened by the current job, this facility allows the control blocks for a data set to be updated to reflect the current structure of the data or index (from the values in the catalog/VVR). To accomplish this, record management will call EOVS and EOVS will use its catalog interface to update the in storage control blocks (i.e. AMDSB, ARDB, EDB) 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 Verifies, possibly causing further OPEN errors.

---

### **3.6 Useful documents**

Here a list of the documents that can help you answer the three "Whats" of problem resolution (which we introduced at the beginning of this chapter):

- VSAM messages explaining the problem, usually starting with the IDC prefix. Those messages are usually produced (through the WTO macro)

by the caller of VSAM record management or catalog manager routines. They are also produced by the caller of a system service routine, for example, Open, Close, EOVS, or IOS. These messages are associated with a return code different from zero produced by the called routines.

In section B.4, “Symptoms (messages) from a broken data set” on page 245, we list these messages. Here are a few of the most common:

- IDC3302I — Action error
- IDC3308I — Duplicate records
- IDC3314I — Out of sequence records, missing records, duplicate records, no record found
- IDC3351I — VSAM logic I/O error RC156, RC24, or RC32
- IDC3350I — No record found or incorrect length
- IDC3009I — VSAM CATALOG RETURN CODE IS return-code — REASON CODE IS IGGOCLaa — reason-code
- OS/390 system messages, generated by OS/390 components describing errors beginning with the IEA prefix, associated with data set allocation, master scheduler functions, and RTM. The IOS prefix is for IOS functions. Usually those messages follow abends:
  - IEC070I — RC32, RC202, RC104, or RC203
  - IOS000I — Command reject (IOS errors in general)
- IDCAMS LISTCAT; refer to 3.9, “IDCAMS LISTCAT output fields” on page 166.
- IGGCSIVS a program for SYS1.SAMPLIB accessing Catalog Search Interface (CSI) produces a list of data set names defined in a given catalog that reside on a specific volume. Refer to 4.3, “Catalog Search Interface” on page 193, for more information.
- SMF records associated with VSAM; refer to 3.11, “SMF record types related to VSAM data sets” on page 174. You can use our SMF 64 sample program as described in that topic.
- The output message of IDCAMS EXAMINE; refer to 3.5.1, “EXAMINE command” on page 134.
- The output messages of IDCAMS VERIFY; refer to 3.5.3, “VERIFY command” on page 135.
- System LOGREC messages.
- A GTF CCW trace.
- DITTO/ESA output; refer to 1.10.3, “DITTO/ESA” on page 29.

- DFSMSdss PRINT command; refer 3.10, “DFSMSdss PRINT command” on page 174.

---

### 3.7 Broken data sets

By broken data sets, we are designating all the problems which may affect the processing and the existence of your VSAM data sets. To simplify our search for a solution to each problem, we can use the three “Whats” — regarding occurrence, recovery, and avoidance.

Broken data sets can be caused by many different components and user errors. When diagnosing these types of problems, the first thing that must be done is to identify what is actually wrong with the data set. The first sign of a problem is the VSAM or the OS/390 system messages. A single error often generates numerous messages. You should focus your attention on the return code presented and the companion explanation. This return code will be the one passed by the component that first found the error. In most cases, you will need additional documentation; refer to 3.6, “Useful documents” on page 137.

In this section we group the errors by categories. However, this is not an easy task. Some of the categories overlap and even interact with others. For example, a bad channel program may be caused by an improper sharing, which caused a structural damage.

In each subsection, we will cover the three “Whats”:

- What happened to cause this problem?
- What must you do in order to recover your data now?
- What must you do to avoid this problem in the future?

#### 3.7.1 Lack of virtual storage

Following are messages that may 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 program's address space for a work area for Close
- 132 (Open): One of the following errors occurred:
  - Not enough storage was 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 program's address space for work areas, control blocks, or buffers.
- 40 (I/O): 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

#### **3.7.1.1 What happened?**

Due to the lack of virtual storage, an abend occurs. In this case, a symptom dump may be included.

#### **3.7.1.2 What to do for recovery?**

Because the data set processing was interrupted (abended) in apparently unknown circumstances, there are two cases:

- VSAM data set is being accessed by a subsystem as CICS, then CICS was doing the Synchpoint, journaling, and is able to recovery your data by rolling it back.
- VSAM data set being accessed by your program; then you should correct the virtual storage problem and re-run the program (if possible) or restore the backup and re-run the job.

Sometimes, the message is not the result of an abend. It can be an alert as with IEC161I, where BLDVRP macro indicates that there was not enough virtual storage to satisfy the request done by System Management Buffer (SMB). SMB gets the available storage, and processing goes on.

#### **3.7.1.3 What to do to avoid future problems?**

Initially, read 2.6.8, "Region size" on page 55. 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. Use the SmartBatch function Data Accelerator; refer to 2.8.1, "SmartBatch highlights" on page 121.

Determine if the VSAM buffers and their control blocks are below or above the 16 MB line. If below, read 2.6.9.6, "Locating VSAM buffers above 16 MB" on page 76 to learn how to move them above with integrity.

### **3.7.2 Initial loading problems**

Following are messages that may 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 {OPENICLOSEII/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): You attempted to store a record out of ascending key sequence in skip-sequential mode; record had a duplicate key; for skip-sequential processing, your GET, PUT, and POINT requests are not referencing records in ascending sequence; or, 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 (that is, when records are being stored in the data set the first time it is opened), GET, POINT, ERASE, direct PUT, and skip-sequential PUT with OPTCD=UPD are not allowed. During initial data set loading, for initial loading of a relative record data set, the request was other than a PUT insert.

### 3.7.2.1 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 a program of yours. Refer to 2.6.7, “Initial load option” on page 54, for more information.

When 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 above 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 for AIX is possible to have duplicated keys, but in this case you should 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 need to be sorted by a key field. The difference is that in mass insertion, we are loading sequentially logical records to a data set which already has previous data.

### 3.7.2.2 What to do for recovery?

Here, there is not a need for recovery. Your data is saved in the input file. It is a matter of sorting and re-running the program.

### 3.7.2.3 What to do to avoid future problems?

After correcting the error, introduce a procedure in production to avoid having the same error again.

### 3.7.3 Mismatch between catalog and data set

Following are the error messages which may 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 indicates that either the data or the index has been 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

#### 3.7.3.1 What happened?

The data set may be intact, but the catalog information describing the data set mismatch problems. It sometimes results in an open failure.

The most common discrepancies between the catalog and cluster are these:

- There were different time stamps between index and data components.

- HURBA and HARBA not correctly updated, mainly caused by an abend, without a normal close.
- Open-for-output bit on for a closed cluster; mainly caused by an abend, without a normal close, or other task accessing from other system. Refer to 3.7.10.2, “Abending task scenario” on page 157, for more information.
- RBAs fields in the VVR do not match the data set attributes, for example:
  - If the RBA of the high-level index CI is corrupted, you are not be able to perform direct requests against the data set.
  - If the RBA of the sequence set index CI is corrupted, you are not able to perform sequential access.

These last two discrepancies are not covered here; refer to 3.7.6, “Structural damage” on page 147.

A LISTCAT output (or CSI report) can help with the documentation for problem determination.

### 3.7.3.2 What to do for recovery?

IDCAMS VERIFY is a requirement in this type of problem. In this case, VERIFY can correct HURBA; verify the open-for-output bit, compare data and index time stamps.

- 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 will keep processing, but another problem is likely to occur.

We suggest you run VERIFY (to be sure and document the mismatch) and then run EXAMINE to guarantee 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 at this point, it must be restored before further use. Refer to 3.7.10.1, “Broken Index scenario” on page 155, to get some information on doing that. Note that EXAMINE may provide messages containing only informational data that may not require restoring the cluster.

- HURBA and HARBA not correctly updated.

If the HURBA is not updated, when the data set is subsequently 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, and a write operation might write records over previously written records. To avoid this, you can use the VERIFY command which corrects the catalog information.

- Open-for-output bit on for a closed cluster.

At next OPEN, VSAM implicitly issues a VERIFY command, when it 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 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.

### 3.7.4 Hardware errors

Following are some of the messages which indicate there may be hardware errors:

IOS000I dev,chn,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:

- 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 the compression management services (CMS) during compression processing)
- 246 (I/O): A severe error was detected by the compression management services (CMS) during decompression processing.
- 250 (I/O): A valid dictionary token does not exist for the compressed data set. The data record cannot be decompressed.



#### **3.7.4.1 What happened?**

Hardware I/O errors usually mean that the I/O hardware (channel, controller, device) had a problem executing that I/O. For compressed VSAM data sets, you may have hardware problems with the CPU compression assist function.

The first thing to do is break down the message to get more details on 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, such as: the physical record size, the device type and the CCHH of all extents. LOGREC output is also valuable. A GTF CCW trace may be necessary to run DIAGNOSE on the problem, however, for such tools, you may need to recreate the situation.

VERIFY IGGCSIVS is a program from SYS1.SAMPLIB accessing Catalog Search Interface (CSI) which produces a list of data set names defined in a given catalog that reside on a specific volume. Such a list might be helpful in a recovery situation affecting that volume.

When accessing with VSAM macros, a VSAM data set where a physical error was detected, the register 15 comes with return code equal to 12.

#### **3.7.4.2 What to do for recovery?**

In the case of 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 do not allow the use of the ICKDSF commands that perform installation, media maintenance and problem determination functions, such as 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.

#### **3.7.4.3 What to do to avoid future problems?**

If the error is in the DASD controller, keep a log of such type of occurrences to force a better quality of the manufacturer or change to other. Another solution (for some media problems) is to implement RAID-1 dual copy (in the same controller) or remote copy (in two controllers), mainly for your most critical data, such as logs.

### **3.7.5 Bad data or bad channel program**

A bad data or channel program may be indicated by the following message:

IDC3351I \*\* VSAM {OPENICLOSEII/O} RETURN CODE IS return-code

- 140 (Open): The catalog indicates this data set has an incorrect physical record size.

- 16 (I/O): Record not found.
- 88 (EOV): A previous extend error has occurred during EOV processing of the data set.

#### 3.7.5.1 What happened?

This problem may 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 may cause data sets to be overlaid with data from other data sets.

VSAM volume record (VVR) is a logical record within VSAM volume data set (VVDS). VVDS is a data set that describes the dynamic characteristics of VSAM and system-managed data sets residing on a given DASD volume. Together with the BCS, it is a part of an integrated catalog facility.

There are quite a few things that can cause the channel program to be bad. The most common causes of a bad channel program is that the data that describes the data set is bad. If a bad VVR is picked up at Open time, VSAM may try to access cylinder and tracks that do not belong to the data set getting various I/O errors.

If another program overlays the VSAM data set, this can cause the channel program to fail at that 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 program incorrectly, or in some cases they may be built correctly, but they are getting modified incorrectly and redriven by ERP or even third party products. Luckily, these reasons are not too common, even with new device types.

In regard to corrupted indexes, refer to 3.7.6, “Structural damage” on page 147.

With problems like these, it is important to get as much information about the data set as you can before the customer restores it. We suggest using:

- LISTCAT or CSI.

- DFSMSdss (PRINT command) or Ditto to print the 3390/3380 logical cylinder/track/head that the I/O error is occurring. It 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 has been recovered, the only “history” that can be EXAMINE’d is the SMF records. If the problem “clears up” after the data set is closed, but without the data set being recovered, you might suspect a problem with an internal control block being overlaid, rather than something on DASD.

#### **3.7.5.2 What to do for recovery?**

This overlay is a hard failure and the data set has to be manually restored from a backup. Often, in this case, the bad data itself gives a clue as to what data set or application has caused the overlay. Trying to avoid the restore for a bad KSDS data component, you may 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 (since 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 so this should be less of a problem.

#### **3.7.5.3 What to do to avoid future problems?**

Experience has shown that the majority of such errors are caused by improper sharing. If you are sharing your VSAM data set, refer to 3.7.7, “Improper sharing” on page 150.

Because such types of errors may also be caused by system errors, you may want to investigate the possibility of APARs and PTFs related to the problem.

### **3.7.6 Structural damage**

Following are messages that may 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 will have to be restored.

- 76 (Open): Attention message: The interrupt recognition flag (IRF) was detected for a data set opened for input processing. This 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 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): Incorrect pointer (no associated base record) in an alternate index.
- 156 (I/O): 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 (logical I/O error)

#### **3.7.6.1 What happened to cause this problem?**

KSDS or VRRDS VSAM data set organizations can “break” in more ways than other data sets 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, much structural information about such data sets is located in the ICF catalog. For example, two RBAs fields in the VVR are very important in accessing a KSDS data set, for example:

- If the RBA of the high-level index CI is corrupted, you are not be able to perform direct requests against the data set.
- If the RBA of the sequence set index CI is corrupted, you are not able to perform sequential access.

Refer to 3.9, “IDCAMS LISTCAT output fields” on page 166, for more information about RBA data in the VVDS catalog.

Then, if these fields are corrupted, errors may prevent you from accessing the data even though the data is intact. Also, it is possible that the index CI's horizontal chain is destroyed, inhibiting the access to the data. One common way these fields can get corrupted is due to 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 and for that reason do not dismiss any possibilities just because the job was not updating the file.

SMF records are very helpful when diagnosing VVR damage. By investigating the SMF records (for example, type 60, 62 and 64) from all systems, improper access of the data set can be identified as well as the time frame of the corruption.

#### **3.7.6.2 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, since 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. Remember that 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 losing the index in a KSDS/VRRDS, one possible recovery path is to read the data (in physical sequential mode) via its data component. Here you may use an assembler program (MACRF=ADR in ACB and OPTCD=ADR in RPL), or by IDCAMS Repro. Then, classify by the key and use an IDCAMS Define and REPRO to recreate the KSDS. Refer to 3.7.10.1, “Broken Index scenario” on page 155 for more details on that.

#### **3.7.6.3 What to do to avoid future problems?**

Experience has shown that some of such errors are caused by improper sharing. If you are sharing your VSAM data set, refer to 3.7.7, “Improper sharing” on page 150.

Because such types of errors may be caused by system errors, you may want to investigate the possibility of APARs and PTFs related to the problem.

### 3.7.7 Improper sharing

Following are messages that may indicate improper sharing:

IDC3351I \*\* VSAM {OPEN|CLOSE|I/O} RETURN CODE IS return-code

- 88 (Open): A previous extend error has occurred during EOVS 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, then data may be lost if processing continues.
  - The data set was not properly closed.
  - The data set is opened for output on another image.
- 16 (I/O): Record not found.
- 20 (I/O): Record already held in exclusive control by another requester.
- 28 (I/O): Data set cannot be extended because VSAM cannot allocate additional DASD space.
- 236 (I/O): Validity check error for SHAREOPTIONS 3 or 4.

IDC11705I INDEX RECORD CONTAINS DUPLICATE INDEX POINTERS  
pointer-value

#### 3.7.7.1 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 make sure the share options are proper. Refer to 1.11.3.3, “Share options” on page 36 to get information on share options. Here are some causes of sharing problems:

- Sharing a data set across regions (cross-region) or across systems (cross-system) without using proper enqueueing procedures to protect data set integrity. For shareoptions shr(3 3) shr(4 3) shr(3 4) see related information in OY36328.
- Sharing a data set across systems — even using the appropriated share option — but without propagating 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.

ENQ on SYSVSAM is the mechanism by which VSAM ensures that only one ACB is open for output. If GRS is not passing this resource name to all systems in the complex (for example, they have SYSVSAM in the exclusion list; see the MVS/ESA Planning Global Resource Serialization Guide), write integrity can be lost.

- Sharing a data set across systems and running explicit VERIFY on one system while the other system still has the data set opened for output. Consider the following scenario:

A data set is used for CICS on-line applications during the day and batch processing on another system at night. It is assumed that they never have the data set opened at the same time. If there is any delay in closing the data set on one system before the other issues an explicit VERIFY on the data set (even before opening for input), index damage can result from the change in the high used RBA.

- 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).
- Perhaps there is not enough primary space left, and a secondary allocation request is attempted to increase the size of a data set while processing with SHROPT=4 and DISP=SHR. Then, VSAM end-of-volume processing acquires new extents for the VSAM data set, and updates the new extent information on the critical control block data in common storage so that this new space is accessible by all regions using this VSAM data set.

Now, if an abend or unexpected error occurs, which prevents this space allocation from being completed, all regions are prevented from further extending the data set. To obtain additional space, you must close the VSAM data set in all regions, then reopen it.

- Another area 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. Refer to 4.2.7, "Control Block Update Facility (CBUF)" on page 192, for more information.

Documentation is of paramount importance to address sharing problems. The necessary documents should be obtained at each system (or address space) accessing the troubled data set. The major document needed here 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 if the users have the data set open for output from different applications at the same time. A common user error in this area is applications or ISV products that were not intended to be run from multiple systems (and so they have no logic to serialize updates), but the customer using them in this manner.

#### **3.7.7.2 What to do for recovery?**

Following is a list of recovery options:

- 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 have two meanings: If VERIFY processing then successfully closes the data set, VERIFY processing issues condition code 0 at the end of its processing. List the catalog entry again. Any change in the data or index component HURBA indicates that the explicit VERIFY was needed and may resolve the damage to the data set.
- Execute the IDCAMS EXAMINE command on the data set. Completion of EXAMINE without error will prove that damage did not occur in a previous job. If the data set shows damage at this point, it must be restored before further use.
- Proceed with the application job execution.
- Execute IDCAMS EXAMINE on the data set when the job completes
- If damage to the cluster has occurred, run EXAMINE on SMF records from all systems which do have the ability to access the DASD volume. If shared access to the data set has occurred, correct or eliminate the contention for the data set

#### **3.7.7.3 What to do to avoid future problems?**

You should issue the VERIFY command every time you open a VSAM cluster that is shared across systems or address spaces. Read carefully 4.2, “Sharing VSAM data sets” on page 183 and use all the serializing techniques to avoid the structural damage and the data integrity of your data set.

### **3.7.8 Mismatch between catalog and VTOC**

In this book we do not cover much catalog recovery; however, some of the catalog problems are mentioned as the ones associated with the IDC3009I message. Refer to 3.8, “IDC3009I message” on page 159, where a more detailed description of the return and reason codes from this message are



presented. Refer also to 3.7.11, “Recovering ICF catalogs” on page 158, where you will find more information about the subject.

IDC3351I \*\* VSAM {OPENICLOSEII/O} RETURN CODE IS return-code

- 104 (Open): 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 indicates that extent information in the catalog record may not agree with the extents indicated in the volume's VTOC.
- 132 (Open): One of the following errors occurred.
  - Not enough storage was available for work areas.
  - The format-1 DSCB or the catalog cluster record is incorrect.
- 200 (Open): Volume is unusable.
- 240 (Open): Format-4 DSCB and catalog time stamp verification failed during volume mount processing for output processing.

IDC3009I VSAM CATALOG RETURN CODE IS return-code — REASON CODE IS IGGOCLaa — reason-code.

### 3.7.9 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 and can often be difficult to analyze because the component affected might not be the one that caused the problem. Review previous messages, abends, console logs, or other system responses. They could indicate the source of the failure.

#### 3.7.9.1 Accumulate as much information as possible.

It can help you isolate or resolve your problem, and the IBM Support Center will request it if trap or trace information is needed:

- When was the problem first noticed?
- How was the problem identified (good output versus bad output)?
- Were any system changes or maintenance recently applied? For example, a new device, software product, APAR, or PTF?

- Does the problem occur with a specific data set, device, time of day, and so forth?
- Does the problem occur in batch or TSO mode?
- Is the problem solid or intermittent?
- Can the problem be re-created?
- EXAMINE the system and console logs for failure-related abends, messages, or return codes. A damaged VSAM data set can also cause incorrect output.
- Add any failure-related return codes to the keyword string, exactly as the system presents them. You can also add the abend or message type-of-failure keywords to the incorrect output keyword string to define the symptoms more closely:
- 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 returned to the caller of record management. Reason codes 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/MVS Access Method Services for ICF and the chapter on checking a VSAM key-sequenced data set cluster for structural errors in DFSMS/MVS Using Data Sets. The EXAMINE command provides details about the nature of data set damage. If these service aids indicate that the data set is not damaged, inform the IBM Support Center if you call for assistance. If they indicate that the data set is damaged, keep a copy of the output for possible use by the IBM Support Center. Be prepared to describe the type of data set damage. You should attempt to recover the data set and rerun the failing job to determine whether the problem is resolved.

### 3.7.10 Recovery scenarios

Following are some real life scenarios covering the rise and fall of a VSAM data set.

### 3.7.10.1 Broken Index scenario

In this scenario, you observe the following conditions:

1. You have a program which opens a KSDS data set for update. The access argument is through RBA. However, by mistake, someone has prepared JCL pointing to the index name instead of cluster name or data name.
2. 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.

```
//*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'
```

3. The next time that you open the cluster or the data, the following happens:
  - At open, the message IEC161I 058(018)-061 is issued and the OPEN return code is 4. The processing continues.

### 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 VRs.

- Going on with the 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 indicating the error. When the program is IDCAMS, the processing stops and the messages below are issued in the SYSPRINT ddname file. If the data set is open for output, the timestamp is corrected, but the I/O error remains, once 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.
```

4. You run an EXAMINE IDCAMS command, getting back the messages shown. In our test, we caused the error by filling the first control interval with X'00':

```

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
00 *.....*
000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000
...

