IBM

MVS/Extended Architecture VSAM Administration Guide

Version 2 Release 4

GC26-4151-5

GC26-4151-5

MVS/Extended Architecture VSAM Administration Guide

Version 2 Release 4

Sixth Edition (September 1989)

This edition replaces and makes obsolete the previous edition, GC26-4151-4.

This edition applies to Version 2 Release 4 of MVS/Extended Architecture Data Facility Product, Program number 5665-XA2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" following the table of contents. Specific changes are indicated by a vertical bar to the left of the change. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

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 licensed program in this publication is not intended to state or imply that only IBM's program may be used. Any functionally equivalent program may be used instead.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

A Reader's Comment Form is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department J57, P. O. Box 49023, San Jose, California, U.S.A. 95161-9023. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks

The following names have been adopted by IBM for trademark use and are used throughout this publication:

MVS/SP™

MVS/XA™

Contents

Summary of Changes

Release 4, September 1989

Service Changes

All information about processing an index has been consolidated in Appendix D, "Processing the Index of a Key-Sequenced Data Set."

Minor technical and editorial changes have been made.

Release 3.0, June 1987

New Device Support

In Chapter 3, "Defining a VSAM Data Set," a new section—"Direct Access Device Characteristics" has been added, as well as Figure 15 on page 26. Chapter 7, "Processing Control Intervals" has also been updated to reflect all models of the 3380. The above changes were made to this manual to reflect basic support of the following types of devices:

DASD

- IBM 3380 Direct Access Storage Models AJ4, BJ4, AK4, and BK4
- IBM 3380 Direct Access Storage Direct Channel Attach Model CJ2

Storage Control

- IBM 3880 Storage Control Model 3 with 3380 AJ4/AK4 Attachment (feature 3005)
- IBM 3990 Storage Control Models 1 and 2

Cache Storage Control

- IBM 3880 Storage Control Model 23 with 3380 AJ4/AK4 Attachment (feature 3010)
- IBM 3990 Storage Control Model 3.

Enhancements

Local shared resource (LSR) support allows separate resource pools for the data and index portions of a VSAM data set. Chapter 10, "Sharing Resources among Data Sets" on page 127 reflects this change.

The alternate index is synchronized with the base data set following errors. Chapter 3, "Defining a VSAM Data Set" on page 19 describes this change.

VSAM control blocks and buffers may be obtained above 16 megabytes to provide virtual storage constraint relief. Appendix A, "Using 31-Bit Support" on page 147, other appendixes and chapters reflect this change.

Information describing the new linear data set has been added.

Customization Restructure

The contents of Chapter 11, "User-Written Exit Routines," and Appendix G, "Datestamp Routine," have been moved to *Data Facility Product: Customization*.

Service Changes

Information has been added to reflect technical service changes.

Release 2.0, June 1986

Enhancements

For VSAM data sets cataloged in integrated catalog facility catalogs as well as the integrated catalog facility catalogs themselves, the physical block sizes supported are the same as the data control interval sizes.

Service Changes

Documentation for the datestamp control module (IDATMSTP) is included in Appendix G, "Datestamp Routine."

Documentation for the VSAM trace facility is included in Appendix H, "VSAM Trace Facility."

Information has been added to reflect technical service changes.

Release 1.0 Update, December 1985

New Programming Support

A new command, EXAMINE, has been added to Access Method Services. In support of the EXAMINE service aid, "Chapter 12. Checking a VSAM Key-Sequenced Data Set Cluster for Structural Errors" describes, with examples, how EXAMINE is used to analyze a VSAM key-sequenced data set cluster for structural inconsistencies.

Erase-on-Scratch Security Enhancement

With the installation of RACF Version 1, Release 7, erase-on-scratch can be controlled by RACF options and data set profiles for integrated catalog facility cataloged VSAM data sets.

Export and Import by Control Interval

Access method services EXPORT command parameters provide a method of requesting exporting by control interval for ESDS base clusters.

Service Changes

Information has been added to reflect technical service changes.

Release 1.0, April 1985

Enhancements

Enhancements have been added to the JRNAD user exit that allow you to cancel a request for a control interval or control area split.

Version 2 Publications

The Preface includes order numbers for Version 2 publications.

RACF Support

Information to support Version 1 Release 7 of the Resource Access Control Facility (RACF) has been added.

Preface

About This Book

This book is intended to help you use access method services commands, VSAM macro instructions, and JCL to process VSAM data sets. This book contains information on the use of the virtual storage access method (VSAM). Unless specifically stated otherwise, the information in this book must not be used for programming purposes. However, this book also provides the following types of information, which are explicitly identified where they occur:

General-Use Programming Interface

General-use programming interfaces are provided to allow you to write programs that use the services of MVS/XA Data Facility Product.

End of General-Use Programming Interface

Product-Sensitive Programming Interface

Installation exits and other product-sensitive interfaces are provided to allow your installation to perform tasks such as product tailoring, monitoring, modification, or diagnosis. They are dependent on detailed design or implementation of the product. Such interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

End of Product-Sensitive Programming Interface

Throughout this book, the term DFDSS refers to either the IBM Data Facility Data Set Services product or an equivalent product. Likewise, the term RACF refers to either the IBM Resource Access Control Facility product or an equivalent product.

Required Product Knowledge

To use this publication effectively, you should be familiar with:

- Catalog administration
- Job control language
- Principles of operation

Required Publications

You should be familiar with the information presented in the following publications:

- *MVS/Extended Architecture VSAM Administration: Macro Instruction Reference*, GC26-4152, describes the macro instructions that are used with VSAM programs.
- *MVS/Extended Architecture Integrated Catalog Administration: Access Method Services Reference*, GC26-4135, or *MVS/Extended Architecture VSAM Catalog Administration: Access Method Services Reference*, GC26-4136, describes the access method services commands that are used with VSAM.
- *MVS/Extended Architecture Catalog Administration Guide*, GC26-4138, describes the administration of tasks for catalogs and how to use the access method services commands to manipulate catalogs, and the objects cataloged in them.
- *MVS/Extended Architecture Data Facility Product Version 2: Customization*, contains consolidated customization information for the DFP library.
- *MVS/Extended Architecture JCL User's Guide*, GC28-1351, and *MVS/Extended Architecture JCL Reference*, GC28-1352, describes the JCL parameters referred to in this publication and describes dynamic allocation.
- *MVS/Extended Architecture Message Library: System Messages*, Volumes 1 and 2, GC28-1376 and GC28-1377, provides a complete listing of the messages issued by VSAM.