ERROR LOCATED AT OFFSET 00000010
MAJOR ERRORS FOUND BY INDEXTEST
LASTCC=8

```

##### 5. For recovering the index:

- Use the REPRO Command to copy just the data component of the KSDS. Specify the data component name (not the Cluster Name) in the REPRO INFILE parameter.
- DELETE and reDEFINE the damaged Cluster.
- REPRO from the Sorted Sequential File to the newly defined Cluster. Record Management will rebuild the Index Component.

Note that this method does not work for a VSAM Catalog, Integrated Catalog (ICF), or for a Spanned KSDS.

#### 3.7.10.2 Abending task scenario

At taskabend, RTM does not properly close the VSAM data set, then:

- Buffers are not flushed (with the exception of cross systems shareoptions 4) and the HURBA is not updated in the catalog. In Language Environment there is an option (TRAP ON) which forces the close (with flush and HURBA actualization), using a STAE exit.

If the HURBA was not updated, when the data set is subsequently 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, and a write operation might write records over previously written records. To avoid this, you can use the VERIFY command, which corrects the catalog information. For additional information about recovering a data set, see DFSMS/MVS Managing Catalogs, SC26-4914-04.

- 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 and issues an informational message stating whether the VERIFY command is successful.

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. Refer to 3.7.3, “Mismatch between catalog and data set” on page 142, for more information.

### **3.7.11 Recovering ICF catalogs**

A sequence of tasks to recovery an ICF damaged catalog follows.

#### **3.7.11.1 Recovering damaged BCS entries**

- Remove the sphere or base record, if it exists.

The damage 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.

- Remove any remaining association records.

You can reexecute 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.

- Reintroduce the removed entries into the catalog.

After the damaged entries have been removed, you can redefine the data sets. For VSAM and SMS-managed non-VSAM data sets, you should 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 proper order after the generation data group has been redefined. You can use the LISTCAT command to determine the current generation data sets.

### **3.7.12 Recovering damaged VVDS entries**

- 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. See 3.7.11.1, “Recovering damaged BCS entries” on page 158 for more details on removing BCS entries.

- Remove the damaged VVDS records.

After you have removed the BCS entries, you can remove the VVDS records by using the Delete command and specifying VVR or NVR. Delete VVR or NVR also removes the Format 1 DSCB from the VTOC.

- 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. The data that you do recover might be damaged or out of sequence.

---

### 3.8 IDC3009I message

When working with catalogs and VSAM data sets, the IDC3009I message is probably the most common message you receive. IDC3009I is issued as a result of a *catalog* error or exceptional condition involving a catalog.

Following is the format of this message:

```
IDC3009I VSAM CATALOG RETURN CODE IS return-code — REASON  
CODE IS IGGOCLaa — reason-code
```

The message specifies a return code and a reason code, and they are described in *OS/390 MVS System Messages, Vol 3 (GDE-IEB)*, GC28-1786. The return codes are as listed in Table 15. The table is not intended to replace the messages manual, but rather to serve as a quick reference.

Table 15. IDC3009I message

Return Code	A brief explanation:
4	Error while performing open/close to a VSAM catalog. See reason code, can be either a small region size.
8	The specified entry does not exist if locate was the action <i>or</i> the entry already exists, if action is one which adds an entry to a catalog
10	An incorrect record type was found in the catalog
12	The component was not found. It can be an AIX, a data or an index, depending on reason code
14	Catalog cell not found.: Run the Access Method Services DIAGNOSE command to check for additional information.
16	Request is not supported for SMS managed volumes
18	ALTER for a backup <i>or</i> migrated data set failed.
20	VSAM catalog has run out-of-space
22	Catalog field vector table (FVT) is zero or an incorrect FVT field was found.
24	Permanent read error in VSAM catalog.
26	ICF catalog: VSAM record management error (catalog record too big or too small)
28	Permanent I/O error in VSAM catalog. Messages IEC331I, IEC332I, and IEC333I have been printed to aid in determining the cause of the error and where the error occurred.
30	Automated Tape Library DataServer (ATLDS) processing error
32	Error in the VSAM catalog parameter list and indicates an internal error in Access Method Services
36	Data set not found or DSCB indicates a VSAM data set
38	Error found in a catalog installation error (user or DFHSM supplied) while processing a CATALOG, INDEX or LOCATE macro. If DFHSM, messages (ARCxxxI) are issued before.
40	Two 2 or more tasks are modifying a catalog entry, causing it to be extended in size, and one task finds that it was unable to specify sufficient virtual storage for catalog management's new requirements.
42	A DADSM error occurred on branch entry to DADSM back end (DADSM rename, locate or scratch, according to reason code).
44	The caller's work area is too small
48	Incorrect VSAM catalog function



Return Code	A brief explanation:
50	An error has been detected in VVDS manager error
52	Permanent I/O error on user volume. An attempt to run a direct execute channel program (EXCP) write of a DSCB to the VTOC failed. The reason codes are the event control block (ECB) completion codes returned after the EXCP write and you can the meaning in "Event Control Block Fields", <i>DFSMS/MVS DFSMSdfp Advanced Services</i> , SC26-4921
54	Incorrect use of JOBCAT or STEPCAT with SMS data sets
56	A security verification failed
58	Error while reading a DSCB into a work area. Reason code is DADSM OBTAIN Return Code and you can find in "Return Codes from OBTAIN", <i>DFSMS/MVS DFSMSdfp Advanced Services</i> , SC26-4921.
60	Incorrect entry type for requested action
62	Error while initializing the extension of a data set. The reason codes are the complement of the error codes returned from the DADSM extend routine; refer to <i>DFSMSdfp Diagnosis Reference</i> , LY27-9606
64	VSAM catalog cannot find either a data or an index entry which is associated with a cluster or alternate index entry.
66	Bad DADSM parameter list. The reason code is the DADSM return code, and you can find in <i>DFSMSdfp Diagnosis Reference</i> , LY27-9606.
68	No space is available on the user volume for the ICF catalog. Only the primary volume will be used.
70	A generation data set component was not found in the GDG sphere record.
72	The user volume is not mounted. The reason codes are from VSAM open/close/end-of-volume, volume mount and verify routine IDA0192V.
74	catalog Cell not found. Different reasons codes from those specified in return code 14.
76	No unit available for mounting or volume not mounted
78	Subrecord move error. It can be: Catalog management was unable to obtain enough virtual storage to contain the catalog record with the addition of the subrecord <i>or</i> verification record was not within the length range of 1 to 256.
80	The object specified in the RELATE parameter of a DEFINE command does not exist, or is improper for the type of object being defined.
82	The number of data set entries passed exceeds the allowed maximum for the catalog name locate.

Return Code	A brief explanation:
84	Date error: a DELETE to a data set with an unexpired purge date and PURGE was not specified or there are conflicting date format (rc=2).
86	Recatalog error, reason vary according to reason codes
88	Error with a catalog recovery area (CRA) define operation: The total space specified was not able to contain the size specified for the catalog and the one cylinder of space required for the CRA.
90	Delete error
92	The maximum number of extents was reached
94	A DADSM OBTAIN request failed during a VSAM catalog delete request. The reason code is the OBTAIN return code and you can find out its meaning in "DADSM OBTAIN Function Return Codes", <i>DFSMSdfp Diagnosis Reference</i> , LY27-9606.
96	An error occurred in specifying key length, key position or record size for an alternate index or spanned cluster.
98	An unusual condition occurred during an ALTER name of a unique or non-VSAM data set. The reason code is <i>status byte</i> returned by the DADSM RENAME function. See the meaning in <i>DFSMSdfp Diagnosis Reference</i> , LY27-9606, "Status Codes from RENAME"
102	A DADSM SCRATCH request failed during a VSAM catalog delete request. for a unique or non-VSAM data set. The reason code is <i>status byte</i> returned by the DADSM SCRATCH function. See the meaning in <i>DFSMSdfp Diagnosis Reference</i> , LY27-9606, "Status Codes from SCRATCH"
104	A DEFINE command is attempting to define a second VSAM master catalog when a VSAM master catalog already exists and is open.
106	A format-4 DSCB processing error was encountered
108	An incorrect field name was found in the field parameter list. The field name passed by AMS does not exist in the VSAM catalog management dictionary. The message indicates that the caller's AMS release level or maintenance level is different from the CATALOG level that is being called.
110	Unable to modify or delete RACF profile. It does not exist (but has RACF indicator (reason 4) or a rename processing for a RACF-protected data set failed because as a result of the new name, the data set cannot be defined to the security subsystem.
112	Incorrect Catalog field parameter list(FPL).
114	As a result of an IMPORT, IMPORTRA, or DEFINE command, VSAM has attempted to establish a RACF profile for a cluster/alternate index, data, or index object (reason codes 0, 4, and 8 respectively). This failed because a profile with the same name already exists.

Return Code	A brief explanation:
116	VSAM catalog records are incorrect. The reason code explain for what kind of object the record can not be obtained.
118	The data set name is ineligible for RACF definition. User does not have authority (reason code 0) or RACF inactive (reason code 12)
120	Attempt to modify the non-existent or system field. This is a system error.
124	Incorrect control interval number
126	Alter new name of a GDS, non-VSAM or cluster failed because an ACS service returns a non-zero return code. ACS reason code = catalog reason code + 1000. services reason codes from 1000 to 1255. The ACS services return and reason codes are documented in the <i>DFSMSdfp Diagnosis Reference</i> , LY27-9606.
128	A user-provided storage is outside the user region. Probable system error.
130	An ALTER RENAME recatalog error. Reason code explains why.
132	Incorrect pointer value in argument list. Probable system error.
136	Required parameters not supplied. Probable system error.
138	DADSM RENAME error. Reason code is the volume status code returned from DADSM.
142	DADSM OBTAIN error. Reason code is the return code from OBTAIN and you find in <i>DFSMSdfp Advanced Services</i> , SC26-4921.
144	An incorrect entry name format or the name has an initial character as a numeric or used a restricted name. See reason code.
148	Volume already owned by another VSAM catalog. Specify a different volume or use the Access Method Services ALTER REMOVEVOLUMES command to reset the volume ownership if a catalog should not own the volume.
150	Name length error for an SMS construct. Storage class (reason code 2) or data class (4) or management class (6)
152	A non-empty catalog cannot be deleted. If the catalog and all of its entries are to be deleted specify the FORCE parameter on the Access Method Services DELETE CATALOG command.
156	The volume does not contain a data space for another VSAM data set. There is insufficient space in the data spaces allocated on the volume to satisfy a request for suballocation.
160	DELETE space is requested for a volume containing a catalog. See explanation and programmer response in reason code.

<b>Return Code</b>	<b>A brief explanation:</b>
164	There is insufficient virtual storage available for VSAM catalog management. Increase the region size available to the step.
168	Unsupported device type.
172	A DEFINE command specifies the name of a data set, with the UNIQUE attribute, but there is already a data set on the specified volume with that name.
176	There is no space in the VTOC for a DSCB. Delete any unneeded data sets or data spaces from the volume or recreate the volume with a larger VTOC.
178	An error occurred during ICF catalog processing of a VSAM partial release request.
180	Data space name not found. Probable system error. The catalog or a volume may have been totally or partially destroyed.
182	Bad DADSM UPDATE parameter list. The reason code is the Volume status code from DADSM. You can see a volume status using ISMF.
184	The data set is currently open and cannot be deleted or altered.
186	Error attempting to lock a catalog or access a locked catalog.
188	Catalog unavailable. See return code in the messages manual.
190	Authorization error on a facility class function applied to SMS data sets. The user need read access authority to 'STGADMIN.IGG.LIBRARY'.
192	Maximum logical record length specified is greater than 32,761 for a non-spanned data set.
194	An error occurred during multi-level alias (MLA) facility processing. See reason code in the message manual.
196	The data component control interval size specified is greater than 32,767.
198	An attempt has been made to use an unsupported feature. Related to a UCB's device defined above 16M, which resides a VSAM catalog.
200	The specified or defaulted control interval size of the index component is greater than the maximum block size of the index device. Reduce the control interval or use a device with a larger maximum block size.
202	Storage management subsystem call error. Redefine the data set with a management class with no retention limit or with a specified retention value equal to or exceeding the date specified in the ALTER command.
204	Key specification extends beyond end of maximum logical record. Reduce the key length, change the key position, or increase the logical record length.
208	The buffer space specified is too small. Do not specify BUFFERSPACE, let the default value.

<b>Return Code</b>	<b>A brief explanation:</b>
210	Subsystem call error. Reason code is the return code from Subsystem call.
212	Control interval size calculation unsolvable. See reason code.
214	Subsystem call error. Reason code is the return code from Subsystem call
216	The volume's VTOC is not interpretable. An incorrect VTOC was deleted during update extend processing for a VSAM data set. Restore the volume in order to correct the VTOC.
220	A DOS VTOC cannot be converted to an OS VTOC. Restore the volume in order to correct the VTOC.
222	Alter new name of a GDS, non-VSAM or cluster failed because an ACS service returns a non-zero return code. ACS reason code = catalog reason code + decimal 256. ACS services return and reason codes are documented in <i>DFSMSdfp Diagnosis Reference</i> , LY27-9606.
224	A field in a catalog entry has exceeded the maximum allowable number of repetitions. For example, trying to add more than 255 volumes or more than 125 alternate indexes in the upgrade set.
226	The caller is not authorized to perform the requested function. The caller must be running in key 0 - 7, must be in supervisor state, or must be APF authorized.
228	An error occurred while processing an Enhanced Catalog Sharing (ECS) request. Refer to reason code in the message manual to correct the problem
230	VSAM catalog retrieve of a control interval failed to get a low range record from the VSAM catalog. Probable system error.
232	An error was encountered while VSAM Catalog Management was performing SMF processing. Use the reason code as a return code to IDC3009I to find out the reason of the error.
234	End of data encountered while reading the low data key range of the VSAM catalog. Probable system error.
236	An error was encountered in space-map. This condition arises when the catalog's volume entry is incorrect. Reconstruct the volume entry record. If that is not possible, restore the catalog.
238	No user catalog entry in the master catalog for Convert Volume processing.
240	Required DD statement not supplied. See reason code in the message manual
242	A physical I/O error occurred trying to erase the data set being deleted. The reason code correspond to the VSAM Record Management error codes. See "Record Management Return and Reason Codes" in <i>DFSMS/MVS Macro Instructions for Data Sets</i> , SC26-4913. Run the job again with the NOERASE option. The data set cannot be deleted.

Return Code	A brief explanation:
244	Erase action failed. The VSAM Catalog Management is unable to open the VSAM data set being deleted. The reason codes correspond to the VSAM OPEN error codes. See "OPEN Return Codes" in <i>DFSMS/MVS Macro Instructions for Data Sets</i> , SC26-4913. Alternatively, you run the DELETE command again with the NOERASE option.
246	CAS service task abended or detected an abnormal condition. See reason code in the messages manual.
248	A function requires a volume that is not owned by the VSAM catalog being used.
250	VSAM Record Management has found a <i>logical error</i> during erase processing while deleting a VSAM data set. The reason codes correspond to the VSAM record management logical error code. See "Record Management Return and Reason Codes" in <i>DFSMS/MVS Macro Instructions for Data Sets</i> , SC26-4913. Alternatively, you run the DELETE command again with the NOERASE option.
254	An error was encountered during catalog reorientation. The reason code indicates in what part the error was found: close (reason code 0), open (2), allocation (4) or an unexpected error (6).
<b>Note:</b> Reason codes, explanations, system actions and programmers response are described in <i>OS/390 MVS System Messages, Vol 3 (GDE-IEB)</i> , GC28-1786, under IDC3009I message.	

### 3.9 IDCAMS LISTCAT output fields

In the ICFcatalog, VSAM maintains special RBA, CI, and Key values together with time stamps, which are very important in accessing a VSAM cluster. The most important ones are located in the VVR AMDSB in the VVDS cell and are updated only at Close time.

IDCAMS LISTCAT command displays this information, which is key to diagnosing what caused the error. The screen below shows some JCL you need in order to issue the LISTCAT command via BATCH. You can issue the same command under TSO/ISPF.

```
//LISTCAT JOB 'LISTCCAT EXAMPLE',NOTIFY=&SYSUID
//*-----
//EXAMPLE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
        LISTC ENTRY (DAWN.KSDSEXG) ALL
/*
```

The output from the LISTCAT command is:

```
1IDCAMS  SYSTEM SERVICES                                TIME:
21:15:46          04/20/00      PAGE      1
0
        LISTC ENTRY (DAWN.KSDSEXG)  ALL
00044403
0CLUSTER  ----- DAWN.KSDSEXG
        IN-CAT --- MCAT.SANDBOX.R9.VSBOX11
        HISTORY
                DATASET-OWNER----- (NULL)      CREATION-----2000.108
                RELEASE-----2          EXPIRATION-----0000.000
        SMSDATA
                STORAGECLASS -----STRIPE      MANAGEMENTCLASS---MCDB22
                DATACLASS  -----KEYEDEXG      LBACKUP ---0000.000.0000
                BWO STATUS-----00000000        BWO TIMESTAMP---00000 00:00:00.0
                BWO----- (NULL)
        RLSDATA
                LOG ----- (NULL)      RECOVERY REQUIRED -- (NO)
                VSAM QUIESCED ----- (NO)      RLS IN USE ----- (NO)
0        LOGSTREAMID----- (NULL)
                RECOVERY TIMESTAMP LOCAL-----X'0000000000000000'
                RECOVERY TIMESTAMP GMT-----X'0000000000000000'
                PROTECTION-PSWD----- (NULL)      RACF----- (NO)
        ASSOCIATIONS
                DATA-----DAWN.KSDSEXG.DATA
                INDEX-----DAWN.KSDSEXG.INDEX
0 DATA  ----- DAWN.KSDSEXG.DATA
        IN-CAT --- MCAT.SANDBOX.R9.VSBOX11
        HISTORY
                DATASET-OWNER----- (NULL)      CREATION-----2000.108
                RELEASE-----2          EXPIRATION-----0000.000
                ACCOUNT-INFO----- (NULL)
                PROTECTION-PSWD----- (NULL)      RACF----- (NO)
        ASSOCIATIONS
                CLUSTER--DAWN.KSDSEXG
```

```

ATTRIBUTES
KEYLEN-----8      AVGLRECL-----300
BUFSIZE-----10240  CFSIZE-----4096
RKP-----0      MAXLRECL-----300
EXCPEXIT----- (NULL)  CI/CA-----192
STRIPE-COUNT-----4
SHROPTS (1,3)  RECOVERY  UNIQUE      NOERASE  INDEXED
NOWRITECHK    NOIMBED    NOREPLICAT
UNORDERED      NOREUSE    NONSPANNED  EXTENDED
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
VOLUME
1IDCAMS  SYSTEM SERVICES
21:15:46      04/20/00      PAGE      2
0      VOLSER-----SBOX31  PHYREC-SIZE-----4096
HI-A-RBA-----736100352  EXTENT-NUMBER-----1
DEVTYPE-----X'3010200F'  PHYRECS/TRK-----12
HI-U-RBA-----203685888  EXTENT-TYPE-----X'00'
VOLFLAG-----PRIME  TRACKS/CA-----4
STRIPE-NUMBER-----1
EXTENTS:
LOW-CCHH----X'000A000A'  LOW-RBA-----0
TRACKS-----3744
HIGH-CCHH---X'01040003'  HIGH-RBA-----736100351
VOLUME
VOLSER-----SBOX30  PHYREC-SIZE-----4096
HI-A-RBA-----736100352  EXTENT-NUMBER-----1
DEVTYPE-----X'3010200F'  PHYRECS/TRK-----12
HI-U-RBA-----0  EXTENT-TYPE-----X'40'
VOLFLAG-----PRIME  TRACKS/CA-----4
STRIPE-NUMBER-----2
EXTENTS:
LOW-CCHH---X'0104000B'  LOW-RBA-----0
TRACKS-----3744
HIGH-CCHH---X'01FE0004'  HIGH-RBA-----736100351

```

TIME:



```

VOLUME
VOLSER-----MHLV14      PHYREC-SIZE-----4096
HI-A-RBA-----736100352  EXTENT-NUMBER-----1
DEVTYPE-----X'3010200F'  PHYRECS/TRK-----12
HI-U-RBA-----0          EXTENT-TYPE-----X'40'
VOLFLAG-----PRIME        TRACKS/CA-----4
STRIPE-NUMBER-----3
EXTENTS:
LOW-CCHH----X'0103000B'    LOW-RBA-----0
TRACKS-----3744
HIGH-CCHH---X'01FD0004'    HIGH-RBA-----736100351
VOLUME
VOLSER-----SBOX28      PHYREC-SIZE-----4096
HI-A-RBA-----736100352  EXTENT-NUMBER-----1
DEVTYPE-----X'3010200F'  PHYRECS/TRK-----12
HI-U-RBA-----0          EXTENT-TYPE-----X'40'
VOLFLAG-----PRIME        TRACKS/CA-----4
STRIPE-NUMBER-----4
EXTENTS:
LOW-CCHH----X'0103000B'    LOW-RBA-----0
TRACKS-----3744
HIGH-CCHH---X'01FD0004'    HIGH-RBA-----736100351
0  INDEX ----- DAWN.KSDSEXG.INDEX
IN-CAT --- MCAT.SANDBOX.R9.VSBOX11
HISTORY
DATASET-OWNER----(NULL)    CREATION-----2000.108
RELEASE-----2            EXPIRATION-----0000.000
PROTECTION-PSWD----(NULL)  RACF----- (NO)
ASSOCIATIONS
CLUSTER--DAWN.KSDSEXG
ATTRIBUTES
KEYLEN-----8            AVGLRECL-----0
BUFSPACE-----0          CISIZE-----2048
RKP-----0              MAXLRECL-----2041
EXCPEXIT----- (NULL)    CI/CA-----21
SHROPTNS(1,3)  RECOVERY  UNIQUE          NOERASE    NOWRITECHK
NOIMBED        NOREPLICAT  UNORDERED
NOREUSE        EXTENDED
STATISTICS
REC-TOTAL-----263        SPLITS-CI-----42
EXCPS-----39045          INDEX:
REC-DELETED-----0        SPLITS-CA-----1
EXTENTS-----2            LEVELS-----3
REC-INSERTED-----0        FREESPACE-%CI-----0
SYSTEM-TIMESTAMP:          ENTRIES/SECT-----13
REC-UPDATED-----21884     FREESPACE-%CA-----0
X'B3ECB2FDD72E7785'      SEQ-SET-RBA-----0

```

```

REC-RETRIEVED-----0    FREESPC-----63488
HI-LEVEL-RBA-----436224
1IDCAMS  SYSTEM SERVICES
21:15:46      04/20/00    PAGE      3
0    ALLOCATION
      SPACE-TYPE-----TRACK    HI-A-RBA-----602112
      SPACE-PRI-----12    HI-U-RBA-----538624
      SPACE-SEC-----2
      VOLUME
      VOLSER-----SBOX31    PHYREC-SIZE-----2048
HI-A-RBA-----602112    EXTENT-NUMBER-----2
      DEVTYPE-----X'3010200F'    PHYRECS/TRK-----21
HI-U-RBA-----538624    EXTENT-TYPE-----X'00'
      VOLFLAG-----PRIME    TRACKS/CA-----1
      EXTENTS:
      LOW-CCHH----X'00000002'    LOW-RBA-----0
TRACKS-----12
      HIGH-CCHH----X'0000000D'    HIGH-RBA-----516095
      LOW-CCHH----X'01040004'    LOW-RBA-----516096
TRACKS-----2
      HIGH-CCHH----X'01040005'    HIGH-RBA-----602111
      VOLUME
      VOLSER-----SBOX30    PHYREC-SIZE-----2048
HI-A-RBA-----1032192    EXTENT-NUMBER-----1
      DEVTYPE-----X'3010200F'    PHYRECS/TRK-----21
HI-U-RBA-----0    EXTENT-TYPE-----X'40'
      VOLFLAG-----CAND-SPACE    TRACKS/CA-----1
      EXTENTS:
      LOW-CCHH----X'01FE0005'    LOW-RBA-----516096
TRACKS-----12
      HIGH-CCHH----X'01FF0001'    HIGH-RBA-----1032191
      VOLUME
      VOLSER-----MHLV14    PHYREC-SIZE-----2048
HI-A-RBA-----1548288    EXTENT-NUMBER-----1
      DEVTYPE-----X'3010200F'    PHYRECS/TRK-----21
HI-U-RBA-----0    EXTENT-TYPE-----X'40'
      VOLFLAG-----CAND-SPACE    TRACKS/CA-----1
      EXTENTS:
      LOW-CCHH----X'01FD0005'    LOW-RBA-----1032192
TRACKS-----12
      HIGH-CCHH----X'01FE0001'    HIGH-RBA-----1548287
      VOLUME
      VOLSER-----SBOX28    PHYREC-SIZE-----2048
HI-A-RBA-----2064384    EXTENT-NUMBER-----1
      DEVTYPE-----X'3010200F'    PHYRECS/TRK-----21
HI-U-RBA-----0    EXTENT-TYPE-----X'40'
      VOLFLAG-----CAND-SPACE    TRACKS/CA-----1

```

TIME:

```

EXTENTS:
LOW-CCHH-----X'01FD0005'      LOW-RBA-----1548288
TRACKS-----12
HIGH-CCHH-----X'01FE0001'      HIGH-RBA-----2064383
1IDCAMS  SYSTEM SERVICES
21:15:46      04/20/00      PAGE      4
0          THE NUMBER OF ENTRIES PROCESSED WAS:
          AIX -----0
          ALIAS -----0
          CLUSTER -----1
          DATA -----1
          GDG -----0
          INDEX -----1
          NONVSAM -----0
          PAGESPACE -----0
          PATH -----0
          SPACE -----0
          USERCATALOG -----0
          TAPELIBRARY -----0
          TAPEVOLUME -----0
          TOTAL -----3
0          THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
0IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
0
0IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

```

Use the output from the LISTCAT command listed above to follow the explanations about the most key catalog fields shown by LISTCAT.

### 3.9.1 High used RBA value (HURBA) for KSDS

There are two HURBAs per KSDS cluster: the data HURBA, and the index HURBA:

- Data HURBA

This is the RBA of the physically highest record in data CI (it does not imply the highest key, because the existence of splits). In other words, it points to the end of the last CA which ever, at any time, contained data. It is incremented by one CA, if a CA split or add-to-the-end in a new CA occurs. It is always on a CA boundary.

If all logical records are deleted the HURBA does not turn to zero — only via create or re-setting. Also, if the data set was defined with the Reuse option, in this case at Open time, HURBA is re-set to zero.

When share option cross-system 4 is set to the data set, this value cannot be modified.

If a data set has HURBA=0, it cannot be opened for input.

- Index HURBA

This points to the end of the last index CI written. It is incremented by one CI if a new record is added to the index. Separate HURBA values are maintained for the imbedded sequence set, and the high level index if IMBED is used. Index HURBA is always on a CI boundary.

There are also two types of data components: ESDS and RRDS:

- ESDS data component

The HURBA points to the end of the last CI which contains data. It is incremented by one CI, if a new CI is entered via add-to-end processing.

It is always on a CI boundary.

- RRDS data component

The HURBA points to the end of the last CA which ever contained data. It is incremented by add-to-end processing which enters a new CA, or by direct insert of a record whose slot number resolves to a new CA.

It is always on a CA boundary.

The name, HURBA, does not mean that every byte up to the HURBA is used. There may be imbedded free space due to distributed space, record deletion, slots which have never been used (RRDS) as well as unused space at the end of the CI or CA. The HARBA is an RBA pointer to the end of the last current extent of that cluster component. Until the HURBA equals the HARBA for that component, another extent will not be taken.

### **3.9.2 High allocated RBA value (HARBA)**

This is the highest RBA available within allocated space to store the data component, its key range, the index component, or the sequence set records of a key range. KSDS has two HARBAs: one for the index; another for data.

The difference (HARBA - HURBA) is the amount of space ready to be freed by the free space release option after close. It includes all the CAs in the physical end of the data sets with only free CIs.

### **3.9.3 FREESPC**

This is the actual number of bytes of free space in the total amount of space allocated to the data or index component. Free space in partially used control intervals is not included in this statistic.

### **3.9.4 High key RBA/CI**

It is the RBA of the logically highest data CI (the one with the record with the highest key). When the cross-system SHAREOPTION 4 is set to the data set, this value cannot be modified.

### **3.9.5 High-level index RBA value**

Any time the data set is accessed directly through a key, VSAM uses this value to find the highest level index record to start the search through lower levels.

If the high-level index RBA is corrupted, the user will not be able to perform direct requests against the data set.

### **3.9.6 Sequence set first RBA value**

If the data set is being read sequentially, from start to finish (for example, REPRO), VSAM uses this value to go directly to the first sequence set record. If Sequence Set RBA is corrupted sequential access will not be possible.

### **3.9.7 Number of index levels**

The number of levels of index records in the index. for an KSDS/VRDSDS cluster. If this number is greater than four (meaning a very big data set, maybe it is an indication for reorganization increasing the size of the CI index.

### **3.9.8 Time stamps**

At close time, if the cluster is open for output, the KSDS time stamps are updated with the current system time (same value for both data and index). However, each component's individual time stamp is updated only if it is less than the current system time stamp. Prior to updating time stamps, Close writes SMF record type 64. The timestamp ordering the SMF record should slightly precede that in the catalog.

At Open time, if the timestamp of the index component is less than that of the data component, the data component is updated separately and after the index component, or vice-versa.

---

### 3.10 DFSMSdss PRINT command

With the PRINT command, you can print:

- A single-volume non-VSAM data set, as specified by a fully qualified name. You must specify the volume where the data set resides, but you do not need to specify the range of tracks it occupies.
- A single-volume VSAM data set component (not cluster). The component name 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 need not be known.

---

### 3.11 SMF record types related to VSAM data sets

Following are the SMF records related to VSAM recovery and VSAM performance.

#### 3.11.1 SMF record type 60

Record type 60 is written when a record is inserted, updated, or deleted from a VSAM Volume Data Set (VVDS). For example, when a VSAM cluster is defined, closed, or deleted.

VVDS is a part of ICF catalog structure (the other is BCS), located in the volume which contains the described data sets. It contains dynamic information, as statistics, about these data sets. VSAM data sets and SMS data sets must be cataloged in an ICF catalog. The record related to a VSAM data set is a VSAM Volume Record (VVR), while the record 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:

- Identifies the VVDS in which the VVR or NVR is written or deleted
- Gives the new, updated, or deleted VVR or NVR.
- Identifies the job by job log and user identifiers.

### 3.11.2 SMF record type 61

One type 61 record is written for each record inserted or updated in a catalog. This record:

- Identifies the entry being defined and the catalog in which the catalog record is written
- Gives the new or updated catalog record.
- Identifies the job by job log and user identifiers.

### 3.11.3 SMF record type 62

Record type 62 is written at the successful or unsuccessful opening of a VSAM component or cluster. The record:

- 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.

### 3.11.4 SMF record type 63

Record type 63 is written when a VSAM catalog entry (a component, cluster, catalog, alternate index, path, or non-VSAM data set) is defined by the DEFINE Access Method Services command and when that definition is altered. For example, 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) or, 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:

- Identifies the catalog in which the object is defined.
- Gives the catalog record for the newly defined object, and, for an alteration, gives 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.

### 3.11.5 SMF record type 64

Record type 64 is written when:

1. A VSAM component or cluster is closed.
2. VSAM must switch to another volume to continue to read or write.
3. There is no more space available for VSAM to continue processing.

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.

#### 3.11.5.1 SMF record type 64 description

The record describes the device and volume(s) on which the object is stored, and gives the extents of the object on the volume(s). It gives statistics about various processing events that have occurred since the object was defined, such as 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 written when the VSAM component or cluster is closed contains changes in statistics from OPEN to time of EOV and CLOSE.

#### 3.11.5.2 SMF64 sample program

The IDCAMS LISTCAT command shows the cumulative number of EXCPs, since the initial load. Sometimes it is more important to know the number of EXCPs executed between OPEN and CLOSE, mainly when you are doing tuning in buffering. Section A.3, “Sample program to extract information from SMF record type 64” on page 214, contains sample source assembler code. It can be used for showing more information about your VSAM data sets. By using JCL parameters, you can determine an EXCPs threshold. Then, the program only shows data covering the data sets with equal or more EXCPs than the threshold that occurred between open and close. The report generated is based on the SMF 64 record, 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:

- 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.
- In the case of LSR buffering and direct access, you can find out if the data sets are using defer write, and if not, whether they could be.
- 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 doing changes in buffering, you can use the report to see how the numbers of EXCPs decreased.

For a description of SMF type 64 fields, refer to *OS/390 MVS System Management Facilities (SMF)*, GC28-1783.

#### **3.11.5.3 SMF record type 65**

Record type 65 is written during any processing that results in a DELETE request to Catalog management services, such as:

- IDCAMS DELETE
- IEHPROGM UNCATLG

One type 65 record is written for each record updated or deleted from a catalog. The record:

- Identifies the entry being deleted
- Identifies the catalog in which the catalog record is updated or deleted.
- Gives 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.

#### **3.11.5.4 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 written or deleted from a catalog. The record:

- Identifies the entry being altered.
- Identifies the catalog in which the catalog record is written or deleted.
- Gives the new, updated, or deleted catalog record.
- Indicates if 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.

#### **3.11.5.5 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 affected by the DELETE Access Method Services command. For example, three records are written for an indexed cluster; one for the relationship between the components of the cluster, one for the data component, and one for the index component. The record:

- 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 zeroes.

#### **3.11.5.6 SMF record type 68**

Record type 68 is written when a VSAM catalog entry (a component, cluster, catalog, alternate index, path, or non-VSAM data set) is renamed using the ALTER Access Method Services command. This record:

- Identifies the VSAM catalog in which the object is defined.
- Gives 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.

#### **3.11.5.7 SMF record type 69**

Record type 69 is written when a VSAM data space is defined, extended, or deleted 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:

- 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 unallocated 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.

#### 3.11.5.8 SMF record type 42

This record provides VSAM record level sharing (RLS) statistics.

---

### 3.12 Resource Recovery Management Services (RRMS) and VSAM

S/390 provides Resource Recovery Management Services (RRMSs), comprising:

- Resource Recovery Services (RRS), which provide sync-point services.
- Registration Services, which allow a resource manager to define itself to the operating system.
- Context services, which allow a resource manager to indicate interest in a work context. A context represents the resources for a work request; a context consists of the application program requesting the work and the protected resources involved in the work. A context represents a business unit of work: one or more units of recovery with the associated application programs, resource managers, and protected resources.

RRS is an OS/390 component capable to coordinate Resource Recovery in MVS.

Resource recovery includes a set of APIs and protocols allowing a transaction executing an application program to *modify* consistently multiple protected resources. The most common is 2-phase commit protocol.

Among these resources, we may have databases, VSAM data sets, and any product specific resource, managed by distinct resource managers. These resource managers maybe located in different systems.

Resource recovery scenarios has three agents:

- Application Program (AP): Requests the changes.
- Resource Manager (RM): Controls the access to the resource, for example, Data Manager (DB2, DL/I, VSAM) and Work Manager (CICS, IMS/DC),
- Synchpoint Manager (SM) guarantees resource recovery by the implementation of 2-phase commit. RRS is an example of an SM.

When an AP is running the same unit of work (transaction) under CICS and IMS/DC, it is able to update multiple data located in DB2, DL/I, VSAM files. In this case, there is not a need for RRS, because the SM role is executed by CICS or IMS/DC through the Commit and Rollback functions.

RRS allows Resource Recovery (2-phase commit) when an unit of work crosses multiple subsystems (MQSeries, CICS, IMS) and multiple OS/390 images.

Unit of recovery (UR) is a set of changes that *all* must be done or *no one* is done. It guarantees the integrity of the updates.

2-Phase Commit Protocol aims the execution of an UR, that is, all or nothing. Has two phases

- Phase 1:
  - AP informs the changes to RMs.
  - RMs log the old (UNDO) and the new (REDO) data.
  - AP asks a commit to Synchpoint Manager.
  - Synchpoint Manager asks RMs if they can commit (prepare commit). If all say “yes”, then Synchpoint Manager hardens the commit in its journal. If *not* all say “yes”, the commit is *not* hardened, and the backout is commanded at once to the RMs (erase the log) at phase-2.
- Phase 2:
  - Synchpoint Manager orders the commit (if hardened) or the backout (if not hardened) of the changes represented by the UR.
  - If all RMs return OK, Synchpoint Manager returns a return code committed to the AP. If not, then application changes will be made during restart.

Then, along a 2-phase commit, we may have two functions: commit or backout.

In a pure VSAM application guaranteeing the APIs for a 2-phase commit when you want to implement a unit of recovery, due to updates in multiple VSAM data sets, you will find the RRs to be very helpful.

For more information, refer to *OS/390 V2R8.0 MVS Programming: Resource Recovery*, GC28-1739.

---

## Chapter 4. Managing your VSAM data sets

In this chapter we provide practical tips for the daily care and feeding of your VSAM data sets. We have included a description of the OS/390 exploiters of VSAM functions.

---

### 4.1 Reorganization considerations

The new DASD storage technology is characterized by:

- Numerous high-capacity, small-size FBA, SCSI/SSA disks
- Redundancy in different types of RAID
- Generous amounts of cache, mainly to avoid the write penalty caused by RAID
- Plenty of microcode in order to support RAID, cache algorithms, the mapping between the disks and the logical 3390/3380 volumes (as seen by OS/390), and in the RVA case, the virtualization of the 3390/3380

Because of that, all the old considerations in order to avoid long seeks and RPS misses in a 3390/3380 logical volume are out-of-date. Does this apply to VSAM KSDS need of reorganization? Let us look at all the performance reasons (old and new) justifying the VSAM reorganization.

#### 4.1.1 CI/CA splits

Consider CI/CA splits, causing long seeks in the data set — this reason does not hold true in the new disk controller scenario. Do not reorganize your KSDS or VRRDS data set to avoid long seeks.

#### 4.1.2 The loss of useful space in data CA

The reasons causing this type of waste are as follows:

- There is no CA reclamation in VSAM KSDS/VRRDS. Then, if at load time a specific data CA was loaded with records belonging to an specific key range and later on the majority of them are deleted (and not re-utilized, as a timestamp key), the CA is underpopulated, and this free space is not reclaimed.

How do we measure that? To answer this question, refer to 3.9, “IDCAMS LISTCAT output fields” on page 166, and the diagram shown in Figure 22, as you follow along with our explanation.

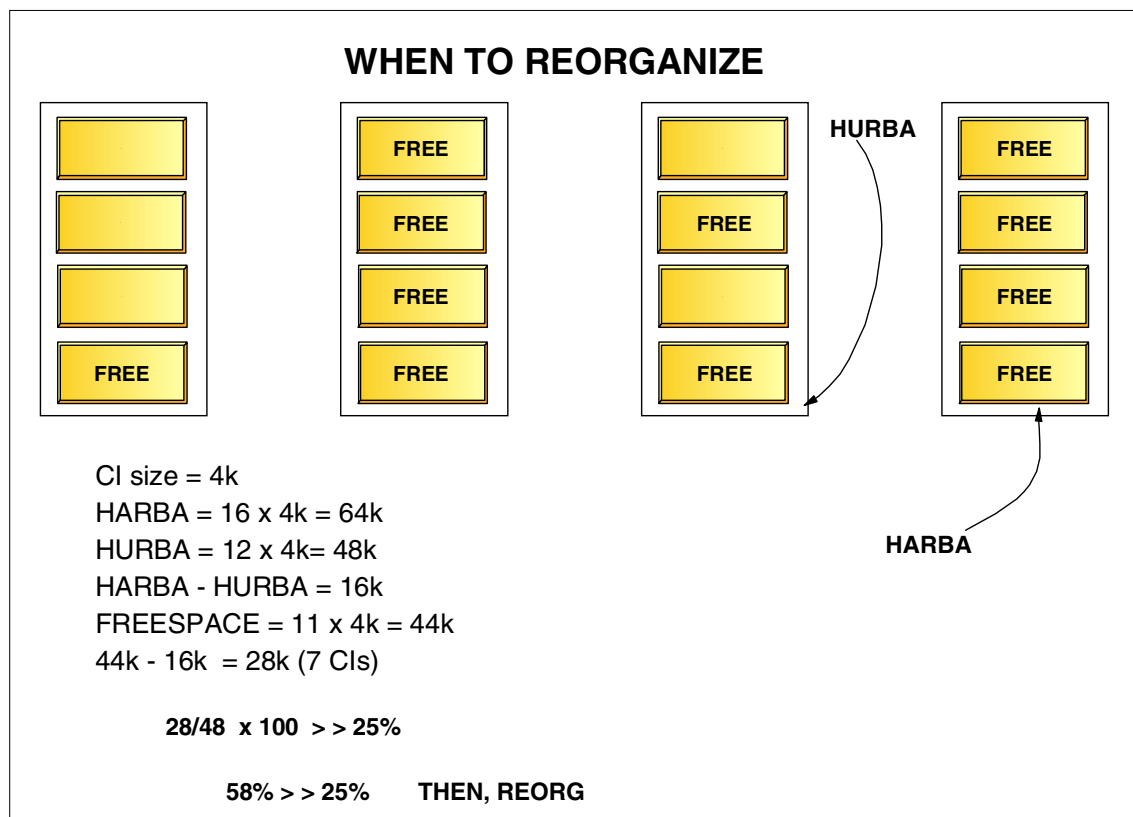


Figure 22. HARBA, HURBA, and free space

From this diagram showing HARBA, HURBA, and free space, you can see:

The subtraction: HARBA - HURBA = X gives the amount of free bytes in the free CIs allocated in the CAs beyond HURBA.

If you subtract FREESPC - X = Y it gives the amount of free bytes in free CIs embedded in CAs included in HURBA.

Y includes the CA Freespace%, 25% for example.

If  $(Y / HURBA) * 100$  is consistently greater than CA Freespace% (25%), and the number of deleted records (in catalog) is a large figure, it means that it is time to reorganize due to non-reclaimed CAs.

Following is a numeric example:

HI-A-RBA-----184320000 (HARBA)  
 HI-U-RBA-----176209920 (HURBA)

FREESPACE-%CA-----10  
FREESPC-----89616384

In view of these figures — Is it time for a reorganization?

- A strong argument to reorganize is the knowledge that those deleted keys are not going to be inserted again.
- Another reason to reorganize, is when at load time the sequence set Index CI is not big enough to contain information about all the data CIs in the data CA. In this case the CA is truncated and the rest is unused.

How do we measure that? The only way to distinguish between wasted CAs due to smaller index CIs or due to deletions is the previous knowledge of the deletions. Then:

- If  $(Y / \text{HURBA}) * 100$  is consistently greater than CA Freespace% and the number of deleted records in the catalog is a small figure, it is time to reorganize, due to the small size of the index CIs. The index CI size must be re-specified as a larger value.
- DFSMSHsm issues message ARC0909E advising that is time to reorganize the VSAM data set, due to a free space threshold being reached.

#### 4.1.3 CI/CA splits causing free space increase

After CI/CA splits the free space per CI (excluding the totally freed) tends to increase. In sequential read processing there is some impact on performance because more free bytes are moved to storage. Refer to 2.7.6, “I/O service time (connect) for VSAM files” on page 117.

How do we measure that? By the number of CI and CA splits in the catalog, together with the free space information.

A general comment about CI/CA splits, is that their number usually grows steady, after reorganization, till they reach a plateau. If this plateau is acceptable, do not try another reorganization.

---

## 4.2 Sharing VSAM data sets

There are three important mechanisms in order to implement VSAM data set integrity, when they are shared among users:

- VSAM SHARE options, playing an important role in VSAM integrity within one OS/390 image. They specify whether and to what extent data is to be shared among tasks in one or multiple OS/390 address spaces. Some

function is also provided for sharing among tasks in multiple OS/390 images.

- ENQ/Reserve serialization functions, mandatory for multiple OS/390 images environment. If is the case of using ENQ, you should specify EXCLUSIVE for writes and updates and specify SHARE for reads
- JCL Disposition (Old or Shr), which issues an ENQ implicitly
- Record Level Sharing (RLS) locking mechanism, a very sophisticated mechanism using the coupling facility. Share options do not apply to VSAM data sets in RLS mode. RLS is not covered in this book.

When you define VSAM data sets, you can specify how the data is to be shared within a single system or among multiple systems that have access to your data and share the same DASD. Before you define the level of sharing for a VSAM data set, you must evaluate the consequences of reading incorrect data (a loss of read integrity) and writing incorrect data (a loss of write integrity). These are situations that can result when one or more of the data set's users do not adhere to guidelines recommended for accessing shared data sets. On the other hand, it is important to avoid the unnecessary use of certain serialization functions which may cause a performance degradation.

#### **4.2.1 Write and read integrity**

When you share a VSAM data set among tasks (from the same or different OS/390 images) there is a need for write and read integrity.

##### **4.2.1.1 Write integrity**

Write I/O operation should guarantee integrity for non-atomic writes (the writes executed in multiple I/O operations), for example:

- Logical record update in-place, where a record is read, updated in memory and written back
- CI index updates due to CI data insertions and deletions (KSDS/VRRDS)
- CI and CA splits, which involves several write I/O operations
- Data set and catalog (usually VVDS) synchronous updates

To have write integrity, when several tasks are accessing the same data set, VSAM (and you) must guarantee that all the non-atomic writes are executed without be pre-empted. This means that no other related intervening write I/O operation should be executed.



#### 4.2.1.2 Read integrity

Read integrity guarantees that the record you read is the most current copy available, it implies:

- Within atomic writes, the related read I/O operation is suspended.
- The read I/O operation is able to find the most current copy of the logical record, meaning:
  - If the data is in the buffer pool, VSAM must guarantee the most current copy to satisfy the read (also called buffer pool coherency). In a shared environment with different OS/390 images accessing the data set, there are two ways of doing this: through Parallel Sysplex RLS cross invalidation (not covered in this book) or by VSAM refreshing the buffer at every read request (specified with SHAREOPTION 4 which we discuss later).
  - The write operation must send the current copy to where the next read can access it, even in a multiple OS/390 image system (to shared DASD or with Parallel Sysplex RLS, to the coupling facility).

#### 4.2.2 Who is sharing the data set?

You can share VSAM data sets between:

- Different step tasks (different ACBs) in a single operating system
- Multiple ACBs in a task or different subtasks
- One ACB shared in a task or different subtasks
- Different tasks in different OS/390 images. To share between different OS/390 images safely, you need Global Resource Serialization (GRS) or an equivalent product. Failure to use GRS or an equivalent can result in both data set, catalog and VTOC corruption.

For the sake of simplicity, we divide this topic into three types of VSAM data set sharing: intra-address space (tasks and subtasks in the same address space), cross-region (tasks within an OS/390 image in different address spaces) and cross-system (tasks in different OS/390 images).

#### 4.2.3 Intra-address space sharing

Before talking about intra-address space sharing, let us introduce the concept of a Resource Pool and the role it plays in the VSAM serialization.

Each VSAM cluster has a Resource Pool (RP) — shared or not with other clusters. A RP is formed by a buffer pool (BP) and a set of control blocks. When your program issues a GET request, VSAM reads an entire control interval into a buffer in the BP (or obtains a copy of the data from a control interval already in the BP). If your program modifies the control interval's data, VSAM ensures within a single control block structure (from RP) that you have exclusive use (locking) of the information in the control interval until it is written back to the data set, or moved back to the BP.

However, if the data set is accessed by more than one task at a time, through another RP, VSAM cannot ensure that your program has exclusive use of the data because the control block structure is not the same. In this case, you must obtain exclusive control, using facilities such as VSAM share options, ENQ/RESERVE, and JCL disposition.

Sometimes this locking in a CI basis done by VSAM may cause contention and even deadlocks. There are no dead lock detection and prevention algorithms implemented in VSAM. If you are facing these drawbacks a recommendation is to have less logical records in a data control interval, or in other words to have better lock granularity.

When in LSR mode you may choose between VSAM deferring (placing the requiring task in wait) the request until the resource becomes available (LEW, default) or VSAM returning the exclusive control return code X'14' to the application program (NLW). The application program is then able to determine the next action. These options are in the MACRF parameter in ACB.

Intra-address space sharing, also called subtask sharing is the ability to perform multiple OPENs to the same data set within a task or from different subtasks in a single address space and still share a single RP control block structure. Subtask sharing allows many logical views of the data set while maintaining a single RP control block structure. With a single control block structure, you can ensure that you have exclusive control of the buffer when updating a CI. All subtasks accessing the data set through this single RP control block structure, do not depend on VSAM Share Options, JCL DISP or GRS specifications to guarantee integrity.

The three methods of achieving a single RP control block structure for a VSAM data set while processing multiple concurrent requests are:

- A single access method control block (ACB) and a STRNO>1. Refer to DFSMS/MVS Using Data Sets, SC26-4922 to get more information about STRNO.

- DDname sharing, with multiple ACBs (all from the same data set, and located in the same address space) pointing to a single DD statement. Example:

```
//DD1 DD DSN=ABC
OPEN ACB1,DDN=DD1
OPEN ACB2,DDN=DD1
```

- 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. Example:

```
//DD1 DD DSN=ABC
//DD2 DD DSN=ABC
OPEN ACB1,DDN=DD1,MACRF=DSN
OPEN ACB2,DDN=DD2,MACRF=DSN
```

When implementing intra-address space sharing, it is necessary that when attaching new subtasks, the subpool zero must be shared by mother and daughter tasks in order to shared the RP control blocks (SZERO=YES in the ATTACH macro).

When a control interval is not available for the type of task processing requested (shared or exclusive), VSAM record management acts differently depending on the selected buffer options (NSR, LSR, or GSR), refer to 2.6.9, “Buffering options” on page 58, to have more informations on buffer options:

- NSR. The requester task gets a second copy from the buffer, with the exception of the case where the owner task and the requester asked for exclusive control, that is both are writing (updating or deleting). In this case the requester gets back a logical error in the RPL feedback code.
- LSR/GSR. The requester task:
  - Shares the same buffer (CI) with the owner task, if both are asking for shared control (read).
  - Gets a logical error in the RPL feedback code, if owner task has exclusive control (write, update or delete).
  - Gets a logical error in RPL feedback code, or is queued (placed in wait state) by VSAM until the buffer is released by the owner task which is in shared control (and the requester ask for exclusive). The NLW parameter in ACB forces the logical error, instead of the queueing.

As you can see, depending on the selected buffer option, NSR or LSR/GSR resources, GET requests to the same CI as that being updated may or may not be allowed.

When a logical error in RPL is presented, the application must decide whether to retry later or to free the resource causing the conflict.

A subtask has an opened ACB which shares a control block structure that can have been previously used. If this subtask now issues the POINT macro to obtain the position for the data set, it should not be assumed that positioning is at the beginning of the data set, as in a more normal situation.

#### 4.2.4 Cross-region options

The extent of VSAM data set sharing among tasks in distinct address spaces within OS/390 images depends on these considerations:

- The requester, through JCL data set disposition (DISP) and eventually local ENQs.
- The data set's cross-region share option, specified when you define (or allocate) the data set — these options are recorded in the catalog.

Cross-region options apply when we have multiple job steps with DISP=SHR in DD card pointing to the same VSAM data set. If you have DISP=OLD, the serialization is done at initiator level and share options are not needed.

Following are the cross-region options:

- (1)

VSAM ensures that only one task has the data set open for output or multiple tasks have the data set open for input only (all tasks in the same OS/390 image). Any other OPEN fails with a return code in ACB. In other words, VSAM is insuring total read and write integrity. The intention of input or output is declared in the ACB MACRF parameter, not the Open input or output options.

- (2)

VSAM ensures that only one task has the data set open for output while multiple tasks have the data set open for input (all tasks in the same OS/390 image). All other open for output fail with a return code in ACB. In other words, VSAM is insuring write integrity. If you require read integrity (with a better performance due to higher granularity than cross-regions (1), it is your responsibility to use the ENQ and DEQ macros appropriately to provide read integrity for the data the program obtains. The intention of input or output is declared in the ACB MACRF parameter.

- (3)

VSAM allows multiple tasks to open the data set for output or input. VSAM does not reject a request due to cross-region share options. The sharing tasks must ensure both read and write integrity through their own ENQs (including Open and Close processing on them). VSAM does not refresh the buffer pools for direct processing

- (4)

VSAM allows multiple tasks to open the data set for output or input. VSAM does not reject a request due to cross-region share options. The sharing tasks must ensure both read and write integrity through their own ENQs (including open and close processing on them). VSAM refreshes the *data* and *index* components buffer pools for direct processing (sequence set is not refreshed), to guarantee 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.

For share options 3 and 4, (for performance reasons) you may use ENQ share for Reads and ENQ exclusive for Writes. This improves the performance of reads with read integrity. If you do not issue an ENQ for reads, you lose the read integrity.

Protecting the cluster name with DISP processing and the components by VSAM share options is the normally accepted procedure.

#### 4.2.5 Cross-system options

The multiple access is done through tasks of the different OS/390 images.

Job steps of two OS/390 images may gain access to the same VSAM data set regardless of the disposition specified in each step's DD statement for the data set. DISP=OLD (without GRS) only serializes Jobs steps from the same OS/390 image. However, the serialization can be implemented and the access may be denied, if the SYSDSN ENQ name is made GRS global.

To get exclusive control of the data set's volume, a task in one system may issue:

- RESERVE macro. However, it is recommended that the RESERVE be converted to a GRS global ENQ instead, to improve the granularity of the serialization (RESERVE locks the full volume)
- ENQ macro. In this case the Qname and Rname must be GRS global

Following are the cross-system options:

- ( 3)

Specifies to VSAM that the data set can be fully shared. VSAM does not reject a request due to cross-system share options. With this option, each task is responsible for maintaining both read and write integrity for the data the program accesses. The RESERVE (or GRS global ENQ) and DEQ macros are required with this option to maintain data set integrity.

For share options 3 and 4 you may use (for performance reasons) GRS global ENQ share for Reads and GRS global ENQ exclusive for Writes. With these options you have write integrity and read integrity.

In this Share option VSAM uses control block update facility (CBUF). Refer to 2.3.6.5, "Control Block Update Facility (CBUF)" on page 56. Because CBUF CA splits and addition of a new high-key data set record are allowed.

The buffers are not refreshed with share options 3, but for tasks executing programs using LSR/GSR, they can be invalidate through MRKBFR macro and forced to be written through WRTBFR macro. Refer to, 2.8, "VSAM and SmartBatch" on page 121, to understand how to improve the performance of a (3 3) data set using SmartBatch.

- (4)

Specifies to VSAM that the data set can be fully shared. VSAM does not reject a request due to cross-system share options. *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 processing that does not change either the high-used RBA or the RBA of the high key data control interval. Then, control area splits and the addition of a new high-key record for a new control interval that results from a control interval split are not allowed. VSAM returns a logical error to the user's program if this condition should occur. If the task running your program does not satisfy the requirements described above, you require cross-system option 3, where due to CBUF, CA splits and addition of a new high-key record are allowed.

With this option, each task is responsible for maintaining both read and write integrity for the data the program accesses. The RESERVE (or GRS global ENQ) and DEQ macros are required with this option to maintain data set integrity.

Table 16 contains a summary of the share options.

Table 16. Relationship between share options and VSAM functions

Share Options (DISP=SHR)	VSAM Function Provided
(3 3)	CBUF and no buffer refresh
(3 4)	Data/Seq Set buffers invalidate. No CA splits
(4 3)	Data/Index buffers invalidate. CBUF
(4 4)	Data/Seq Set/Index buffers invalidate. No CA splits

#### 4.2.6 General share options — considerations

User tasks running user programs that ignore the write integrity guidelines in share options 3 and 4 can cause VSAM program checks, lost or inaccessible records, un-correctable data set failures, and other unpredictable results. This option places responsibility on each user sharing the data set. Refer to 3.7, “Broken data sets” on page 139.

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

As stated before in cross-region option 4 and cross-system option 4, buffers for direct processing are refreshed by VSAM for each request (or in other words, buffering is not saving I/O operations). Following are more details on this:

- Each PUT request results in the appropriate buffer being written immediately into the VSAM object's DASD. VSAM writes out the buffer in the user's address space that contains the new or updated data record. The data and sequence-set control interval buffers are marked invalid following the I/O operation to DASD
- Each GET request results in all the user's input buffers being refreshed. The contents of each data and index buffer used by the user's program is retrieved from the VSAM object's direct access device.

In addition, VSAM provides assistance to the application to aid in preserving the integrity of the data:

- When sharing data sets in a cross-regions or cross-systems environment, you should run IDCAMS VERIFY before opening a data set. VERIFY updates the catalog description of the data set and discards some

erroneous information that can result from improper closing of the data set. This erroneous information and its effects cannot be evident to all systems sharing the data set. VERIFY eliminates the problem if it is running as the first step of a job stream. Refer to 3.5.3, “VERIFY command” on page 135, for more information

- GRS serialization

Open/Close/EOV routines use ENQ in SYSVSAM.dsn.catname.IIOIB, to implement serialization when processing a VSAM data set, as well as to ensure proper sharing based on share options. Add the SYSVSAM Qname to the GRS RNL inclusion list to avoid integrity exposures

- Pay attention that during load mode processing, you cannot share data sets. Share options are overridden during load mode processing to (1 3). Refer to 2.6.7, “Initial load option” on page 54.

#### **4.2.7 Control Block Update Facility (CBUF)**

CBF is active, whenever a data set is opened with DISP=SHR, and share options (3 3) or (4 3). In this case, VSAM record management maintains a copy of the critical control block data in OS/390 common storage. Obviously, this common storage is available only to address spaces within your operating system. Passing this information to another operating system is your responsibility. CBUF eliminates the restriction that prohibits control area splits. However, under share options 4 these restrictions still exist.

Cross-systems sharing with CA splits can be accomplished (with integrity) by sending the VSAM shared information (VSI) blocks to the other host 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 VSIs sent to other OS/390 images has not changed and only a check occurs. Refer to the A.2, “Accessing the VSAM Shared Information (VSI)” on page 213, where there is an example about how to get the VSI.

About sending the VSI to the other OS/390 image, you may choose:

- XRC APIs, as: IXCCONN, IXCMMSGO, IXCMSGI
- APPC VTAM LU 6.2, as: RECEIVE\_AND\_WAIT and SEND\_DATA

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



Because programs in many regions can share the same data set, an error that occurs in one region can affect programs in other regions that share the same data set. If a logical error (register 15=8) or physical error (register 15=12) is detected, any control block changes made before the error was detected will be propagated to the shared information in common storage.

In *DFSMS/MVS Using Data Sets*, SC26-4922 topic Techniques of Sharing - Examples there are examples in how to implement VSAM sharing with integrity.

---

## 4.3 Catalog Search Interface

The Catalog Search Interface (CSI) is shipped as a component of base DFSMS/MVS 1.5. As its use is not wide spread, we take this opportunity to remind you of its availability and to point out its advantages over other methods of obtaining catalog information. The following sections describe:

- CSI setup
- CSI programming considerations
- IBM supplied sample program
- Real-world example of CSI usage with sample code

### 4.3.1 CSI setup

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 may be invoked by assembler programs, high-level language programs and REXX execs. See *Managing Catalogs*, SC26-4914 for a complete description of the interface.

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

- - Using a Generic Filter Key:

When requesting information from the CSI for specific catalog entries, you may specify a generic filter key. This key can contain the following symbols used 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 may specify a list of field names. For example, if you were only interested in the volume and the file sequence number for specific data sets, you could specify the catalog field names VOLSER and FILESEQ in the field name list when calling the CSI. Obtaining this information from IDCAMS would require you to use the IDCAMS LISTCAT ALL command and to scan the output to retrieve the desired information.

- Performance benefits

Using the CSI generally results in significantly 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 from the CSI results in additional performance improvements, since you eliminate the retrieval of information that you discard later.

- CSI programming considerations

The CSI is distributed as load module IGGCSI00 in SYS1.LINKLIB. It is reentrant and reusable, can be invoked 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* used to return error or status information
- A variable length *selection criteria list (for input)*
- *Work area* used to return the requested catalog data

- IBM supplied sample programs

IBM provides three sample assembler programs and one REXX exec in SYS1.SAMPLIB. Here we provide a short summary of their 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 given catalog. This 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 defined in a given catalog that reside on a specific volume. Such a list might be helpful in a recovery situation affecting that volume.

- IGGCSIRX is a REXX exec that produces a list of data set names matching a generic filter key. When you call it from a TSO/E session, it will prompt you for the filter key, and return matching data set names, their type, and volume definition.

---

## 4.4 VSAM exploiters

Following is a list of the applications that exploit VSAM functions.

### 4.4.1 DB2

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

### 4.4.2 Hierarchical File System (HFS)

HFS is an Unix Service data organization with no determined logical records - it is a byte string - made of embedded directories and data. It can be accessed by Posix code (a UNIX standard for operating system interfaces - APIs) running under OS/390 or by OS/390 applications. In the second case an HFS looks like a VSAM ESDS organization and is accessed in the same way.

### 4.4.3 CICS

CICS uses a VSAM linear data set for logging.

CICS/RLS control data sets are VSAM linear.

### 4.4.4 DFSMSHsm

DFSMSHsm has three control data sets, respectively migration (MCDS), backup (BCDS) and offline (OCDS). All of them are KSDS VSAM. In a multi MVS image environment these data sets can be shared between distinct DFSMSHsm instances. This environment is called HSMplex.

#### 4.4.4.1 Control Data Sets (CDS) Serialization

Prior to OY40882, the data sets serialization was done with hardware reserves on the index volume. This technique left the data set exposed to breakage, if the user did not have their index component on the same volume as their data component.

However, the name of the ENQ resource was not associated with the CDS name (it was a fixed name), then in the same GRSplex with more than one set of DFSMSHsm control data sets, we have performance problems due to false contention. In DFSMS/MVS V1.5, this problem was solved by appending the RNAME to the BCDS name.

This support helps customer to merge disparate DFSMSHsm systems into a sysplex piece by piece by using:

- VSAM RLS for all CDSs (exception CDSQ and CDSR). So, you may explore RLS data sharing in a parallel sysplex topology. You can specify CDSSHR=RLS in DFSMSHsm startup.
- A large CDS.
- Extended Addressing available for MCDS (greater than 16 GB) and OCDS (greater than 4 GB)

#### **4.4.4.2 Control data set sharing options**

DFSMSHsm provides an appropriate multiple OS/390 image serialization protocol to ensure read and update integrity of the CDSs accessed by multiple DFSMSHsm instances.

In relation to the VSAM sharing aspects of DFSMSHsm control data sets, it is recommended you use SHAREOPTIONS(3 3), as already defined by the starter set.

However, it requires an environment with GRS or equivalent function. These options allow an easy start in a multiple OS/390 image environment. For example, a DFSMSHsm instance can be started on one OS/390 image while a utility job is reorganizing the CDSs on another OS/390 image.

This share option is also recommended when accessing the CDSs in RLS mode.