Referenced Publications

Within the text, references are made to the publications listed in the table below:

Short Title	Publication Title	Order Number
Access Method Services Reference	*MVS/Extended Architecture Integrated Catalog Administration: Access Method Services Reference*	GC26-4135
	MVS/Extended Architecture VSAM Catalog Administration: Access Method Services Reference	GC26-4136
Catalog Administration Guide	*MVS/Extended Architecture Catalog Administration Guide*	GC26-4138
Checkpoint/ Restart User's Guide	*MVS/Extended Architecture Checkpoint/Restart User's Guide*	GC26-4139

Short Title	**Publication Title**	**Order Number**
Data Administration: Macro Instruction Reference	*MVS/Extended Architecture Data Administration: Macro Instruction Reference*	GC26-4141
Data Facility Product: Customization	*MVS/Extended Architecture Data Facility Product Version 2: Customization*	GC26-4267
Data Facility Product: Master Index	*MVS/Extended Architecture Data Facility Product Version 2: Master Index*	GC26-4146
Data Facility Product: Planning Guide	*MVS/Extended Architecture Data Facility Product Version 2: Planning Guide*	GC26-4147
DFDSS: User's Guide and Reference	*Data Facility Data Set Services: User's Guide and Reference*	SC26-4125
Global Resource Serialization	*MVS/Extended Architecture Planning: Global Resource Serialization*	GC28-1062
JCL User's Guide	*MVS/Extended Architecture JCL User's Guide*	GC28-1351
JCL Reference	*MVS/Extended Architecture JCL Reference*	GC28-1352
RACF General Information Manual	*Resource Access Control Facility General Information*	GC28-0722
System – Data Administration	*MVS/Extended Architecture System – Data Administration*	GC26-4149
Supervisor Services and Macro Instructions	*MVS/Extended Architecture Supervisor Services and Macro Instructions*	GC28-1154
System Macros and Facilities	*MVS/Extended Architecture System Programming Library: System Macros and Facilities* Volumes 1 and 2	GC28-1150 and GC28-1151
System Messages	*MVS/Extended Architecture Message Library: System Messages* Volumes 1 and 2	GC28-1376 and GC28-1377
System Modifications	*MVS/Extended Architecture System Programming Library: System Modifications*	GC28-1152

Short Title	Publication Title	Order Number
VSAM Administration: Macro Instruction Reference	*MVS/Extended Architecture VSAM Administration: Macro Instruction Reference*	GC26-4152

Notational Conventions

A uniform system of notation describes the format of VSAM macro instructions. This notation is not part of the language; it merely provides a basis for describing the structure of the macros.

The macro format illustrations in this book use the following conventions:

- Brackets [ ] indicate optional parameters.
- Braces { } indicate a choice of entry; unless a default is indicated, you must choose one of the entries.
- Items separated by a vertical bar (|) represent alternative items. No more than one of the items may be selected.
- An ellipsis (...) indicates that multiple entries of the type immediately preceding the ellipsis are allowed.
- Other punctuation (parentheses, commas, and so forth) must be entered as shown.
- **BOLDFACED** type indicates the exact characters to be entered. Such items must be entered exactly as illustrated (in uppercase, except in TSO).
- *Italic* type specifies fields to be supplied by the user.
- <u>**BOLDFACED UNDERSCORED**</u> type indicates a default option. If the parameter is omitted, the underscored boldfaced value is assumed.
- A ' ' in the macro format indicates that a blank (an empty space) must be present before the next parameter.

Chapter 1. Introduction

The virtual storage access method (VSAM) is an access method used to organize data and maintain information about that data in a catalog.

There are two major parts of VSAM:

- **Catalog management.** VSAM maintains extensive information about data sets and direct access storage space in a catalog. The catalog can be either an integrated catalog facility catalog or a VSAM catalog. The catalog's collection of information about a particular data set defines that data set. Every VSAM data set must be defined in a catalog. You cannot, for example, load records into a VSAM data set until its definition has been established. For information about catalog management, see *Catalog Administration Guide*.

- **Record management.** VSAM can be used to organize records into four types of data sets. These data sets are called key-sequenced data sets, entry-sequenced data sets, linear data sets and relative record data sets. The primary difference among these types of data sets is the method of storing and accessing records.

VSAM programming is performed using access method services commands and VSAM macro instructions.

- **Access method services.** In VSAM, you define data sets and establish catalogs using a multifunction services program called access method services. Access method services is directed by a set of control statements called commands.

- **VSAM macro instructions.** Two types of VSAM macros are used to process VSAM data sets.

  - *Control block macros* generate control blocks of information needed by VSAM to process the data set.

  - *Request macros* are used to retrieve, update, delete, or insert logical records.

You can use either 24-bit or 31-bit addresses for VSAM programs. If you use 31-bit support, see Appendix A, "Using 31-Bit Support" on page 147 for procedures and restrictions.

VSAM allows you to do the following:

- Share a data set among different operating systems, different jobs in a single operating system, or different subtasks in an address space.

- Share buffers and control blocks among VSAM data sets.

- Provide exit routines to analyze logical errors, and physical errors, to perform end-of-data processing, to keep a record of transactions against a data set, to perform user-security verification, and for special user processing.

- Back up and recover data sets.

- Provide measures to maintain data security and integrity.

Chapter 2. VSAM Data Set Organization

Data Storage

When you request information from or supply information to VSAM data management, the unit of information you request or supply is a *logical record.* Logical records of VSAM data sets are stored differently from logical records in non-VSAM data sets. VSAM uses *control intervals* to contain records. Whenever a record is retrieved from direct access storage, the entire control interval containing the record is read into a VSAM I/O buffer in virtual storage. From the VSAM buffer, the desired record is transferred to a user-defined buffer or work area. Figure 1 shows how a logical record is retrieved from direct access storage.

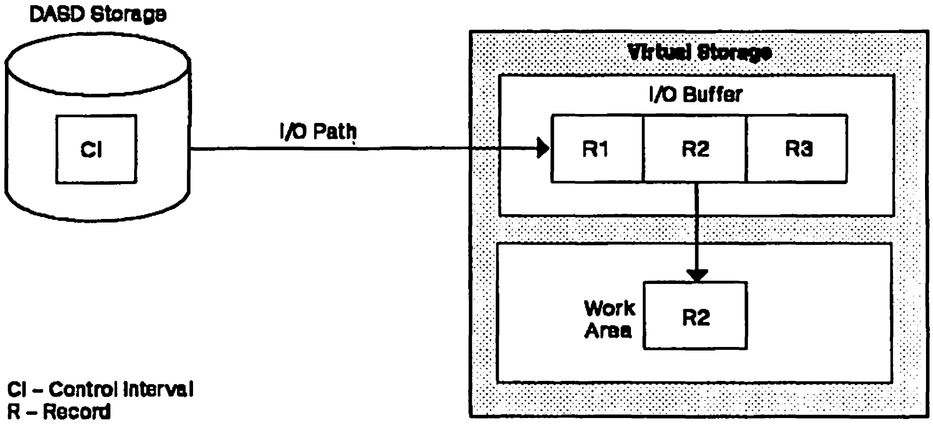


Figure 1. VSAM Logical Record Retrieval

Data Set Size

A VSAM data set can be extended beyond its original size to include up to 123 extents, or to a maximum size of 2^{32} (4,294,967,296) bytes. For additional information on space allocation for VSAM data sets, see "Allocating Space for a Data Set" on page 23.

Control Intervals

A *control interval* is a continuous area of direct access storage that VSAM uses to store data records and control information that describes the records. A control interval is the unit of information that VSAM transfers between virtual storage and disk storage.

The size of control intervals can vary from one VSAM data set to another, but all the control intervals within the data portion of a particular data set must be the same length. You can let VSAM select the size of a control interval for a data set, or you can request a particular control interval size using the access method services DEFINE command. For information on selecting the best control interval size, see "Optimizing Control Interval Size" on page 65.

A control interval consists of the following:

- Logical records
- Free space
- Control information fields

Note: In a linear data set all of the control interval bytes are data bytes. There is no imbedded control information.

Control Information Fields

Figure 2 shows that control information consists of two types of fields: one control interval definition field (CIDF), and one or more record definition fields (RDFs). CIDFs are 4 bytes long, and contain information about the control interval, including the amount and location of free space. RDFs are 3 bytes long, and describe the length of records and how many adjacent records are of the same length.

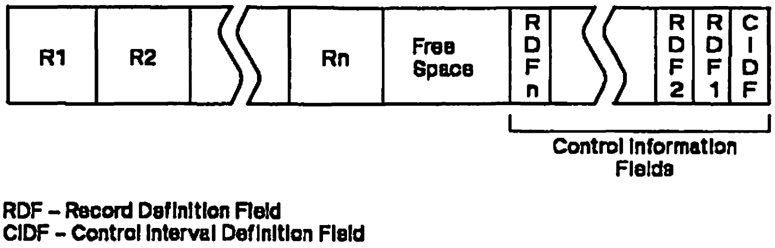

R1
R2
Rn
Free Space
RDFn
RDF2
RDF1
CIDF
Control Information Fields
RDF - Record Definition Field
CIDF - Control Interval Definition Field

Figure 2. Control Interval Format

There is no parameter in VSAM to define the data set records as fixed length or variable length (like RECFM for non-VSAM data sets). VSAM takes into consideration the length of adjacent records in a control interval when writing a record. If two or more adjacent records have the same length, only two RDFs are used for this group. One RDF gives the length of each record, and the other gives the number of consecutive records of the same length.

Figure 3 on page 5 shows RDFs for records of the same and different lengths.

Control Interval 1

Control interval size = 512 bytes
Record length = 160-byte records
Record definition fields—Only 2 RDFs are needed because all records are the same length.

	R1	R2	R3	FS	RDF	RDF	CIDF
Record Length	160	160	160	22	3	3	4

Control Interval 2

Control interval size = 512 bytes
Record length—All records have different lengths
Record definition fields—One RDF is required for each logical record (RDF 1 for record 1, RDF 2 for record 2, and so forth).

R1	R2	R3	R4	FS	RDF	RDF	RDF	RDF	CIDF
130	70	110	140	46	3	3	3	3	4

Control Interval 3

Control interval size = 512 bytes
Record length—Records 1 through 3 are 80-byte records.
Records 4 and 5 have different lengths.
Record definition fields—Two RDFs are used for records 1 through 3.
Records 4 and 5 each have their own RDF.

R1	R2	R3	R4	R5	FS	RDF	RDF	RDF	RDF	CIDF
80	80	80	100	83	63	3	3	3	3	4

FS – Free Space

Figure 3. Relation between Records and RDFs

Control Areas

The control intervals in a VSAM data set are grouped together into fixed-length contiguous areas of direct access storage called *control areas.* A VSAM data set is actually composed of one or more control areas. The number of control intervals in a control area is fixed by VSAM.

The maximum size of a control area is one cylinder, and the minimum size is one track of DASD storage. When you specify the amount of space to be allocated to a data set, you implicitly define the control area size. For information on finding the best control area size, see "Optimizing Control Area Size" on page 69.

Spanned Records

Sometimes a record is larger than the optimal control interval size for a particular data set. In VSAM, you do not need to break apart or reformat such records, because you can specify spanned records when defining a data set. The SPANNED parameter allows a record to extend across or span control interval boundaries.

Spanned records may reduce the amount of DASD space required for a data set when data records vary significantly in length. Figure 4 and Figure 5 demonstrate how spanned records can be employed for more efficient use of space.

In Figure 4, each control interval is 10240 bytes long. Control interval 1 contains a 2000-byte record. Control interval 2 contains a 10000-byte record. Control interval 3 contains a 2000-byte record. 30720 bytes of storage are used to contain these 3 records.

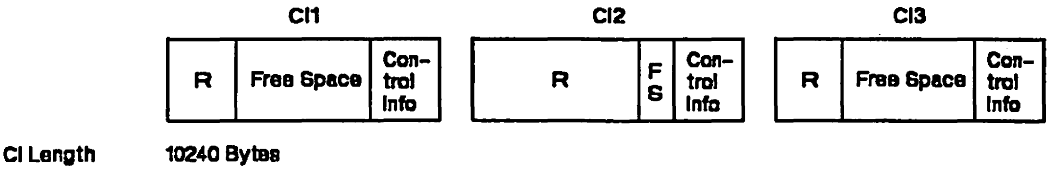


Figure 4. Data Set with Nonspanned Records

Figure 5 shows a data set with the same space requirements as that in Figure 4, but one that allows spanned records.

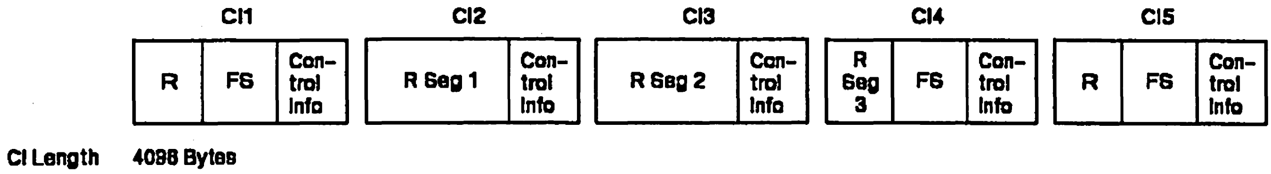


Figure 5. Data Set with Spanned Records

The control interval size is reduced to 4096 bytes. When the record to be stored is larger than the control interval size, the record is spanned between control intervals. In the figure, control interval 1 contains a 2000-byte record. Control intervals 2, 3, and 4 together contain one 10000-byte record. Control interval 3 contains a 2000-byte record. By changing control interval size and allowing spanned records, the three records can be stored in 20480 bytes, reducing the amount of storage needed by 10240 bytes.