Nevertheless, with the preceding VSAM SHAREOPTIONS (3 3), a data integrity exposure could still exist if DFSMSHsm is not active in all connected OS/390 images and periodic maintenance is being executed on the CDSs. Strictly controlled procedures need to be in place. One of them can be allocating the utility job with a disposition of OLD, which causes an exclusive enqueue on the SYSDSN resource for the CDS cluster name. Because DFSMSHsm must have a shared enqueue on the same resource, this approach prevents DFSMSHsm from running at the same time as the utility in a GRS environment.

#### 4.4.5 DFSMSrmm

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

#### 4.4.6 OS/390 data sets

Several OS/390 data sets are VSAM:

- SMF uses ESDS.
- ASM for page data sets uses ESDS.
- ASM for STGINDEX data set uses KSDS.

#### 4.4.7 Java/VSAM

Java started its life in 1991 as part of a Sun project called OAK, a software environment (“virtual machine”) and programming language aimed at cable television “set-top” boxes. To meet the objectives of this project, Java was designed from the outset to be small, portable, fast, and safe. These characteristics would later prove to be an essential part of Java's success, as they made Java an ideal language for the explosion in growth of the Web and the Internet.

##### 4.4.7.1 Java Language and JVM

Java is a high-level language, similar to C and C++. Java is claimed to be (relatively) simple when compared to C++, as many of the more error-prone aspects of C++ have been eliminated. Java has no preprocessor, pointers, or go to, only limited casting, and automatic garbage collection. So, there are good chances that if you write it, you can read it.

Java is also an OO programming language and execution environment that offers significant new opportunities for software development, inter-operability, and portable execution, but without all the complications. It has a single inheritance model, simple data types, and code that is organized into classes.

As part of this explosive growth, Sun released a Web browser called HotJava. To implement the browser, Sun licensed the source code for the HotJava environment, the Java Virtual Machine (JVM). It did not take long for technologists to realize the potential advantages offered by the JVM and its portable byte code.

Today, Java can be deployed in applications, in full executable programs that perform functions similar to applications written in traditional languages, and in applets, which are small reusable extensions to a Web browser. A Java program can also run in a Web server environment as a servlet thus extending the function provided by a Web server.

IBM provides a plug-in, WebSphere Application Server, that allows Web servers to run servlets.

As a language, Java is statically typed, and most type checking is done at compile time. However, run-time checks such as array bounds, are still required. A key function of Java is that numeric precision is defined with the IEEE floating point notation, which ensures the same results everywhere for mathematical calculations.

Figure 23 shows a simple Java class model. Java has a simple class hierarchy, and a single inheritance model. The example shows two types of classes: Vehicle and Building. Both classes can inherit from the class above. Classes further down in the hierarchy can inherit from either the Vehicle or the Building class but not both. Other object-based technologies also have single inheritance; some complex technologies have multiple inheritance, where a class could inherit from both the Vehicle and Building class.

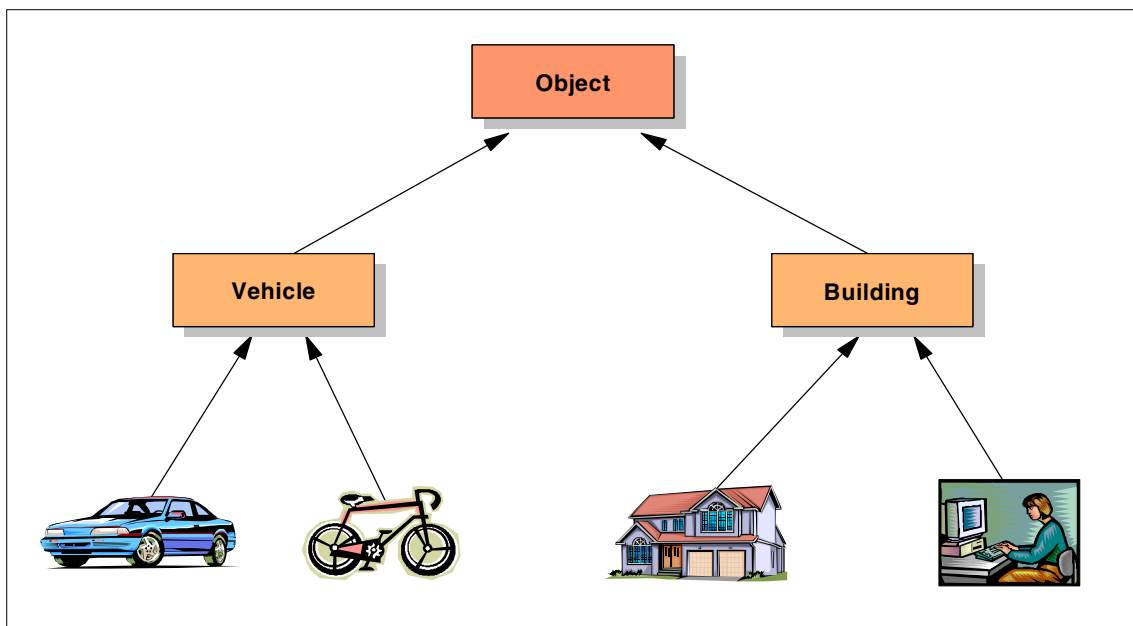


Figure 23. Java class model example

#### 4.4.7.2 Java Record I/O (JRIO)

JRIO is a set of APIs under OS/390 2.6 (JDK 1.1.8) level allowing the access of Java code to I/O record oriented data organizations. It is an additional to java.io APIs which only supports HFS type of access.

JRIO lets Java applications access traditional OS/390 file systems in addition to the Hierarchical File System (HFS). JRIO makes it easier for Java applications to access records within files and to access file systems through native methods when java.io Application Programming Interfaces (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 major differences are:

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

JRIO lets record-oriented applications (supporting multiple file systems) run using files on different file systems. It also provides a set of OS/390 native code drivers to access:

- Virtual Sequential Access Method (VSAM) data sets (KSDS only)
- Non-VSAM record-oriented data sets (sequential or random access)
- The system catalog (listing the High Level Qualifiers (HLQ) from the system catalog and data sets for a given 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, which reside in the same subdirectory as the Java for OS/390 classes.

Then, you should 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 Key Sequenced Data Set (KSDS). The VSAM support uses OS/390 native code to provide a set of concrete classes that implement the KeyedAccessRecordFile class.

This lets you access records:

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

In the A.1, “JRIO API examples” on page 209, you find a set of APIS that you may use to access KSDS VSAM data sets

---

## **4.5 Media Manager, Open, Close, EOVS in VSAM**

Media Manager is the I/O driver code. It stands between the access method and the Input Output Supervisor. VSAM has two I/O drivers depending on the required function:

- Block Processor (SVC 121), that is old and there is a SOD for being inactivated
- Media Manager, which is modern. There is a trend to all access methods to use Media Manager. Between its multiple roles, Media Manager has the I/O channel program support for implementing Extended Format.

Media Manager executes all these functions:

- Creates Channel Programs with virtual addresses (this was done before by VSAM)



- Page-Fix (or Page-Free) buffers
- Verifies that buffers are accessible in user key
- Translates virtual addresses to/from real addresses in the channel program
- Validates that RBA is within data set
- Re-drives Channel Programs to IOS (through STARTIO macro)
- Provides for DCME statistics
- Invokes SMFIOCNT

#### **4.5.1 OPEN macro**

Before an application program can access a data set, it must first issue the OPEN macro to open the data set for processing. The OPEN is issued against a user Access method Control Block (ACB). Opening a data set causes VSAM to:

- Mount the volumes where the data set resides. To get the volume information, VSAM examines the DD statement indicated by the ACB macro and the volume information in the catalog.
- Verify that the data set matches the description 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.
- Loads the access method (based in the ACB information) placing the address in the ACB for the next GET or PUT
- Check your program security authorization (RACF)

#### **4.5.2 CLOSE macro**

The CLOSE macro disconnects your program from the data set. VSAM does the following during CLOSE:

- Writes any unwritten data or index records whose contents have changed.
- Writes SMF records if using SMF.
- Updates the catalog entry for the data set. Updates the data set's high-used RBA (HURBA).
- Restores control blocks to the status they had before the data set was opened.

- Release virtual storage obtained during OPEN processing for additional VSAM control blocks and VSAM routines.
- For extended format KSDS, release all space between HURBA and HARBA if partial release was specified at OPEN.

If an abend happens during CLOSE, the HURBA which is updated during CLOSE processing, may be incorrect. In this case, VERIFY can correct the RBA. VERIFY determines the correct HURBA by reading the CIs.

The next VSAM OPEN performs an implicit VERIFY. If the VERIFY is not successful, VSAM OPEN passes a return code and reason code.

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

### 4.5.3 End-of-Volume (EOV) macro

EOV function is invoked by VSAM Record Management, when a VSAM data set requires additional space. The lack of this additional space is perceived by:

- High-used RBA (HURBA) = High-allocated RBA (HARBA)
- RBA of next CI/CA greater than HARBA during create
- No extent in the current volume contains a specific searched RBA

Then, EOV acquires new extents interfacing with DADSM, 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, so that this new space is accessible by all regions using this VSAM data set.

---

## 4.6 Transactional VSAM

Transactional VSAM is an enhancement in the DFSMS component of OS/390 release 10.

The main objective of Transactional VSAM is to add the ability to share VSAM data between CICS and batch for update as well as for reads. In doing this, the integrity of the data used by CICS cannot be compromised so, Transactional VSAM must offer the same features that CICS offers: logging for forward and backward recovery, backout and a two-phase commit process.

Transactional VSAM builds on:

- VSAM RLS
- OS/390 system logger
- OS/390 Recoverable Resource Services (RRS). Refer to 3.12, “Resource Recovery Management Services (RRMS) and VSAM” on page 179.

to add the ability for concurrent updates to the same VSAM data by CICS systems and batch jobs. It also adds a repeatable read option for direct use by VSAM to mirror the same facility provided by CICS. This means that, if a transaction or a batch job rereads a record, it will get the same data because any other transaction or job is locked out from updating that record.

Transactional VSAM extends RLS and uses the same locking protocols. It also shares the same restrictions as RLS. In summary, Transactional VSAM and RLS together add these functions to VSAM:

- Record level locking
- Commit and backout
- Undo and redo logging

Figure 24 shows the Transactional VSAM environment, following is a brief explanation of each part.

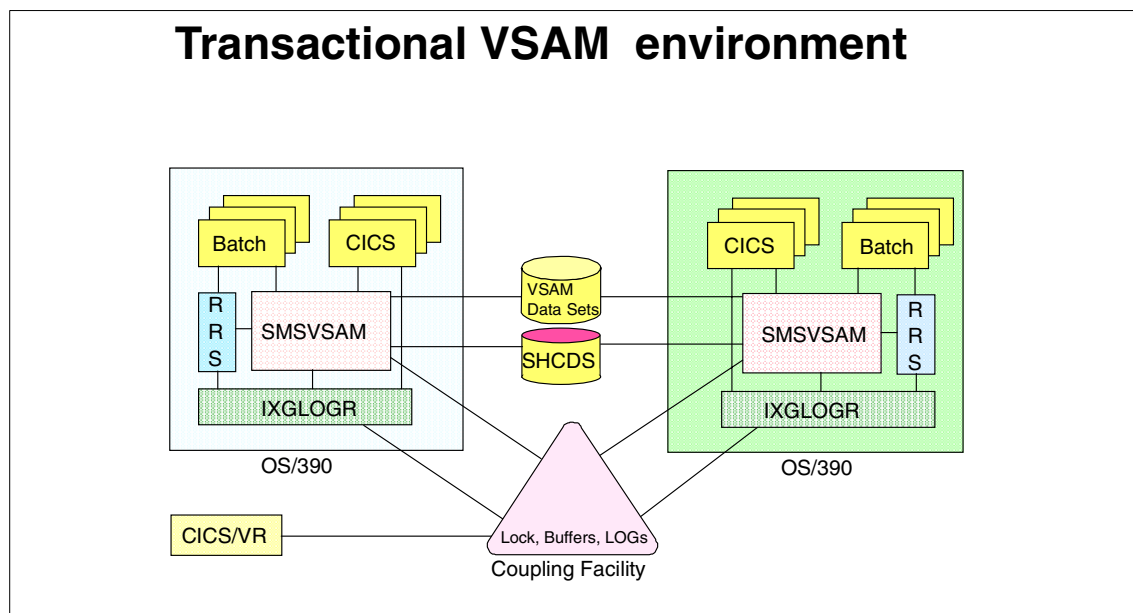


Figure 24. Transactional VSAM environment

### **VSAM data sets**

A VSAM data set being accessed by Transactional VSAM must be SMS managed. The VSAM data set organizations supported are KSDSs, ESDSs, RRDs, and VRRDs. You enable Transactional VSAM access to a data set by using the IDCAMS ALTER command to mark it as recoverable or by defining it with the LOG parameter set to UNDO or ALL. Figure 25 illustrates this:

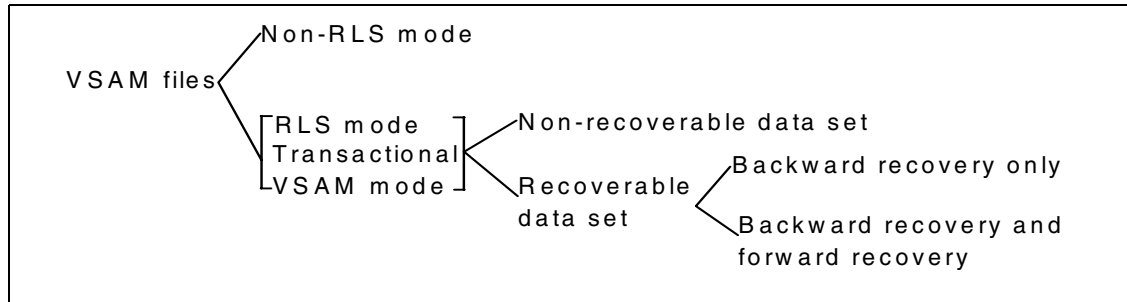


Figure 25. Modes of access to VSAM data sets

If UNDO is specified, then only undo logging is done for the data set; forward recovery logging is not done. If ALL is specified, then both undo and forward recovery logging are done and a log stream id needs to be specified. If you need to share data sets for update between batch and CICS, they should be defined as recoverable.

### **Resource Recovery Services (RRS)**

RRS is part of S/390 Recoverable Resource Management Services (RRMS) and is the most important component from a Transactional VSAM use perspective. For more about RRMS refer to 3.12, “Resource Recovery Management Services (RRMS) and VSAM” on page 179.

RRS is a system-wide commit coordinator and the use of RRS by other systems such as DB2 means that a full two-phase commit can be provided even though a program may access VSAM, DB2 and other services.

When migrating batch programs to a Transactional VSAM environment, frequent sync points are needed to avoid holding so many locks that other users are impacted by having to wait for the locks to be released. These sync points are managed by RRS. RRS uses the System Logger for logging.

### ***SMSVSAM address space***

In a Transactional VSAM environment, SMSVSAM is responsible for:

- Providing the necessary buffering and the locking needed to provide record level serialization.
- Providing logging or two-phase commit and backout protocols.

Providing these functions at the file system level allows batch (non-CICS) applications to access recoverable data sets for update without first taking access to those data sets away from CICS. This support is made possible by support that was put into OS/390, namely the System Logger and Recoverable Resource Management Services (RRMS).

Only one SMSVSAM address space is supported per OS/390 image. If the SMSVSAM address space fails, it will be restarted automatically.

### ***Coupling Facility***

The Coupling Facility (CF) provides locking, caching and list services for coupling-capable S/390 processors running OS/390. The Coupling Facility links are used to connect Coupling Facilities to the coupling-capable processors.

In a VSAM sharing environment, SMSVSAM uses a lock structure and multiple cache structures to manage VSAM buffering and locking. The lock structure and cache structures are mandatory, even if you only use Transactional VSAM within a single OS/390 instance.

### ***Sharing control data sets (SHCDS)***

The Sharing Control Data Sets (SHCDS) are a key element in maintaining data integrity in a shared environment. Because persistent record locks are maintained in the Coupling Facility, several new classes of failure need to be considered. These include a loss of connectivity to the CF, failure of an individual OS/390 system, an SMSVSAM address space restart or a Coupling Facility lock structure failure. The SHCDS is designed to contain the information required for DFSMS/MVS to continue processing with a minimum of unavailable data and no corruption of data when failures occur. The SHCDS acts as a log for sharing support. It is a logically partitioned linear data set that can be defined with secondary extents, although all extents for each data set must be on the same volume.

An SHCDS contains the following information:

- The name of the Coupling Facility lock structure in use
- The status for each system or failed system instance
- The time that the system failed
- A list of subsystems and their status
- A list of open data sets using the Coupling Facility
- A list of data sets with unbound locks
- A list of data sets in permit non-RLS state

If a permanent I/O error occurs for an active SHCDS, or if a SHCDS becomes inaccessible from one or more systems, it is automatically replaced by one of the spare SHCDSs. When a system is forced to run with only one SHCDS, it issues a message requesting that you add another active SHCDS. If a system does not have access to a SHCDS, all opens for Transactional VSAM processing are prevented on that system until a SHCDS becomes available. SMSVSAM will not start unless you have two active SHCDSs and at least one spare SHCDS.

### ***The System Logger***

The System Logger provides logging functions for CICS TS, SMSVSAM and RRS. The System Logger address space is named IXGLOGR. Each OS/390 image has one IXGLOGR address space.

The System Logger provides a single image log environment. This means that CICS TS, SMSVSAM and RRS do not need to care where the log is, whether in the CF or on DASD.

The log data area which subsystems use to write log entries is called a log stream. When IXGLOGR receives the log write request from CICS, SMSVSAM or RRS, IXGLOGR will write it to a log stream within the CF. If the log stream buffer becomes full, IXGLOGR will allocate a data set on disk and move the log data. This data set is called the *DASD log data set*. Figure 26 shows a system logger overview.

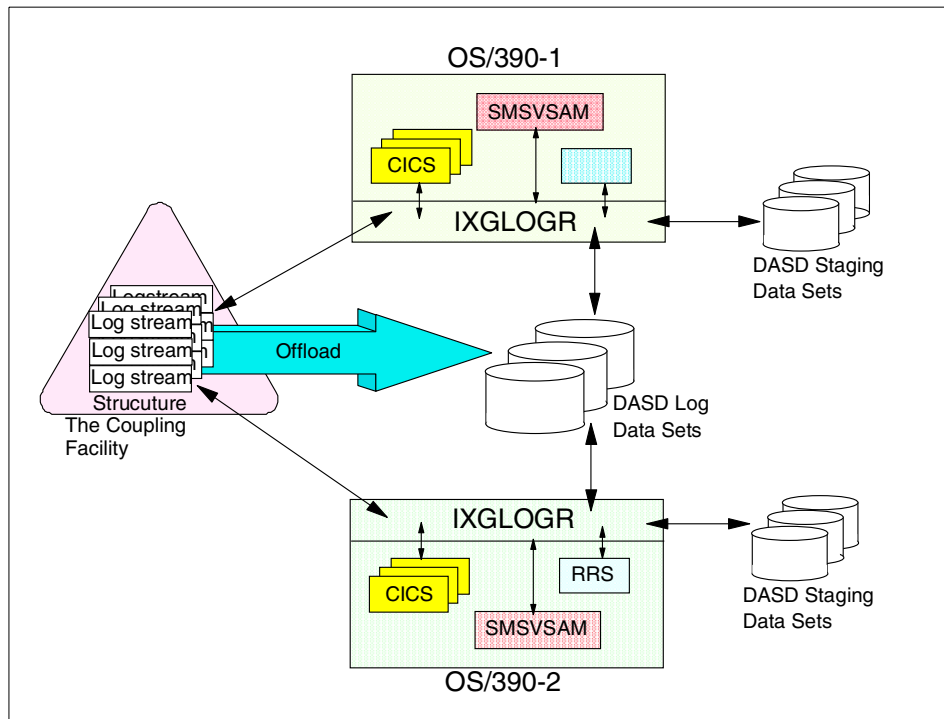


Figure 26. System logger overview

Transactional VSAM uses these log streams:

1. Backout or undo log stream, one for each SMSVSAM.
2. Shunt log stream: used when back out requests fail and for long running units of recovery. Each SMSVSAM has one.
3. User defined forward recovery log streams. When a VSAM cluster is defined with LOG(ALL), a log stream identifier must be specified. This log stream is used by Transactional VSAM to provide forward recovery capability.
4. Log of logs. This is an optional log stream and contains copies of log records that are used to automate forward recovery.

RRS uses five log streams that are shared by all the systems in the sysplex.

***Batch applications***

Batch applications may share recoverable VSAM data sets with CICS for update access using the existing VSAM programming interfaces when they access those data sets in Transactional VSAM mode. However, when they do so, the batch applications should issue frequent sync points to ensure that they do not tie up resources by holding locks for excessive periods of time and that they do not cause the undo log to become excessively large. To take a sync point, issue the appropriate RRS calls for a commit or a back out.

***CICS***

CICS does not use Transactional VSAM, so the CICS environment is not changed when migrating batch work to a Transactional VSAM environment. CICS continues to work not only as a user of VSAM RLS but also as the commit coordinator for CICS transactions and as a log writer for forward recovery and backward recovery.



---

## Appendix A. Sample code

This appendix contains useful JCL and code samples that can be modified and used in your installation.

---

### A.1 JRIO API examples

Here some examples of JRIO APIs for accessing VSAM data sets:

#### A.1.1 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();
```

#### A.1.2 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();

```

### **A.1.3 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();

```

#### **A.1.4 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();

```

#### **A.1.5 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();