You should remember:

- A spanned record always begins on a control interval boundary and fills more than one control interval within a single control area.
- For key-sequenced data sets, the entire key field of a spanned record must be in the first control interval.
- The control interval that contains the last segment of a spanned record may also contain unused space. This unused space can be used only to extend the spanned record; it cannot contain all or part of any other record.
- Spanned records can only be used with key-sequenced data sets and entry-sequenced data sets. (For information on key-sequenced data sets and entry-sequenced data sets, see "Choosing a Data Set Type.")
- To span control intervals, you must specify the SPANNED parameter when you define your data set. VSAM decides whether a record is spanned or nonspanned, depending on the control interval length and the record length.

Choosing a Data Set Type

VSAM supports four different data set types: key-sequenced, entry-sequenced, linear and relative record. Before you choose a data set type, consider the following questions:

- Will you need to access the records in sequence, randomly, or both ways?
- Are all the records the same length?
- Will the record length change?
- How often will you need to move records?
- How often will you need to delete records?
- Do you want spanned records?
- Do you want to keep the data in order by the contents of the record?
- Do you want to access the data by an alternate index?
- Do you want to use access method services utilities with an IBM DATABASE 2 (DB2) cluster?

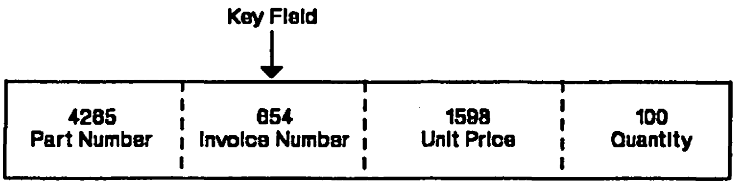
Figure 6 on page 8 is a quick summary of what each data set type offers.

Figure 6. Comparison of Key-Sequenced, Entry-Sequenced, Relative Record and Linear Data Sets

Key-Sequenced Data Set	**Entry-Sequenced Data Set**	**Relative Record Data Set**	**Linear Data Set**
Records are in collating sequence by key field	Records are in order in which they are entered	Records are in relative record number order	No processing at record level
Direct access by key or by RBA	Direct access by RBA	Direct access by relative record number	Access with Data-In-Virtual (DIV)
Alternate indexes allowed	Alternate indexes allowed	No alternate indexes allowed	No alternate indexes allowed
A record's RBA can change	A record's RBA cannot change	A record's relative record number cannot change	No processing at record level
Free space is used for inserting and lengthening records	Space at the end of the data set is used for adding records	Empty slots in the data set are used for adding records	No processing at record level
Space given up by a deleted or shortened record becomes free space	A record cannot be deleted, but you can reuse its space for a record of the same length	A slot given up by a deleted record can be reused	No processing at record level
Spanned records allowed	Spanned records allowed	No spanned records	No spanned records

Key-Sequenced Data Sets

In a key-sequenced data set, logical records are placed in the data set in ascending collating sequence by a field, called the *key*. As shown in Figure 7 on page 9, the key contains a unique value, such as an employee number or invoice number, which 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.



The key must be:

- Unique
- In the same position in each record
- In the first segment of a spanned record

Figure 7. Record of a Key-Sequenced Data Set

When a new record is added to the data set, it is inserted in its collating sequence by key, as shown in Figure 8.

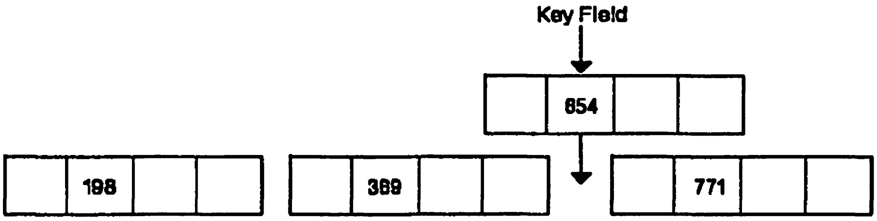


Figure 8. Inserting Records in a Key-Sequenced Data Set

Free Space

When a key-sequenced data set is created, unused space can be scattered throughout the data set to allow records to be inserted or lengthened. This space is called *free space*. When a new record is added to a control interval or an existing record is lengthened, subsequent records are moved into the following free space to make room for the new or lengthened record. Conversely, when a record is deleted or shortened, the space given up is reclaimed as free space for later use. When you define your data set, you can use the FREESPACE parameter to specify what percentage of each control interval is to be set aside as free space when the data set is initially loaded.

Within each control area, you can reserve free space in the form of free control intervals. If you have free space in your control area, it is easier to avoid splitting your control area when you want to insert additional records or lengthen existing records. When you define your data set, you can specify what percentage of the control area is to be set aside as free space, using the FREESPACE parameter.

For information on specifying the optimal amount of control interval and control area free space, see "Optimizing Free Space Distribution" on page 70.

Using Control Interval Free Space

Figure 9 shows how control interval free space is used to insert and delete a logical record in a key-sequenced data set.

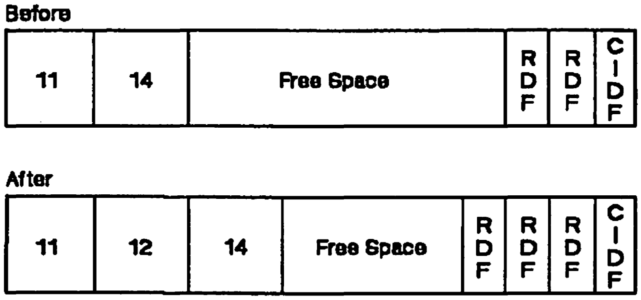


Figure 9. Inserting a Logical Record in a Control Interval

Two logical records are stored in the first control interval shown in Figure 9. Each logical record has a key (11 and 14). The second control interval shows what happens when you insert a logical record with a key of 12.

1. Logical record 12 is inserted in its correct collating sequence in the control interval.
2. The control interval definition field (CIDF) is updated to show the reduction of available free space.
3. A corresponding RDF is inserted in the appropriate location to describe the length of the new record.

When a record is deleted, the procedure is reversed, and the space occupied by the logical record and corresponding RDF is reclaimed as free space.

Prime Index

A key-sequenced data set always has an index that relates key values to the relative locations of the the logical records in a data set. This index is called the *prime index*. The prime index, or simply index, has two uses:

- To locate the collating position when inserting records
- To locate records for retrieval

When initially loading a data set, records must be presented to VSAM in key sequence. The index for a key-sequenced data set is built automatically by VSAM as the data set is loaded with records.

When a data control interval is completely loaded with logical records, free space, and control information, VSAM makes an entry in the index. The entry consists of the *highest possible key* in the data control interval and a pointer to the beginning of that control interval.

Key Compression

The key in an index entry is stored by VSAM in a compressed form. This key compression eliminates from the front and back of a key those characters that aren't necessary to distinguish it from the adjacent keys. Compression helps achieve a smaller index by reducing the size of keys in index entries.

Control Interval Splits

When a data set is first loaded, the key sequence of data records and their physical order are the same. However, when data records are inserted, *control interval splits* can occur, causing the data control intervals to have a physical order that differs from the key sequence.

Entry-Sequenced Data Sets

An entry-sequenced data set is comparable to a sequential access method (SAM) data set. It contains records that may be either spanned or nonspanned. As Figure 10 shows, records are sequenced by the order of their entry in the data set, rather than by a key field in the logical record.

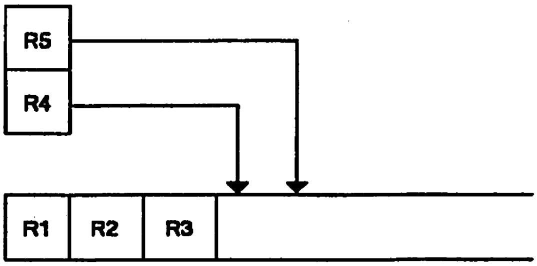


Figure 10. Entry-Sequenced Data Set

Records are only added at the end of the data set. Existing records cannot be erased. If you want to delete a record, you are responsible for flagging that record as inactive. As far as VSAM is concerned, the record is not deleted. Records can be updated, but they cannot be lengthened. To change the length of a record in an entry-sequenced data set, you must store it either at the end of the data set (as a new record) or in the place of a record of the same length that you have flagged as inactive, or that is no longer required.

Because entries are always added to the end of an entry-sequenced data set, the reserved free space concept does not apply.

VSAM does not maintain a prime index for an entry-sequenced data set. To retrieve records directly (not in sequence) from an entry-sequenced data set, you must keep track of the record's relative byte address (RBA) and associate this RBA with the contents of the record.

The RBA of a logical record is the offset of this logical record from the beginning of the data set. The first record in a data set has an RBA of 0; 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. Figure 11 on page 12 illustrates the record lengths and corresponding RBAs for the data set shown in Figure 10.

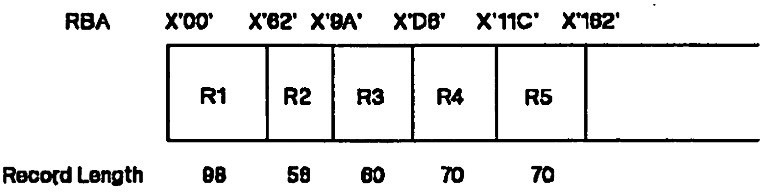


Figure 11. RBAs of an Entry-Sequenced Data Set

When a record is loaded or added, VSAM indicates its RBA. You can build an *alternate index* to keep track of these RBAs. (See "Creating an Alternate Index" on page 32.)

Linear Data Set

A linear data set (LDS) is a VSAM data set with a control interval size of 4096 bytes and a block size of 4096 bytes. An LDS does not have imbedded control information. All LDS bytes are data bytes. Only integrated catalog facility catalogs can support an LDS.

An LDS is processed as an entry-sequenced data set, with certain restrictions:

- Because an LDS does not contain any definition fields (CIDFs and RDFs), it cannot be processed as if it contained individual records. An LDS is processed using the DIV macro. For information about the DIV macro, see *Supervisor Services and Macro Instructions.*
- An LDS cannot have the spanned attribute.
- An LDS cannot have an alternate index.
- Other processing restrictions are covered in Chapter 4, "Processing a Data Set" on page 39.

Relative Record Data Sets

A relative record data set consists of a number of fixed-length slots. Each slot has a unique relative record number, and the slots are sequenced by ascending relative record number. Each 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; its relative record number cannot change. There is no prime index for a relative record data set.

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 relative record data set. The record definition field (RDF) indicates whether the slot is occupied or empty. Free space is not provided in a relative record data set because the entire data set is divided into fixed-length slots.

In a relative record data set, each control interval contains the same number of slots. The number of slots is determined by the control interval size and the record length. Figure 12 on page 13 shows the structure of a relative record data set after adding a few records. Each slot has a relative record number and an RDF.

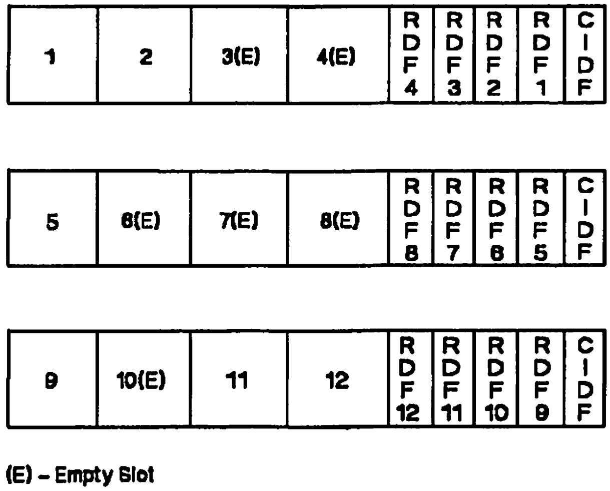


Figure 12. Relative Record Data Set

Types of Data Set Access

VSAM allows both sequential and direct access for all types of VSAM data sets—key-sequenced, entry-sequenced, linear, and relative record. Skip-sequential access is possible with key-sequenced data sets and relative record data sets. For a key-sequenced data set, the primary form of access is keyed access, using the primary key as a search argument. Another way of accessing a key-sequenced data set is addressed access, using the RBA of a logical record as a search argument. If you use addressed access to process key-sequenced data, you should be aware that RBAs may change when a control interval split occurs or when records are added, deleted, or changed in size. Therefore, access by address is not suggested for normal use.

An entry-sequenced data set without an alternate index can be accessed directly by address (using the RBA determined for a record when it was stored in the data set), or accessed sequentially.

A relative record data set can be accessed either directly, by providing the relative record number, or sequentially.

A linear data set can be accessed by control interval access. The other three types of data sets can also be accessed by control interval access, but this is used only for very specific applications. Control interval access is described in Chapter 7, "Processing Control Intervals" on page 87.

Accessing Records in a Key-Sequenced Data Set

The most effective way to access records of a key-sequenced data set is by key, using the associated prime index. The three methods of keyed access that can be specified for a key-sequenced data set are:

- Sequential access
- Direct access
- Skip-sequential access

Sequential Access

Sequential access is used to load a key-sequenced data set and to retrieve, update, add and delete records in an existing data set. When you specify sequential as the mode of access, 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 record in sequence.