```

---

## A.2 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:

- Load ACB address into register RY.
- To 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
```

- To 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 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.

---

### A.3 Sample program to extract information from SMF record type 64

The sample below can be used to find more information about jobs using heavily VSAM data sets. It is based on SMF record type 64 issued when data set is closed.

For those data sets, it extracts the following information:

- Data set name, jobname, ddname
- The kind of access used such as: direct (DIR), sequential (SEQ) or SKIP
- When direct access, if the records were accessed by KEY or RBA
- The buffering technique used (LSR or NSR)
- If the data set was open for input (IN) or output (OUT)
- For LSR, with direct access, shows if defer write was used (DF WTR)
- Where VSAM buffers were located (above 16M, R31 or below 16M, R24)
- Total number of free control intervals in the data component at close
- The number of EXCPs from open to close

```
//YOURJOB  JOB    '(VSAM - MHL)',
//          REGION=4M,NOTIFY=&SYSUID
//*-----*
//* SMF REPORT - COMPILING JOB
//*-----*
//COMP      EXEC  ASMACL,DPRTY=9,
// PARM.L='LIST,LET,XREF,MAP,AMODE=31,RMODE=24'
//C.SYSIN    DD    *
//          TITLE ' READ SMF  RECORD TYPE 64 AND EXTRACTS REPORT'
*
* FUNCTION -
*   THIS PROGRAM READS SMF RECORD TYPE 64 AND EXTRACTS
*   INFORMATION FOR DATA SETS THAT HAD MORE THAN 10,000 EXCPS
*   (YOU CAN ALTER BY PARM) FROM OPEN-TO-CLOSE.
* REPORT:
*   DATA SET NAME, JOBNAME, DDNAME, SYSTEM ID, HOW MANY EXCPS
*   FROM LAST OPEN-CLOSE, BUFFERING TECHNIQUE USED (NSR/LSR), *
```

```

*      LOCATIONS OF BUFFERS (ABOVE: R31 OR BELOW: R24), THE NUMBER*
*      OF FREE CONTROL INTERVALS, IF THE DATA SET HAS EXTENDED *
*      FORMAT (EXT.FORMAT) OR NOT (NON-EF), IF THE DATA SET WAS *
*      OPEN FOR INPUT (IN) OR OUPUT (OUT), FOR LSR AND DIRECT *
*      ACCESS, IF WAS USED DEFER WRITE (DF WTR), FOR DIRECT ACCESS*
*      IF THE RECORDS ARE ACCESSED BY KEY OR RBA *
*
*
*
* MACROS USED: OPEN, CLOSE, IDASMF64 *
*
* PARM:  NUMBER OF EXCPS BELOW NOT TO GENERATE REPORT *
*        LIMIT VALUE: 999999999 *
*
* RETURN CODES:  64 - PARM WITH MORE THAN 9 BYTES *
*                32 - PARM NOT NUMERIC *
*                32 - PARM NOT NUMERIC *
*                OR CLOSE ERROR CODE *
*
* ABENDS:          ON OPEN ERROR, ABEND CODE IS OPEN ERROR CODE *
*
* JCL TO EXECUTE THIS PROGRAM: (CUSTOMIZE PARM VALUE) *
* //READSMF EXEC  PGM=SMF64,PARM=40000 *
* //SMF          DD DISP=SHR,DSN=SMFDUMP_DATASET *
* //REPORT       DD SYSOUT=*,LRECL=140 *
*
*
* JCL TO GENERATE SMFDUMP_DATASET: *
* //STEP         EXEC  PGM=IFASMFDP *
* //INDD1        DD    DSN=SYS1.XXXX.MANX,DISP=SHR *
* //OUTDD1       DD    DSN=SMFDUMP_DATASET,DISP=(,CATLG), *
* //             LRECL=32760,RECFM=VB,BLKSIZE=0, *
* //             UNIT=SYSDA,SPACE=(TRK,(30,10),RLS *
* //SYSPRINT     DD    SYSOUT=* *
* //SYSIN        DD    * *
*               INDD(INDD1,OPTIONS(DUMP)) *
*               OUTDD(OUTDD1,TYPE(64)) *
* /* *
*
SMF64 CSECT
SMF64 AMODE 31
SMF64 RMODE 24
        STM    R14,R12,12(R13)      * LINKAGE CONVENTION
        LR     R12,R15
        USING  SMF64,R12            * BASE REGISTER
        LA     R15,SAVE              * LINKAGE CONVENTION

```

	ST	R15,8(R13)	
	ST	R13,SAVE+4	
	LR	R13,R15	* R13 LOCAL SAVEAREA
	L	R4,0(R1)	
	LH	R1,0(R4)	
	LTR	R3,R1	* ANY PARM?
	BZ	OPEN	* NO. USE DEFAULT VALUE
	LA	R15,64	* RC=64 PARM TOO HIGH
	LA	R1,9	* MAX LENGTH
	CR	R3,R1	
	BH	RETURN	
	BCTR	R3,0	* FOR EXECUTE
	SRL	R15,1	* RC=32 PARM NOT NUMERIC
	EX	R3,VALID	
	BNZ	RETURN	
	EX	R3,PACK	
	CVB	R3,DOUBLE	
	ST	R3,BELOW	
OPEN	OPEN	(SMFDUMP,,REPORT,(OUTPUT))	
	LTR	R3,R15	
	BNZ	ER_OPEN	
	PUT	REPORT,HEADER	
INITPAGE	LA	R8,55	* LINES PER PAGE
READ	GET	SMFDUMP	
	LR	R10,R1	* IDASMF64
	USING	SMFRCD64,R10	
	CLC	SMF64LEN,H6	* AVOID S04C FOR SMALL RECORDS
	BNH	READ	
	CLI	SMF64RTY,64	
	BNE	READ	
	TM	SMF64RIN,X'88'	* CLOSE?
	BNM	READ	
	TM	SMF64DTY,X'80'	* DATA SET?
	BNO	READ	
	TM	SMF64DTY,SMF64RLS	* RECORD LEVEL SHARED?
	BO	READ	
	LH	R2,SMF64ESL	* LENGTH OF EXTENT ENTRY PORTION
	LA	R9,0(R2,R10)	
	MVC	JOBNAME,SMF64JBN	
	DROP	R10	
	USING	SMFRCD64,R9	
	CLC	SMF64DEP,BELOW	* # EXCPS HIGHER THAN BELOW LIMIT?
	BL	READ	
	TM	SMF64MC1,X'01'	* USER MANAGEMENT OF I/O BUFFERS?
	BO	READ	



	TM	SMF64MC3,X'30'	* ICI OR GSR?
	BM	READ	
	LA	R7,MISC	
	TM	SMF64MC1,X'08'	* DIRECT ACCESS?
	BNO	S_SEQ	
	MVC	0(L'WDIR,R7),WDIR	
	LA	R7,L'WDIR+1(R7)	
	TM	SMF64MC1,X'80'	* RECORD ACCESSED BY KEY?
	BNO	S_RBA	
	MVC	0(L'WKEY,R7),WKEY	
	LA	R7,L'WKEY+1(R7)	
	B	S_BTECH	
*			
S_RBA	TM	SMF64MC1,X'40'	* RECORD ACCESSED BY RBA?
	BNO	S_BTECH	
	MVC	0(L'WRBA,R7),WRBA	
	LA	R7,L'WRBA+1(R7)	
	B	S_BTECH	
*			
S_SEQ	TM	SMF64MC1,X'10'	* SEQUENTIAL ACCESS?
	BNO	S_SKIP	
	MVC	0(L'WSEQ,R7),WSEQ	
	LA	R7,L'WSEQ+1(R7)	
	B	S_BTECH	
*			
S_SKIP	TM	SMF64MC2,X'10'	* SKIP SEQUENTIAL ACCESS?
	BNO	READ	
	MVC	0(L'WSKIP,R7),WSKIP	
	LA	R7,L'WSKIP+1(R7)	
*			
S_BTECH	MVC	0(L'WNSR,R7),WNSR	
	TM	SMF64MC3,X'40'	* LSR?
	BNO	I_MISC	
	MVC	0(L'WLSR,R7),WLSR	
I_MISC	LA	R7,L'WLSR+1(R7)	
	TM	SMF64MC1,X'04'	
	BO	IN	
	MVC	0(L'WOUT,R7),WOUT	
	LA	R7,L'WOUT+1(R7)	
	B	S_DEFWR	
*			
IN	MVC	0(L'WIN,R7),WIN	
	LA	R7,L'WIN+1(R7)	
S_DEFWR	TM	SMF64MC3,X'48'	* LSR AND DEFERRED-WRITE?
	BNO	S_BUF31	

```

MVC    0 (L'WDEFW,R7),WDEFW
LA      R7,L'WDEFW+1(R7)

*
S_BUF31 MVC    0 (L'WBUF,R7),WBUF
TM      SMF64MC3,X'01'      * BUFFERS 31 BITS ADDRESSING?
BNO     I_BUFF
MVC     L'WBUF-2(2,R7),=C'31'
I_BUFF  LA      R7,L'WBUF+1(R7)
L        R5,SMF64NFS      * UNUSED CONTROL INTERVALS AT OPEN
L        R1,SMF64DFS      * CHANGE IN UNUSED CI AT CLOSE
AR       R5,R1
L        R1,SMF64DCI      * CI SIZE

*
DROP    R9
USING   SMFRCD64,R10
TM      SMF64DTY,SMF64EA
BO      CI_OK
LA      R4,0      * IF NON-EXTENDED ADDRESSABLE
DR      R4,R1      * CONVERT TO NUMBER OF CI
CI_OK   CVD      R5,DOUBLE
MVC     0 (L'WFREE,R7),WFREE
MVC     L'WFREE(8,R7),MODEL
ED      L'WFREE(8,R7),DOUBLE+4
LA      R7,L'WFREE+9(R7)
MVC     0 (L'WNEF,R7),WNEF  * MOVE NON-EXTENDED
S_EXTF  TM      SMF64DTY,SMF64EF  * EXTENDED FORMAT?
BNO     NON_EF
MVC     0 (L'WEXTF,R7),WEXTF
LA      R7,L'WEXTF+1(R7)
B       MV_DATA

*
NON_EF  MVC     0 (L'WNEF,R7),WNEF  * MOVE NON-EXTENDED
LA      R7,L'WNEF+1(R7)

*
MV_DATA MVC     SYSID,SMF64SID
MVC     JOBNAM,SMF64JBN
MVC     DSN,SMF64DNM
DROP    R10
USING   SMFRCD64,R9
MVC     DDNAME,SMF64DDN
L        R2,SMF64DEP      * # EXCPS
CVD     R2,DOUBLE
MVC     EXCPS,MODEL
ED      EXCPS,DOUBLE+2
PUT     REPORT,RECORD

```

```

MVI    RECORD,X'40'
MVC    RECORD+1(L'RECORD-1),RECORD          FILL WITH BLANKS
BCT    R8,READ
PUT    REPORT,HEADER
B      INITPAGE
DROP   R9

*
CLOSFILE CLOSE (SMFDUMP,,REPORT)
RETURN  L      R13,SAVE+4
        L      R14,12(R13)
        LM     R0,R12,20(R13)
        BR     R14

*
ER_OPEN WTO 'OPEN ERROR',ROUTCDE=11
        ABEND (R3),,STEP

*
PACK    PACK DOUBLE,2(0,R4)
VALID   TRT    2(0,R4),TBNUM
*-----*
*   DCBS
*-----*
SMFDUMP DCB    DSORG=PS,EODAD=CLOSFILE,MACRF=(GL),DDNAME=SMF
REPORT  DCB    DSORG=PS,MACRF=(PM),DDNAME=REPORT,LRECL=140,RECFM=FBA
*-----*
*   AREAS DE TRABALHO
*-----*
SAVE    DC     18F'0'
DOUBLE  DS     D
BELOW   DC     F'00200'          CUSTOMIZE - DEFAULT 10,000 EXCPS
H6      DC     H'6'
HEADER  DC     140C' '
        ORG    HEADER
        DC     C'1'
        ORG    HEADER+1
        DC     C'DATA SET NAME'
        ORG    HEADER+46
        DC     C'JOBNAME'
        ORG    HEADER+55
        DC     C'DDNAME'
        ORG    HEADER+71
        DC     C'EXCPS'
        ORG    HEADER+78
        DC     C'MISCELLANEOUS'
        ORG    HEADER+131
        DC     C'SYSTEM ID'

```

[illegible]