Sequential processing can be started anywhere in the data set. Positioning is necessary if your starting point is within the data set. Positioning can be done in two ways:

- Using the POINT macro
- Issuing a direct request, then changing the request parameter list with the MODCB macro from "direct" to "sequential."

Sequential access allows you to avoid searching the index more than once, thereby taking less time than direct access. (Direct access processing searches the index from top to bottom for each record.)

Direct Access

Direct access is used to retrieve, update, add and delete records. When direct is specified as the mode of access, the prime index is used to directly access selected records by key value; the index is searched from top to bottom for each record. You need to supply a key value for each record to be processed.

For retrieval processing, you can either supply the full key, or a generic key. The generic key is the high-order portion of the full key. For example, you might want to retireve all records whose keys begin with the generic key AB, regardless of the full key value.

Direct access allows you to avoid retrieving the entire data set sequentially in order to process a relatively small percentage of the total number of records.

Skip-Sequential Access

Skip-sequential access is used to retrieve, update, add and delete records. When skip sequential is specified as the mode of access, 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 prime index to be searched from the top to the bottom level for each record

Accessing Records in an Entry-Sequenced Data Set

Entry-sequenced data sets are accessed by address, either sequentially or directly. When addressed sequential processing is used to process records in ascending relative byte address (RBA) sequence, VSAM automatically retrieves records in stored sequence.

To access a record directly from an entry-sequenced data set, you must supply the RBA for the record as a search argument. For information on how to obtain the RBA, refer to "Entry-Sequenced Data Sets" on page 11.

Skip-sequential processing is not supported for entry-sequenced data sets.

Accessing Records in a Linear Data Set

Because linear data sets do not contain control information, they cannot be accessed as if they contained individual records. You can access a linear data set with the DIV macro. For information on how to use Data-In-Virtual (DIV), see *Supervisor Services and Macro Instructions*.

Accessing Records in a Relative Record Data Set

For a relative record data set, keyed-sequential, keyed-skip-sequential, and keyed-direct processing are supported. The relative record number is always used as a search argument.

Sequential processing of a relative record data set is the same as sequential processing of an entry-sequenced data set. Empty slots are automatically skipped by VSAM.

Skip-sequential processing is treated like direct requests, except that VSAM maintains a pointer to the record it just retrieved. When retrieving subsequent records, the search begins from the pointer, rather than from the beginning of the data set. Records must be retrieved in ascending sequence.

A relative record data set can be processed directly by supplying the relative record number as a key. VSAM converts the relative record number to an RBA and determines the control interval containing the requested record. If a record in a slot flagged as empty is requested, a "no-record-found" condition is returned. You cannot use an RBA value to request a record in a relative record data set.

Accessing Records through an Alternate Index

An *alternate index* provides a way to access records by more than one key field. This eliminates the need to store multiple copies of the same information for different applications. Unlike primary keys, which must be unique, the key of an alternate index may refer to more than one record in the base cluster. An alternate key value that points to more than one record is *non-unique*; if the alternate key points to only one record, it is *unique*.

You use access method services to define and build one or more alternate indexes over a key-sequenced or an entry-sequenced data set, which is referred to as the *base cluster*. In terms of access, an alternate index performs the same function as the prime index of a key-sequenced data set. Alternate indexes are *not* supported for linear data sets, relative record data sets or reusable data sets (data sets defined with the REUSE attribute). For information on defining and building alternate indexes, see "Creating an Alternate Index" on page 32.

In structure, the alternate index is a key-sequenced data set. It consists of an index component and a data component. 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.

Each record in the data component of an alternate index is of variable length and contains header information, the alternate key, and at least one pointer to a base data record.

Header information is fixed length and indicates:

- Whether the alternate index data record contains prime keys or RBA pointers
- Whether the alternate index data record contains unique or non-unique keys
- The length of each pointer
- The length of the alternate key
- The number of pointers

Figure 13 illustrates the structure of an alternate index with non-unique keys connected to a key-sequenced data set. The salesman's name is the alternate key in this example. The customer number is the prime key.

Note: The maximum number of pointers associated with the alternate index data record cannot exceed 32767.

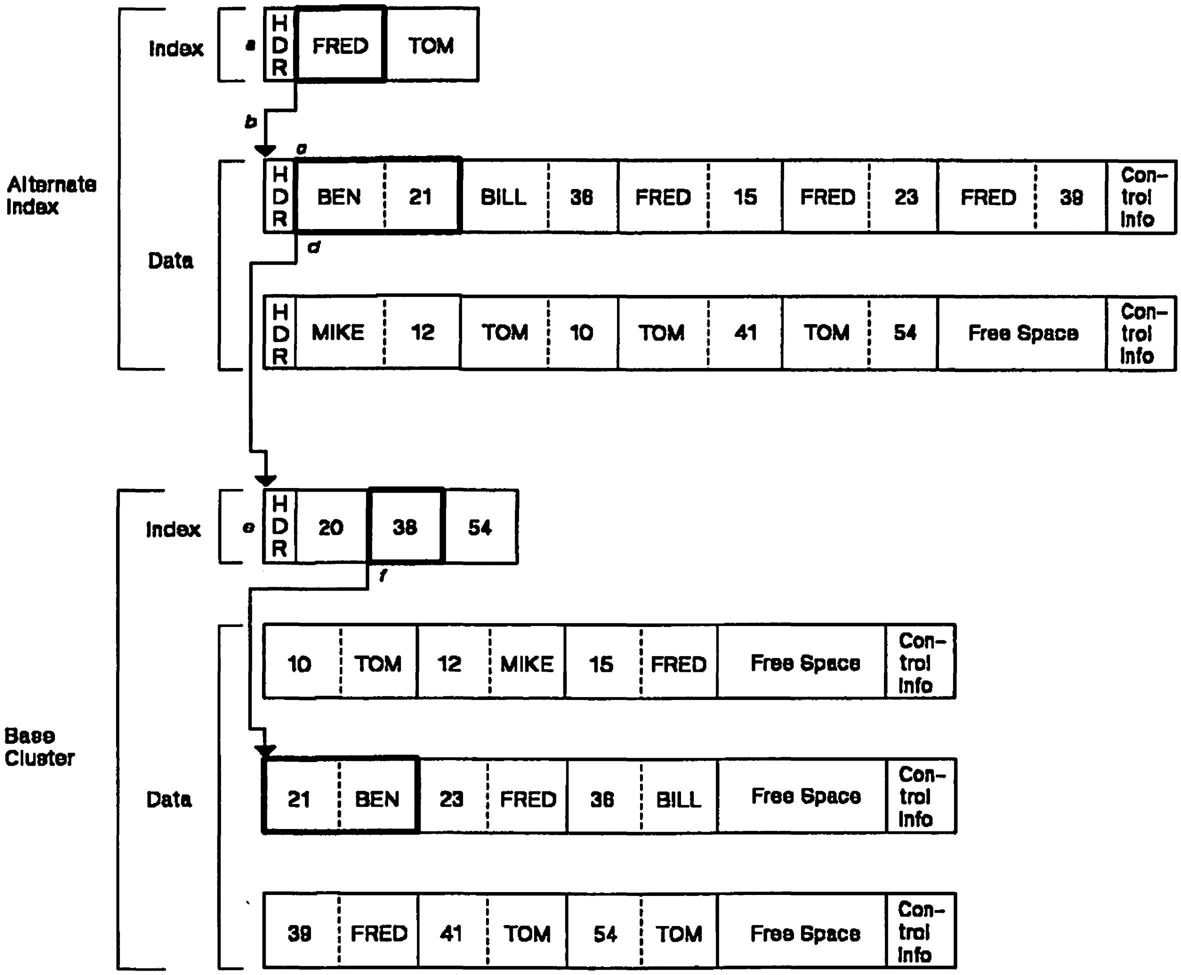


Figure 13. Alternate Index Structure for a Key-Sequenced Data Set

If you ask to access records with the alternate key of BEN, VSAM does the following: ***

a. VSAM scans the index component of the alternate index, looking for a value greater than or equal to BEN.

b. The entry FRED points VSAM to a data control interval in the alternate index.

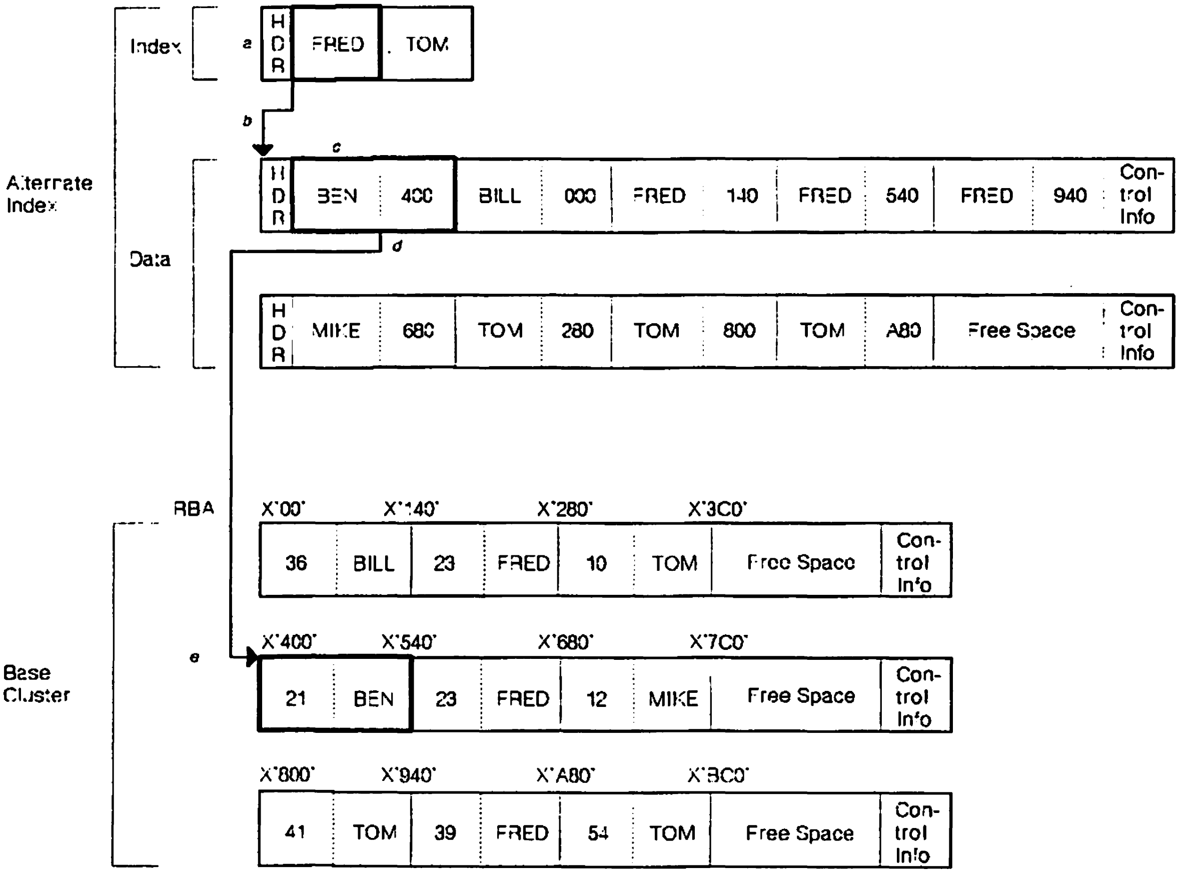
c. VSAM scans the alternate index data control interval looking for an entry that matches the search argument, BEN.

d. When located, the entry BEN has an associated key, 21. This key points VSAM to the index component of the base cluster.

e. VSAM scans the index component for an entry greater than or equal to the search argument, 21.

f. The index entry, 38, points VSAM to a data control interval in the base cluster. The record with a key of 21 is passed to the application program.

Figure 14 illustrates the structure of an alternate index connected to an entry-sequenced data set. The salesman's name is the alternate key in this example.



Note: RBAs are always written as fullword binary integers.

Figure 14. Alternate Index Structure for an Entry-Sequenced Data Set

If you ask to access records with the alternate key of BEN, VSAM does the following:

a. VSAM scans the index component of the alternate index, looking for a value greater than or equal to BEN.

b. The entry FRED points VSAM to a data control interval in the alternate index.

c. VSAM scans the alternate index data control interval looking for an entry that matches the search argument, BEN.

d. When located, the entry BEN has an associated pointer, 400, which points to an RBA in the base cluster.

e. VSAM retrieves the record with an RBA of X'400' from the base cluster.

When building an alternate index, the alternate key can be any field in the base data set's records having a fixed length and a fixed position in each record. The alternate key field must be in the first segment of a spanned record. Keys in the data component of an alternate index are not compressed; the entire key is represented in the alternate index data record.

A search for a given alternate key reads all the base cluster records containing this alternate key. For example, Figure 13 and Figure 14 show that in some cases one salesman has several customers. For the key-sequenced data set, several prime key pointers (customer numbers) are in the alternate index data record. There is one for each occurrence of the alternate key (salesman's name) in the base data set. For the entry-sequenced data set, several RBA pointers are in the alternate index data record. There is one for each occurrence of the alternate key (salesman's name) in the base data set. The pointers are ordered by arrival time. When multiple pointers are associated with a given alternate key value, the alternate key is said to be *non-unique*; if only one pointer is associated with the alternate key, it is *unique*.