```
//L.SYSIN DD      *  
  ENTRY    SMF64  
  NAME      SMF64 (R)  
/*
```



---

## Appendix B. Miscellaneous performance items

In this appendix we describe the lab environment used for our measurements throughout the book, a general discussion on caching, and common error messages indicating broken data sets and output from the EXAMINE command.

---

### B.1 Our laboratory

We ran a few experiments in order to clarify some performance and usability aspects of VSAM. However, it happened in a non-controlled environment, where we cannot guarantee the same level of multiprogramming, and the same load in our DASD controller. Nevertheless, the results are sound, if you take into consideration such variables as number of EXCPs, I/O Connect time, and CPU time in order to compare the runs.

The jobs are totally I/O bound as read-and-forget and write-from-thin-air.

All the experiments are made in an ESS controller with a relative load.

#### B.1.1 General lab description

The majority of them are accessing a KSDS master cluster with the following characteristics:

- LRECL = 300 bytes
- Keylength = 8 bytes
- Number of records = 450,002 (not uniform key values distribution)
- Free Space = 10% 10%
- Data component CI size 4096
- Index component CI size chosen by VSAM

There is also a physical sequential file where the 300-bytes records represents updates, inclusions and deletions in the master. They can be organized sequentially or randomly (according to the experiment). About 10% of them have repeated keys causing revisits to the same record in the master. There are holes in the key values and some of them are cluster to emulate mass insertion or deletion.

DIRECT GET: 50000 records, 20%

GET SEQUENTIAL: Reads sequentially all records of the VSAM data set (450,002 records), with minimum record processing. Tests executed before insertions, so there were 12 record per control interval

UPDATE Program: insertions(3%) update (30%). Splits after insertions:

- Data component:
  - control interval: 5457
  - Control Area: 61
- Index component:
  - Control interval: 61
  - Control area: 1

### B.1.2 What do we measure?

To compare the performance results among the runs, we select the following variables:

- Total I/O connect time

*Connect time* is when the channel is transferring data from or to the cache. As we keep in all the experiences, always the same volume located in the same controller and accessed by the same channel type (so, always the same data rate), the connect time is an indirect measurement about how many bytes were transferred along the I/O operations. Then, the only way of decreasing the total I/O connect time is by moving less data to or from storage.

Variations in the load of the controller should not affect this value.

The total connect time was captured with GTF.

- Total I/O Disconnect time

*Disconnect time* occurs when the channel is not doing activities related to the execution of the channel program along the I/O operation. It means that the target record for a read is not in the cache and the disk access is a must. Then, the only ways of decreasing the total I/O disconnect time is by moving less data to or from storage or improving the use of the cache

Variations in the load of the controller should affect this value.

The total diconnect time was captured with GTF.

- Number of EXCPs

In SMF account information for SAM data sets, one EXCP equates one physical block transfer.



For VSAM data sets, one EXCP equates to one real I/O operation for data or index. It is not true, that the number of EXCPs for VSAM measures the number of transferred CIs. Therefore, one EXCP in may mean more than one CI being transferred. The number of CIs per I/O depends very much in the number of buffers and in the buffering technique. Consequently, the number of EXCPs is not repetitive, that is, the same job reading the same data set may present a different number of EXCPs. The reasons justifying the number of I/O operations (instead of CIs) are:

- We can measure the VSAM capacity in saving I/O operations and consequently saving TCB and SRB time.
- The number of I/O transferred CIs can be caught from LISTC in the catalog or from SMF record 64

In our measurements, the number of EXCPs corresponds to the number of I/O operations. Variations in the load of the controller should not affect this value.

- Elapsed time

Elapsed time is the wall clock time to run the job used in the test. If the I/O is faster, in I/O bound job the elapsed time should be less.

Variations in the load of the controller and the system should affect this value.

- Total CPU time (TCB and SRB)

All the runs are totally I/O bound, meaning no processing at all. Then the TCB time can be charged to the preparation of the I/O operation, including VSAM buffer pool management. SRB time here is spent along I/O interrupts processing only.

Variations in the load of the controller and system should affect very mildly this value.

- Total number of transferred bytes

Here, we measured only the data bytes, excluding the index bytes. The only way of decreasing it is by moving less data to or from storage.

Variations in the load of the controller should not affect this value.

The total number of transferred bytes was captured with GTF.

### B.1.3 DASD cache concepts

A cache is a fast storage (no mechanical movement) located in the DASD controller with two functions: to minimize access to disks (by having hits) and to serve as a *speed matching buffer* to synchronize elements with different speeds as channels and disks in a cache miss. In this appendix the word disk does not have the same meaning of DASD. Disk implied the RAID media (FBA, SSA or SCSI) used by modern controllers. DASD still means the logical 3390/3380.

In order to have random hits (saving disk access) for reads and writes, the I/O workload must revisit its data, however in certain cases it does not happen. In this case such workloads are called *cache unfriendly*. Usually, we may have two types of hits for VSAM data, when the application revisits:

- Exactly the same logical record in a CI already in cache
- The same data CI (already in cache) because other logical record (a sort of lucky)

For sequential access, it is important to say that, cache does not save data CI disks I/O operations. The cache only tries to match the speed of the disks and channels.

With the new controllers a cache miss implies accessing the RAID disks and 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, it is not important anymore the relative location of files in order to avoid long Seeks or even RPS misses.

It is interesting to note that heavy cache access reduces DASD skew (unequal 3390/3380 device utilization) because data formerly in heavily used devices now became electronically accessible due to the LRU algorithm. Refer in DASD Activity report this phenomena:

#### DASD Activity Report

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

Increasing the I/O rate, decreases the disconnect time (less the disconnect more hits in cache) due to the LRU algorithm is keeping in cache the most used elements.

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 we use the word cache meaning the volatile cache and NVS for the non-volatile.

The cache is made of CMOS DRAM storage for Data and Directory. The directory entries describes 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 (to be covered later) and it is asynchronous with the I/O operation which executed the writes. By the way, 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 done. There are three types of destaging:

- To relieve contention, when NVS or Cache become full. Controlled by modified LRU algorithms, when the percentage of NVS occupancy is greater than X%. If greater than Y% ( $Y > X$ ), then the DFW I/O operation bypass the NVS cache going synchronously from volatile cache to disk (called 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 an hardware error

The amount of data destaged is usually more than one 3390/3380 track. A smart controller can identify other records changed in adjacent tracks of the record to be destaged and destage all of them. It saves rotational delays in the disks and bypass 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 may overlap with staging/destaging operations, for the same 3390/3380 logical device. It is possible to have concurrent access to same 3390/3380 device data but not in the same track, for example, 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 here there is just one active I/O operation, that is the one with the hits in the cache, the destage is just controller housekeeping.

Controller accounts data about number of I/O operations, hits, misses, destages, in a volume or in a data set basis. These values are shown by IDCAMS LIISTDATA command, by RMF Cache report and 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 may be asked explicitly by software through the Define Extent CCW (per each I/O operation), in some cases the controller can adaptively change the mode due to the observed pattern of access. Following is the description of such CCW.

Define Extent CCW provides a 16 bytes parameters, which define limits on subsequent operations (extent information, avoidance of writes, for example), provide a blocksize value, and specify caching control mode (or hints) for the channel program. The caching mode have the granularity of I/O operation. These modes are not totally mutual exclusive, and the same I/O operation may have more than one mode. Following are the modes:

- Track Level Cache (TLC)
  - Record Level Cache (RLC)
  - Sequential:
    - Reads:
      - Sequential Access
      - Sequential Pre-stage
    - Writes
  - Least Recently Used (LRU)
  - Normal Caching
  - CFW
  - DFW
- With the possibility of Quick Writes

- Bypass Cache
- Inhibit Cache Loading

#### B.1.4 Cache Modes

Following is the explanation of these modes:

- Track Level Cache (TLC)

In TLC the unit of transport between cache and disks are the 3390/3380 tracks. TLC is more adequate to sequential. When in TLC mode the 3390/3380 track is staged in cache in one pass (if first miss is in record zero (R0)) or in two pass (if first miss is not in R0). RVA only has track level cache because the compact/compress data cause little traffic when moving logical 3390/3380 tracks.

- Record Level Cache (RLC)

In RLC the unit is the referred logical 3390/3380 physical record. RLC is very adequate to random (also called direct) processing.

Let us explain what do we mean by “logical 3390/3380 physical record”. The word *logical* indicates that the 3390/3380 does not real exist. The word *physical* indicates that we 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 out weights the benefits. RLC reduces the costs associated with cache miss I/Os by staging less data into cache (less cache pollution), which frees the volume and other activity more quickly. Reading less data into cache also enables data to stay in cache longer, which increases the chances of future cache hits.

ESS controller is able to switch from one to the other depending on the access pattern, independently of the Define Extent CCW.

RLC is mutually exclusive from TLC

RLC can be activated for VSAM SMS managed maybe-cache data sets. SMS via DCME decides in Define Extent when to use RLC for reads. Refer to B.1.6, “Using cache in an SMS data set” on page 236. On top of deciding to enable or to disable the cache for maybe-cache SMS data sets, SMS picks up between RLC or TLC for reads.

- Sequential

When in sequential mode, the controller uses the cache just as a speed matching buffer, to synchronize different speeds. There are two types:

- Sequential Read

In sequential read mode the controller does the pre-staging of a certain number of future referenced logical 3390/3380 tracks. After used, the tracks are or not demoted from cache depending on the sub mode:

- Sequential Access, where the used data is a strong candidate to be demoted
- Sequential Pre-stage, where the used data is protected by the LRU algorithm

Sequential read can be activated:

- 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).

Because KSDS/VRRDS VSAM organizations (in certain conditions) do not use the Define Extent. Then, in this case, it is interesting to avoid CI /CA splits in data sets usually processed sequentially. Splits make the logical sequence different from the physical sequence, and the controllers only detects the physical sequence pattern

- Least Recently Used (LRU)

LRU is not properly a mode, but a technique to maintain in the cache the most referenced elements. It is the major algorithm to control cache demoting, admitting that, if an element was referenced in the past, it is going to be again 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 LRU update (meaning that the data is to be kept in cache for a while).
- For a read miss, the channel is disconnected, the disk is accessed to stage data to cache (here, we may have the delay caused by the disk being already busy or all lower interfaces are busy). After that, the channel is reconnected and the data is transferred to channel from

cache, CE/DE I/O interrupt, stage rest of track (if in track mode), LRU update.

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 only occur when the internal path to disks are busy and the disk needs a new revolution.

- For a write hit, because the NVS cache is not used, it is like a miss. The channel moved data to volatile cache and disconnects. The disk is accessed synchronously to write data (here, we may have the delay caused by the disk being already busy or all lower interfaces are busy).

The word *synchronously* means that the write to disk is done without the end of the I/O operation be posted to the application.

After the write to disk, the channel is reconnected and CE/DE I/O interrupt is presented. LRU update (meaning that the data is to be kept in cache for a while)

- For a write miss the channel move data to cache and disconnects. The disk is accessed synchronously to write data (here, we may have the delay caused by the disk being already busy or all lower interfaces are busy).

The word *synchronously* means that the write to disk is done without the end of the I/O operation be posted to the application.

After the write to disk, the channel is reconnected and CE/DE I/O interrupt is presented. No LRU update, meaning that the cache copy if the data is ready to be demoted

- Cache Fast Write (CFW)

Is used for temporary data sets and consequently does not use the NVS for writes. It must be allowed by the Set Subsystem CCW issued by IDCAMS.

- For Reads and Write Miss 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 and the directory entry is LRU update (meaning that the data is to be kept in cache for a while). Later on asynchronously the data will be destaged to disks.

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 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 and the directory entry is LRU update (meaning that the data is to be kept in cache for a while). Some modern controllers as ESS sends the CE/DE earlier with one copy of the data in the NVS (other copy in the channel adapter buffer) doing the copy to the volatile cache immediately after the CE/DE.

Later on and asynchronously the data will be destaged to disks. However, if the NVS cache is under stress the controller is smart enough in sending the data from volatile cache directly to disks (synchronously). This situation is called DFW bypass and the channel stays disconnected along this data transfer. If you recall the dam story is like creating a bypass in the dam...

- For a write miss is identical to normal mode, with the difference that the LRU is updated.

- 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, its 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 (provided that adequate NVS space is available).

The reason causing the access of disks for a write data miss in a DFW mode is the verification of the record length. To clarify this point refer to Figure 27 and follow the description of the existent 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 and creating a count with the respective data (usually with a zero content), the rest of the track is erased. The length of the data record is informed 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, change the contents of the data portion of a pre-formatted record which length is already described in the count. The length of the data record is also informed 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) and in some cases, it may stop the execution of the channel program. Then is clear the need of accessing the 3390/3380 track in a DFW write modified miss because the controller is not able to verify (and compare) the length in the write modified CCW and the count. On the other hand is now clear why the write format is always a hit, because there is no need of such verification.

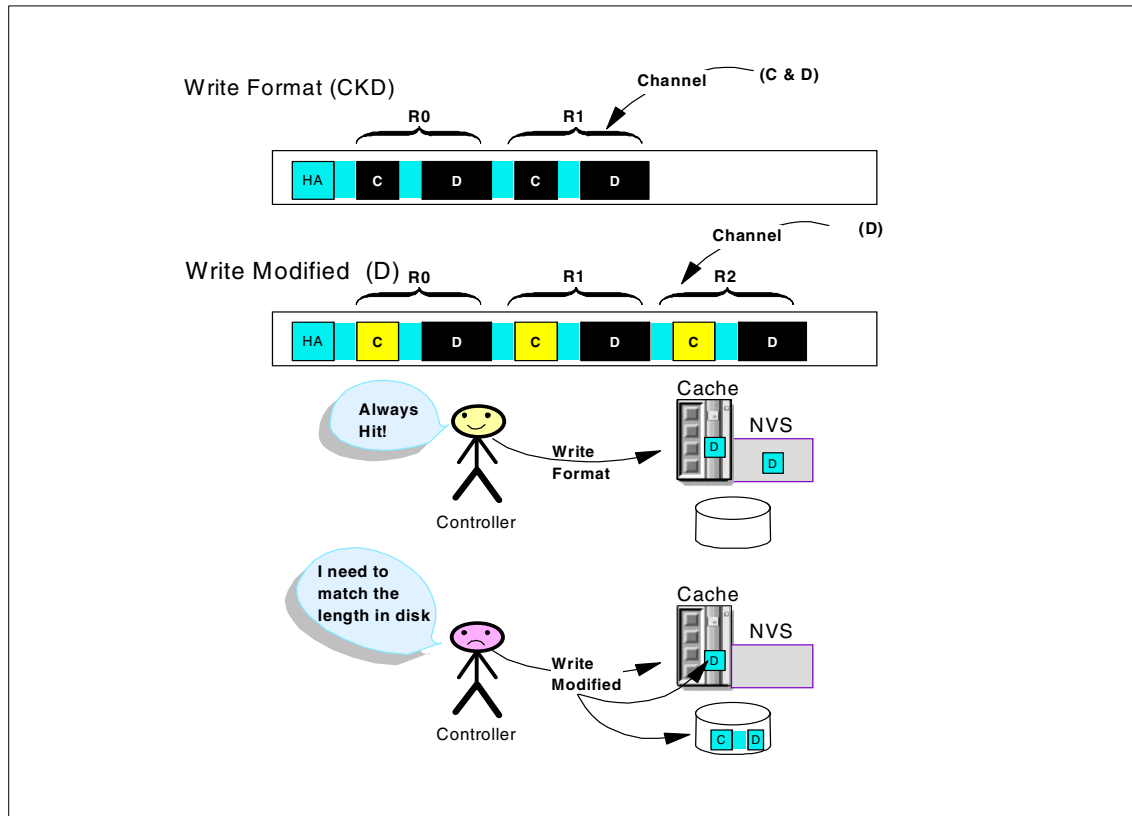


Figure 27. Types of writes

For quick writes the controller must know or be able to predict the length of the record avoiding the access to the disks.

Quick writes is implemented in two sub-modes 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:

It is a joint VSAM/controller implementation.

Because VSAM is a trustable partner (regular data format) all writes become quick writes, that is, the controller decides not to go to disk to verify the data record length. This function benefits IMS, DB2 and CICS users of VSAM. The data set could be SMS or not, it requires DFSMS/MVS 1.2 or PTF equivalent.

- Record Level Cache II:

It is non-VSAM adaptive (the controller does 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 requires:

- 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, read from cache and LRU update. If a read miss or a write access the disks through volatile cache, no LRU update. Then, 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 (Sortin), SMS (never-cache and some of the maybe-cache data sets). As a general rule, it maybe used by an application which knows that the record to be requested is not going to be used in the near future (as for cache unfriendly accesses).

Ignore ICL is an option set in VPD, when your workload is not aware of a huge cache size

- Bypass Cache

Where the I/O request must be executed in the disks (even if the data is in the cache). However, the data always pass through cache without LRU update, consequently the copy is ready to be demoted. If a hit in the cache, the LRU is updated. Cache images of tracks modified on the device will be invalidated. Tracks modified in cache but not on DASD are destaged to DASD before access is allowed. For Duplex volumes this includes destaging to both devices. DASD Fast Write operation is disabled with this attribute active.

It is used by ICKDSF, paging, and Write Check access method option.

Ignore BYP is an option set in VPD, being used when your workload is not aware of a huge cache size

### B.1.5 Using cache modes in a non-SMS data set

An I/O operation towards a non-SMS data set is able to use some of the cache modes. In order to do that, you must use IDCAMS Setcache command to activate:

- Globally in the controller:
  - Cache in general
  - DFW (or the use of NVS)
  - CFW

RMF in Cache Activity report shows the result of such operation:

Cache Subsystem Status

----- CACHE SUBSYSTEM STATUS -----				
SUBSYSTEM STORAGE		NON-VOLATILE STORAGE		STATUS
CONFIGURED	256.0M	CONFIGURED	8.0M	CACHING
AVAILABLE	254.9M	PINNED	0.0	NON-VOLATILE STORAGE
PINNED	0.0			CACHE FAST WRITE
OFFLINE	0.0			IML DEVICE AVAILABLE

- Locally in each volume:
  - Normal cache
  - DFW (or the use of NVS)
  - Dual Copy

RMF in Cache Activity report shows the result of such operation:

Cache D

VOLSER	D83STE	NUM	0D84
-----			
CACHE DEVICE STATUS			
-----			
CACHE STATUS		DUPLEX PAIR STATUS	
CACHING	- ACTIVE	DUPLEX PAIR	- NOT ESTABLISHED
DASD FAST WRITE	- ACTIVE	STATUS	- N/A
PINNED DATA	- NONE	DUAL COPY VOLUME	- N/A

These options are passed to the controller by IDCAMS. Without SMS all the data sets in volume follow these rules (normal or DFW). The other cache modes must be declared explicitly by the requester through an IOS, which builds the Define Extent CCW.

## B.1.6 Using cache in an SMS data set

SMS uses the Define Extent CCW to set some cache modes along 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: IDCAMS says non-DFW for the volume and Define Extent says DFW for the I/O operation, then it will be non-DFW.

However, there are certain cache modes not set by SMS as bypass cache and CFW. In this case the requester should use interface directly with IOS in order to have these options in the Define Extent CCW.

### B.1.6.1 Cache usage attributes

An opened SMS data set, may have one of three cache usage attributes, as DASD cache is concerned:

- Must-cache data set, which uses cache/NVS for the I/O operations
- Never-cache data set , which does not use cache (only for buffering) and does not use NVS. The ICL mode is requested in Define Extent CCW
- May-cache data set, which uses cache/NVS depending on the cache/NVS constraints

The same data set may have one of the above 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 (SC) installation parameters (MSR for Direct, MSR for sequential or BIAS respectively). MSR is the desired I/O service time in milliseconds and BIAS the expected dominance of reads or writes operations.

.The cache usage attributes are assumed at open time, for a data set. These cache attribute is kept constant till the data set be closed. Refer to Figure 28. that xplains how the cache usage attribute is determined based in the MSR value.

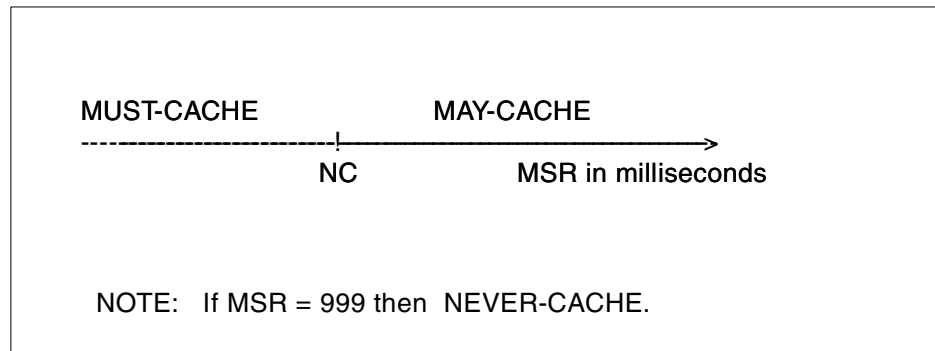


Figure 28. Association between MSR and cache usage attributes

Where NC is the native capability in milliseconds of the 3390/3380 logical volume without DASD cache for a 4-Kb data movement. That is, the theoretical average amount of milliseconds to execute a channel program without using the cache. NC depends on the DASD device type (3390 or 3380). The values of NC has not been published so far. Therefore, to have a must-cache data set, you may declare a MSR value below 10 milliseconds and to have a may-cache data set something above 100 milliseconds.

In the present design there is not a practical difference between a two or a five milliseconds, assuring that both values are below NC. The same is true for the values above NC. The fact that 3390 or 3380 does not exist anymore does not play in the determination of the cache usage attributes, if you understand how it works.

If the BIAS parameter is indicating R (a dominance of reads) and MSR is less than NC, then the cache usage attribute is must-cache for reads and may-cache for writes.

If the BIAS parameter is indicating W (a dominance of writes) and MSR is less than NC, then the cache usage attribute is must-cache for reads and must-cache for writes.

The majority of data sets in a installation should be may-cache in order to have a best utilized and self tuned cache.

Some data sets on a cached volume such as VTOC, VTOC Index, and VVDS are always must-cache.

Note that the MSR and BIAS values you specify in the storage class (SC) can be used to determine how buffers are to be allocated when system-managed buffering is used for VSAM applications. Please refer to XXX to get more information on system management buffering.

An I/O operation towards a may-cache data set maybe or maybe not cached, depending on the analysis of the controller global cache statistics and a data set cache usage.

#### **B.1.6.2 Dynamic Cache Management Enhanced (DCME)**

DCME is a function in the controller able to produce cache information in a system and in a data set basis. SMS uses data from DCME in order to adjust the use of the cache for may-cache data sets. With DCME, SMS distinguishes between good (cache-friendly) and poor cache (cache unfriendly) candidate data sets when deciding 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 achieved by I/Os to each may-cache data set. DCME continuously updates this information so that it can make decisions on which I/Os to which data sets should be cached, based on the most recent I/O activity.

When considering whether or not to cache I/Os to a data set, two controller resources must be accounted for: the cache and the NVS. A data set might very 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 specific 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 than 0.5 ms.

These indicators are weighed averages of previous values to avoid sudden changes.

- Subsystem load (here the word subsystem means DASD controller)

The DCME in controller produces global data about the cache usage.

A Subsystem Threshold (ST) indicator is timely calculated by SMS from the global DCME data. Higher the ST more cache contention. These statistics are periodically collected by SMS. The time is controlled by DINTERVAL at IGDSMSxx Parmlib (default 150 seconds).

ST reflects the DASD cache performance and is composed by the figures of 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, due to contention. In this case, the I/O request bypass the NVS and is executed directly from the volatile cache to the disks.

Also, based on the statistics, SMS maintains two global average indicators:

- Cache Control Indicator (CCI), the percent of may-cache I/O requests allowed to use cache.
- NVS Control Indicator (NVSCI), the percent of may-cache DFW I/O requests allowed to use NVS.

These values used for such calculation can be displayed on D SMS,CACHE, an MVS command:

```

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

```

Following is the legend:

- Ssid = Subsystem identifier
- Devs = Number of managed devices attached to 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 due to 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, in order to inhibit the staging of the cache. In the case of a Write channel program, the Inhibit DFW bit is set on the Define Extent first CCW of a Write channel program, to inhibit DFW, and consequently the staging of NVS
- *Force*, the data set I/O is cached so that the indicators can be evaluated

The logic is the following:



- After open for the first 100 IOs and the first 100 Writes the IOs are *forced*
- The data set indicator is compared with Subsystem Threshold. If does not exceed, the data set IOs are going to be inhibited for the next 5000 IOs. If exceeds, the data set IOs are going to be normal for a certain amount of time, where the comparison is going to be done again.

So, if the ST indicator is going up, the exclusion from cache for may-cache data sets is gradual, no sudden and dramatic changes in the DASD subsystem performance are caused.

As a DCME by-product DFSM I/O statistics (I/O rates, I/O response time, I/O service time components, caching statistics for reads and writes) are collected in new SMF records in data set and storage class basis.

### **B.1.6.3 Never-cache candidates data sets**

There are some data sets that are not good candidates for DASD caching, such as:

- Data sets that are processed track-by-track, as the input to the 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 which have a very poor direct revisit pattern
- ASM paging data sets

---

## **B.2 Cache analogy**

For sequential access, it is important to say that, cache does not save data CI disks I/O operations. The cache only tries to match the speed of the disks and channels. Consequently, using the faster resource less.

To make the previous sentence crystal clear take a look in the Figure 29 on page 242 and read the following analogy. Out of the parenthesis is an human story, in the parenthesis the analogous one for data processing.

Once upon a time, there was in a valley (system), a city (application program), named Piracicaba. This city (application program) needs in average per day, one trillion (million) 3-atoms H<sub>2</sub>O molecules (300-bytes I/O records) to consume (process). The H<sub>2</sub>O molecules (I/O records) are taken (read) *sequently* from a Lake (ESS disk) located in the high mountain (ESS controller). There is a pipe (I/O subsystem) composed of many tubes (channel, SAP and CPU IOS processing) to do that.

However, the flow (disk I/O rate) from the lake (disk) usually was short for the city (application program) average demand (demand). However, because this demand fluctuates along the day (application program run) due to the human daily tasks (CPU busy status and application program logic), the civil engineers (ESS engineers) decided to build a dam (cache) to relieve the problem. It is key to note that the existence of the dam (cache) does not vary the amount of H<sub>2</sub>O molecules (I/O records) that the lake (disk) can provide per day. The advantage of the dam (cache) is that when the city (application program) demand decreases the lake (disks) fill in advance (look ahead) the dam (cache), without a shortage (cache miss) in the city (application program) later on.

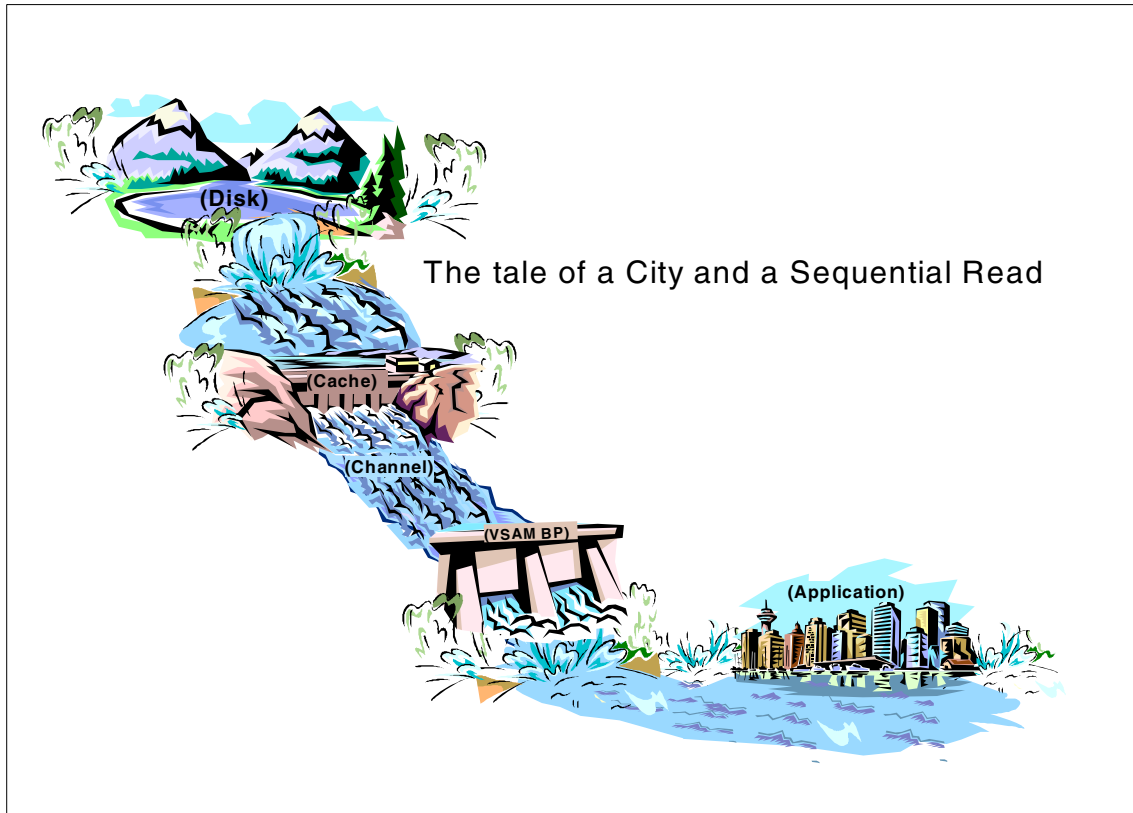


Figure 29. The tale

Now, here are a few questions to see if you are still awake:

- What happens if the instantaneous demand in the city (application program) is always bigger the instantaneous lake flow (disk I/O rate)?

Answer: The dam (cache) becomes empty and it is worthless. The total flow (I/O records per second) is determined by the slower stage, that is the lake (disk).

- What happens if it rains too much in the lake (the ESS controller is low utilized and there is no contention at all in the disks) and the lake flow (disk I/O rate) becomes higher than the city (application) demand?

Answer: The dam (cache) becomes full. Here, the fate of the city and the application diverge. The city suffers a flood. The application does not suffer a flood (S390 is not an Unix server in Denying of Service status) but the cache is worthless again. The total flow (I/O records per second) is determined by the slower stage, that is the application. We are loosing disk I/O capacity, by spinning wheels (in this case, loosing revolutions).

- Then, how come to build a dam (cache) can improve the efficiency and how can you improve this efficiency of the dam (cache)?

Answer: As we saw, if the city (application) or the lake (ESS disk) always has the higher flow, the dam is worthless, does not importing the size of the dam (cache). Repeating, the advantage of the dam (cache) is that when the city (application program) demand decreases the lake (disks) fill in advance (look ahead) the dam (cache), without a shortage (cache miss) in the city (application program) later on.

Making the dam (cache) bigger is a solution, to exploit better the fluctuations in the demand in both sides. However, steel and cement (circuits) have a price, do not exaggerate. In our story was created a project to increase the dam (cache) capacity. Guess what? The engineers disagree in the optimum size and they split in two teams:

- High mountain engineers (ESS engineers), which defend to increase the current status, that is the natural lake (non-volatile disks) plus the dam (cache)
- Low mountain engineers (VSAM engineers), which defend to build an artificial lake (volatile MVS/VSAM virtual storage) used as a second dam (VSAM buffer). Here the first dam (cache) and the second dam (VSAM buffer) are still connected by pipes (channels).

This was the final design and the final of the story. The second dam (VSAM buffer) is used has an extension of the first dam (cache) trying to balance overflows caused by variations in the demand or in the intake. It is important to note that the H<sub>2</sub>O molecule (I/O record) is moved and not

copied. Then, when the H<sub>2</sub>O molecule (I/O record) arrives in the city (application) there are no copies of it already in the first dam (cache), or in second dam (VSAM buffer).

Maybe a good name for this story is: The Tale of One City (almost by Dickens) or the Adventures of a Sequential Read (by us).

Random reads and random writes have a completely different stories.

### B.3 Share options analogy

Refer to Figure 30, to follow this explanation:

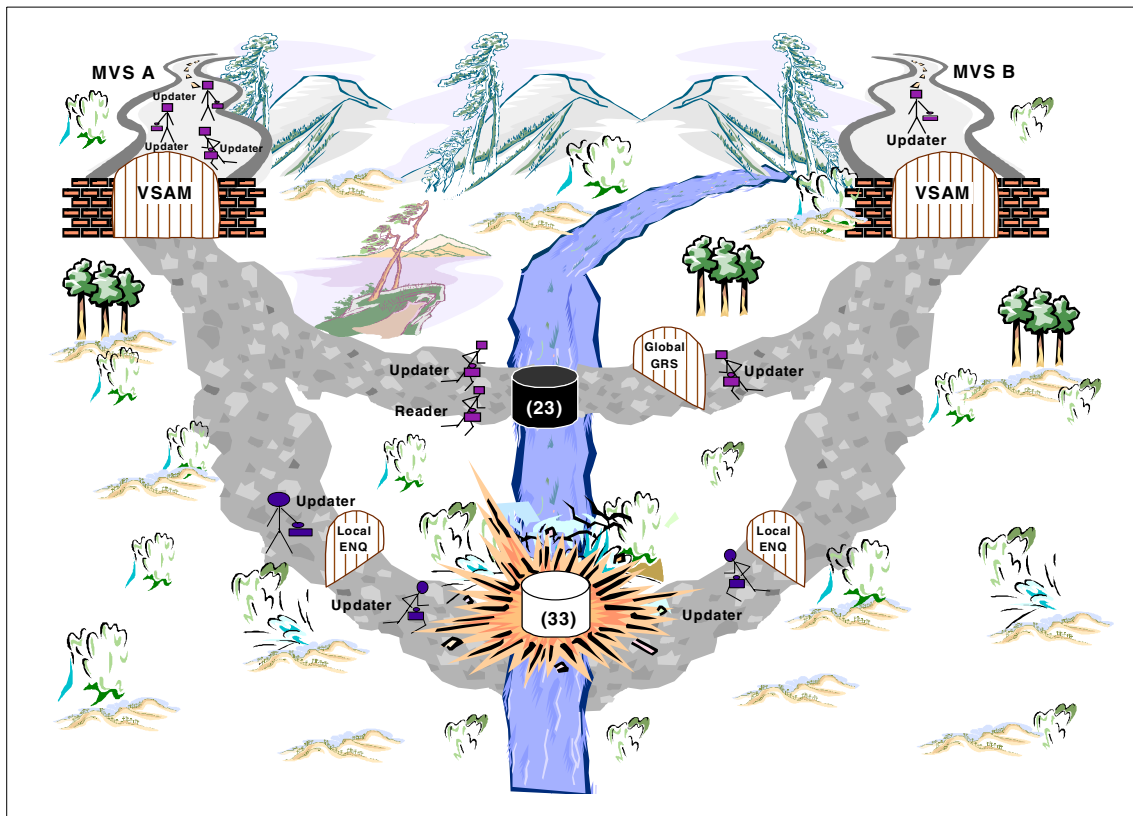


Figure 30. Sharing VSAM data sets

Once upon a time, there were two lands (MVS A and MVS B) separated by a river. All the culture accumulated by the people living there was stored in two shared VSAM data sets strategically located in the middle of the separating river. In each land, there were two sets of people, the

round-head and the square-head (pay attention that each set live on both sides of the river). Each head shaped people used their own data set (black data set for the square and white data set for the round head). The data sets were accessed by students, the readers (for read) and by the professors, the updaters (for writes).

The square-heads (from the both lands) care about write integrity and not about read integrity, so they choose shareoptions (2 3) for their data set and they use GRS adequately for their purpose.

The round-heads (from the both lands) cared about write and read integrity. They decided to implement that through GRS/ENQ mechanism only. The shareoptions of the round-head data set is (3 3).

The picture is showing what finally happened:

- In the square-head story, we may see, updaters being held by the VSAM gate in both sides of the river. This was caused by cross region 2 in shareoptions. Only one updater succeeds in passing the gate (in each side). All other open for output fail with a return code in ACB. However, the reader in MVS A was not blocked by VSAM gate (no read integrity).

To guarantee write integrity between updaters from the two lands a global GRS/ENQ is implemented guaranteeing that a second updater (from MVS B in the picture) which arrived last be held at the GRS gate.

- About the round-head story, there are not VSAM gates for the round-heads because cross region option 3. In both sides they implement a local ENQ gate to guarantee read and write integrity. That is, only one updater or several readers from each MVS are allowed to the round-head data set.

However, they did not read the GRS Primer book. The ENQ name is not made global to GRS and then two updaters (each from each land) are allowed to update concurrently the round-head data set, then blowing up the integrity.

---

## B.4 Symptoms (messages) from a broken data set

The most common messages associated with broken data sets events are:

- IDC3302I ACTION ERROR ON dsname

Explanation: 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

Explanation: 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.

System Action: 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, you should specify the replace option. If not check your Repro input.

- IDC3314I RECORD xxx OUT OF SEQUENCE

Explanation: 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 The first five bytes in hexadecimal format of the key of the record that is out of sequence.

System Action: If the output data set is a virtual storage access method (VSAM) data set, the system ends processing of the command after four errors.

Application Programmer Response: Rearrange the records to be written so that they are in ascending key sequence. The record can be written to the data set using skip sequential processing. Run the job again and the output data set will be opened for skip sequential processing (because data already exists in the data set) and records that were out of sequence will be written.

- IDC3351I \*\* VSAM {OPENICLOSEII/O} RETURN CODE IS return-code  
{RPLFDBWD=nnnnnnnn}

Explanation: 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.

rc The return code, as follows:

- For a CLOSE errors, we present only the return codes associated with broken data set situation:

- 128 Index search horizontal chain pointer loop encountered.
- 136 Not enough virtual storage was available in the program's address space 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 an OPEN errors, we present only the return codes associated with broken data set situation:
- 76 Attention message: The interrupt recognition flag (IRF) was detected for a data set opened for input processing
- This indicates that DELETE processing was interrupted. The structure of the data set is unpredictable; the access method services DIAGNOSE command may be used to check the data set for structural errors.
- 88 A previous extend error has 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 doesn't match the system time stamp in the volume record in the catalog; this indicates that extent information in the catalog record may not agree with the extents indicated in the volume's VTOC.
- 108 Attention message: the time stamps of a data component and an index component do not match; this indicates that either the data or the index has been updated separately from the other. 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, then data may 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 have two meanings:
- The data set was not properly closed.
- The data set is opened for output on another processor.
- Note: If you use the VERIFY command, this message can appear 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 One of the following errors occurred:

Not enough storage was available for work areas.

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 this data set has an incorrect physical record size.

160 The operands 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 being opened 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 will have to be restored.

192 An unusable data set was opened for output.

193 The interrupt recognition flag (IRF) was detected for a data set opened for output processing.

194 Direct access of a compressed data component is not allowed.

200 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 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 You attempted to store a record out of ascending key sequence in skip-sequential mode; record had a duplicate key; for skip-sequential processing, your GET, PUT, and POINT requests are not referencing records in ascending sequence; or, for skip-sequential retrieval, the key requested is lower than the previous key requested. For shared resources, buffer pool is full.

16 Record not found.

20 Record already held in exclusive control by another requester.

28 Data set cannot be extended because VSAM cannot allocate additional direct-access storage space. Either there is not enough space left to make the secondary allocation request, 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 could also be due to a data set trying to extend beyond 4GB on a system that does not support extended addressability.

32 An RBA specified that does not give the address of any data record in the data set.

40 Insufficient virtual storage in the user's address space to complete the request.

116 During initial data set loading (that is, when records are being stored in the data set the first time it's opened), 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.
- ThenumberofrecordsdoesnotmatchagroupRDFrecordcount.
- 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 the compression management services (CMS) during compression processing.

246 A severe error was detected by the compression management services (CMS) during decompression processing.

250 A valid dictionary token does not exist for the compressed data set. The data record cannot be decompressed.

254 I/O activity on the data set was not quiesced before the data set was closed.

- IDC3350I synad[SYNAD]message[from VSAM]

Explanation: 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

Explanation: An error occurred during EOV (end-of-volume) processing for a VSAM data set. In the message text:

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 (error information returned by another subsystem or component). This field appears only for certain return codes, and its format is shown with those codes to which it applies.

ccc 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 will be 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

---

## B.5 IDCAMS Examine messages

The most frequent error messages issued by Examine command are:

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



---

## Appendix C. Special notices

This publication is intended to provide users of VSAM data sets with the information required to understand, evaluate, and use VSAM properly. The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/390 or the DFSMS product. See the PUBLICATIONS section of the IBM Programming Announcement for OS/390 2.10 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.


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

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers

attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)® 

IBM ®

Hiperbatch

Hiperspace

MQSeries

MVS

RACF

S/390

VTAM

WebSphere

XT

Redbooks

Redbooks Logo 

IMS

Language Environment

Lotus

MQ

OS/390

Parallel Sysplex

QMF

RMF

SP

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

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

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.





---

## Appendix D. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

---

### D.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 259.

- *Enhanced Catalog Sharing and Management*, SG24-5594
- *Integrated Catalog Facility Backup and Recovery*, SG24-5644

---

### 4.7 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at [ibm.com/redbooks](http://ibm.com/redbooks) for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

---

### D.2 Other resources

These publications are also relevant as further information sources:

- *OS/390 MVS JCL Reference*, GC28-1757
- *OS/390 MVS JCL User's Guide*, GC28-1758
- *DFSMS/MVS Managing Catalogs*, SC26-4914
- *DFSMS/MVS Using Data Sets*, SC26-4922
- *DFSMS/MVS DFSMSdfp Storage Administration Reference*, SC26-4920

- *OS/390 Security Server (RACF) Command Language Reference*, SC28-1919
- *DFSMS/MVS Access Method Services for the Integrated Catalog Facility*, SC26-4906
- *DFSMS/MVS Macro Instructions for Data Sets*, SC26-4913

---

### D.3 Referenced Web sites

These Web sites are also relevant as further information sources:

- <http://knowledge.storage.ibm.com>    Storage information

---

## How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** [ibm.com/redbooks](http://ibm.com/redbooks)

Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

Send orders by e-mail including information from the IBM Redbooks fax order form to:

	<b>e-mail address</b>
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the "How to Order" section at this site: <a href="http://www.elink.ibm.link.ibm.com/pbl/pbl">http://www.elink.ibm.link.ibm.com/pbl/pbl</a>

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: <a href="http://www.elink.ibm.link.ibm.com/pbl/pbl">http://www.elink.ibm.link.ibm.com/pbl/pbl</a>

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: <a href="http://www.elink.ibm.link.ibm.com/pbl/pbl">http://www.elink.ibm.link.ibm.com/pbl/pbl</a>

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

### IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at <http://w3.itso.ibm.com/> and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at <http://w3.ibm.com/> for redbook, residency, and workshop announcements.

---

## IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

---

First name	Last name
------------	-----------

---

Company
---------

---

Address
---------

---

City	Postal code	Country
------	-------------	---------

---

Telephone number	Telefax number	VAT number
------------------	----------------	------------

☐ Invoice to customer number

---

☐ Credit card number

---

---

Credit card expiration date	Card issued to	Signature
-----------------------------	----------------	-----------

**We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.**

---

## Index

### A

access method services 27  
alternate index 10

### B

backup-while-open 130  
base cluster 62  
batch local shared resources 81  
BLDVRP macro 71  
broken index 154  
buffer pools 107  
buffers 58  
BUFND 61  
BUFNI 61  
BUFSP 61

### C

cache 113  
cache candidates 241  
cache modes 229  
cache usage attributes 236  
catalog 152, 158  
    recovery 158  
catalog management 2  
catalog search interface 138, 193  
catalogs 91  
CICS 195  
CLOSE macro 201  
cluster 8  
commands  
    AMS DEFINE PATH 11  
    BLDINDEX 11  
    DIAGNOSE 134, 147, 149  
    EXAMINE 134, 149  
    EXPORT 128  
    IDCAMS DEFINE 13  
    IMPORT 129  
    LISTCAT 166, 176  
    PRINT 174  
    REPRO 141  
    VERIFY 135, 143, 145  
component 6  
compression dictionaries 84  
control area 4  
    splits 4

control block update facility 192  
control interval 3, 48  
control interval definition field 3

### D

data buffers 65  
data component 6  
data compression 84  
data decompression 88  
data set recovery 127  
data striping 92  
    CA size 102  
    implementing 95  
    JCL 96  
    multi-layering 94  
    recommendations 103  
data-in-virtual 17  
DB2 195  
defer write requests 75  
DFSMSHsm 195  
DFSMSrmm 197  
direct access 14  
DITTO 29  
dynamic cache management enhanced 238

### E

end-of-volume macro 202  
entry sequenced data set 11  
extended addressability 19  
    macros 21  
extended format data set 18, 22

### F

free space 183  
FREESPACE 35

### G

generic compression 87  
generic key 13  
global shared resources 75

### H

HARBA 172  
HFS 29  
hierarchical file system 195

high used RBA 47  
hiperbatch 107  
hiperspace 74, 78  
history 24  
HURBA 171

## I

IDCAMS 28, 33  
IDCAMS LISTCAT 145  
IEFUSI exit 56  
II08859 APAR 128  
IMBED 36  
index component 6  
index options 52  
    REPLICATE 52  
index set 7  
indexed sequential access method 1  
initial load 140  
initial load mode 54, 67  
integrity 184

## J

Java 197

## K

keys 9  
knowledge database 128

## L

lab environment 223  
    cache concepts 226  
linear data set 16  
local shared resources 71  
logical record 2  
LSR buffering 73

## M

messages  
    IDC3302I 138  
media manager 200  
messages  
    ARC0909E 183  
    IDC11709I 142  
    IDC11712I 142  
    IDC11727I 142  
    IDC3009I 138, 159  
    IDC3308I 138, 140

IDC3314I 138  
IDC33351I 49, 55  
IDC3350 138  
IDC3350I 142  
IDC3351I 22, 138, 139, 141, 142, 144, 145,  
    147, 150, 153  
IEC070I 138  
IEC161I 140  
IOS000 138  
IOS000I 144

## N

non-shared resource 68  
non-shared resources 62, 64

## O

online transaction program 72  
OPEN macro 201  
OY40882 APAR 195

## P

parameters  
    ACB 59  
    AMP 78  
    buffer allocation 59  
    BUFFERSPACE 60  
    BUFND 61  
    BUFNI 61  
    BUFSP 60  
    FREESPACE 50  
    MACRF 60  
    performance 42  
    RECOVERY 67  
    SHAREOPTIONS 53  
    SMB services 77  
    SPEED 67  
    STRNO 69  
    SYSTEM 80  
partial release 46  
path 11  
Performance  
    hiperspace 74  
performance  
    BSLR 81  
    buffer allocation 62  
    buffer location 76  
    buffering options 58

- buffers 65
- BUFFERSPACE parameter 48
- catalog search interface 194
- CI size 48
- compression 90
- connect time 117
- constraint Relief 47
- data buffers 66
- data compression 84
- data striping 92, 102
- disconnect time 113
- global shared resources 75
- guaranteed space 43
- I/O response time 62
- index buffers 69
- index component buffers 67
- index I/O buffer 66
- IOS queue time 111
- KSDS 69
- lab environment 224
- management 39, 40
- NSR buffering 70
- optimizing CA size 45
- parameters 42
- pending time 112
- REGION 56, 57
- response time 119
- rule-of-thumb 41
- service level agreement 37
- SMB 77
- transaction 38
- physical record 2
- processing
  - direct 12
  - direct access 13
  - KSDS 13
  - relative record 15
  - sequential 12
  - sequential access 13

## R

- RACF 107
- record definition field 3
- record management 2
- recovery
  - scenarios 154
  - task abend 157
  - VVDS records 158

- recovery termination manager 61
- relative byte address 5
- relative record data set 14
- reorganization 183
- resource pool 58
- resource recovery management services 179
- resource recovery services 204
- RMF 103
- RMODE31 76

## S

- sample code 209
  - extract data from SMF 64 record 214
  - JRIO APIs 209
  - VSAM shared information 213
- sequence set 6
- SHAREOPTIONS 36, 65
- sharing
  - control block update facility 192
  - cross-region 188
  - cross-system 189
  - global resource sharing 192
  - intra-address space 185
  - options 188, 191
- SmartBatch 121
- SMF 91, 138, 174
- SMS managed 67
- space constraint relief 131
- spanned records 5
- sphere 10
- splits 4, 110, 181
  - FREESPACE parameter 50
- system managed buffering 77
- system-managed data sets 34

## T

- tailored compression 86
- transactional VSAM 202
  - batch applications 208
  - environment 203
  - sharing control data sets 205
  - SMSVSAM 205
  - system logger 206

## V

- variable relative record 16
- VSAM 24

- buffering 120
- catalog management 2
- cluster 8, 10
- component 6
- compression 89
- control interval 3
- data set organization 110
- data set recovery 127
- defining 33
- entry sequenced data set 11
- exploiters 195
- extended addressability 19
- extended format 18
- history 1
- initial load mode 54
- integrity 53
- keyed sequenced data set 13
- linear data set 16
- managing data sets 181
- non-shared resource 68
- non-shared resources 62
- organizations 25
- parameters 34
- performance 37
- performance management 103
- record management 2
- recovery 141, 143, 145, 147, 149, 152
- relative record data set 14
- sharing 150
- sharing data sets 183
- SMB 77
- structural damage 147
- VRRDS 16
- VTOC 92

## **W**

- wasted space 181



## IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at [ibm.com/redbooks](http://ibm.com/redbooks)
- Fax this form to: USA International Access Code + 1 845 432 8264
- Send your comments in an Internet note to [redbook@us.ibm.com](mailto:redbook@us.ibm.com)

<b>Document Number</b>	SG24-6105-00
<b>Redbook Title</b>	VSAM Demystified
<b>Review</b>	<div></div> <div></div> <div></div> <div></div> <div></div> <div></div>
<b>What other subjects would you like to see IBM Redbooks address?</b>	<div></div> <div></div> <div></div>
<b>Please rate your overall satisfaction:</b>	<input type="radio"/> Very Good <input type="radio"/> Good <input type="radio"/> Average <input type="radio"/> Poor
<b>Please identify yourself as belonging to one of the following groups:</b>	<input type="radio"/> Customer <input type="radio"/> Business Partner <input type="radio"/> Solution Developer <input type="radio"/> IBM, Lotus or Tivoli Employee <input type="radio"/> None of the above
<b>Your email address:</b> The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	<input type="checkbox"/> Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
<b>Questions about IBM's privacy policy?</b>	The following link explains how we protect your personal information. <a href="http://ibm.com/privacy/yourprivacy/">ibm.com/privacy/yourprivacy/</a>





# VSAM Demystified

(0.5" spine)

0.475" <-> 0.875"

250 <-> 459 pages







# VSAM Demystified



**Redbooks**

## **Understand VSAM architecture**

## **Manage VSAM data**

## **Improve VSAM performance**

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

This IBM Redbook will give you the information required to understand, evaluate, and use VSAM properly. It will clarify VSAM functions for application programmers who will be working with VSAM. The practical, straightforward approach should dispel much of the complexity sometimes associated with VSAM. Wherever possible an example is used to reinforce a description of a VSAM function.

This redbook is intended as a supplement to existing product manuals. It is intended to be used as an initial point of reference for VSAM functions. For example, parameters used in data set allocation to improve performance are described, and code examples provided, but the actual manual, *DFSMS/MVS Access Method Services for the Integrated Catalog Facility*, SC26-4906, must be consulted for complete syntax rules.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

### **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)

SG24-6105-00

ISBN 0738418110