Note: The maximum number of pointers associated with the alternate index data record cannot exceed 32767.

Alternate Index Paths

Before accessing a base cluster by way of an alternate index, a path must be defined. A path provides a way to gain access to the base data through a specific alternate index. You define a path with the access method services command DEFINE PATH. You must name the path and may also give it a password. The path name refers to the base cluster/alternate index pair. When you access the data set through the path, you must specify the path name as well as the DSNAME parameter in the JCL. For information on how to define a path, see "Defining a Path" on page 35.

Chapter 3. Defining a VSAM Data Set

VSAM data sets are defined using access method services commands. The following is a summary of how to define a VSAM data set.

1. Define a catalog using access method services commands, unless a catalog you can use already exists. The procedure for defining a catalog is described in *Catalog Administration Guide*.
2. Define a VSAM data set in the catalog, using the access method services DEFINE CLUSTER command. A VSAM data set does not exist until it is defined in a catalog.
3. Load the data set with data by using the access method services REPRO command, or by writing your own program to load the data set.
4. Optionally, create any alternate indexes, and relate them to the base cluster. Use the access method services DEFINE ALTERNATEINDEX, DEFINE PATH, and BLDINDEX commands to do this.

After any of these steps, you can use the access method services commands LISTCAT and PRINT to verify what has been defined, loaded, and processed. This is useful for identifying and correcting problems.

Appendix C, "Examples of Defining and Manipulating Data Sets" on page 157, provides a set of examples illustrating various aspects of creating VSAM data sets.

Cluster Concept

For a key-sequenced data set, a *cluster* is the combination of the data component and the index component. The cluster provides a way to treat the index and data as a single component with its own name. This allows you to print or dump the data component and the index component of a key-sequenced data set individually. You can give each component a name, and process the data portion separately from the index, or vice versa.

The concept of a cluster is carried over to entry-sequenced, linear and relative record data sets. These are considered to be clusters without index components. To be consistent, they are given cluster names, which are normally used when processing the data set.

Defining a Data Set

VSAM data sets are defined with the DEFINE CLUSTER command. When a cluster is defined, VSAM uses the following catalog entries to describe the cluster:

- A cluster entry describes the cluster as a single component.
- A data entry describes the cluster's data component.
- For a key-sequenced data set, an index entry describes the cluster's index component.

All of the cluster's attributes are recorded in the catalog. The information that is stored in the catalog provides the details needed to manage the data set and to access the VSAM cluster or the individual components.

Attributes of the components can be specified separately from attributes of the cluster.

- If attributes are specified for the cluster and not the components, the attributes of the cluster (except for password and USVR security attributes) apply to the components.
- If an attribute that is applicable to the data or index component is specified for both the cluster and the component, the component specification overrides the cluster's specification.

Naming a Cluster

You specify a name for the cluster when defining it. Generally, this name is given as the *dsname* in JCL. A cluster name that contains more than eight characters must be segmented by periods; one to eight characters may be specified between periods. A name with a single segment is referred to as an *unqualified name*. A name with more than one segment is referred to as a *qualified name*. Each segment of a qualified name is referred to as a *qualifier*.

You can, optionally, name the components of a cluster. Naming the data component of an entry-sequenced cluster, the data and index components of a key-sequenced cluster, or a linear data set, makes it easier to process the components individually.

If you do not specify a name for each component of a cluster, VSAM generates a name. VSAM uses the following format in order to generate 44-byte unique names:

clustername.Tbbbbbbb.DFDyyddd.Taaaaaaa.Tbbbbbbb

where:

clustername
is the first qualifier of the cluster name

yyddd
is the date (year, last two digits only, and day of the year)

aaaaaaa and **bbbbbbb**
are system generated values

Duplicate Data Set Names

Catalog management prevents you from cataloging two objects with the same name in the same catalog, and from altering the name of an object so that its new name duplicates the name of another object in the same catalog. However, this does not prevent duplication of names from one catalog to another. **If you have multiple catalogs, you should ensure that a data set name in one catalog is not duplicated in another catalog.**

The section, "Order of Catalog Use: DEFINE," in *Access Method Services Reference*, describes the order in which one of the catalogs available to the system is selected to contain the to-be-defined catalog entry. When you define an object, you should ensure that the catalog the system selects is the catalog you want the object entered into.

Note also that data set name duplication is not prevented when a user catalog is imported into a system; no check is made to determine whether the imported catalog contains an entry name that another catalog already in the system contains.

Note: If an unqualified data set name is the same as the first qualifier of a qualified data set name, the two cannot be placed in the same catalog unless one is the name of the catalog. For example, PAYROLL and PAYROLL.DATA cannot exist on the same catalog unless one is the catalog name.

Specifying Cluster Information

All the necessary descriptive information and the performance, security, and integrity options must be specified when you create a cluster. This information can apply to the data, the index, or both. Specify information for the cluster as a whole in the CLUSTER parameter. Specify information for the data only or the index only in the parameters DATA or INDEX.

Descriptive Information

Descriptive information includes:

- Type of data organization (key-sequenced, entry-sequenced, linear or relative record), as specified in the INDEXED|NONINDEXED|NUMBERED|LINEAR parameter.
- Average and maximum lengths of data records, as specified in the RECORDSIZE parameter. This parameter is not used for a linear data set.
- Length and position of the key field in the records of a key-sequenced data set, as specified in the KEYS parameter.
- Name and password of the catalog in which the cluster is to be defined, as specified in the CATALOG parameter.
- Volume serial number(s) of the volume(s) on which space is allocated for the cluster, as specified in the VOLUMES parameter.
- Amount of space to allocate for the cluster, as specified in the CYLINDERS|RECORDS|TRACKS parameter.
- Space is allocated for a linear data set with the number of control intervals equal to the number of records.
- Whether entries are re-created from information in the VVDS, or defined for the first time, as specified in the RECATALOG parameter.
- Whether the cluster is reusable for temporary storage of data, as specified in the REUSE|NOREUSE parameter. (See "Reusing a VSAM Data Set as a Work File" on page 30.)
- Minimum amount of I/O buffer space that must be allocated to process the data set, as specified in the BUFFERSPACE parameter. (See "Determining I/O Buffer Space for Nonshared Resources" on page 76.)

Performance Information

Information for performance includes:

- Whether records can span control intervals, as specified in the SPANNED parameter. This parameter is not allowed for linear and relative record data sets.
- The control interval size for VSAM to use (instead of letting VSAM calculate the size), as specified in the CONTROLINTERVALSIZE parameter. This parameter is not allowed for a linear data set.
- Whether to preformat control areas during initial loading of a data set, as specified in the SPEED|RECOVERY parameter. (See "Using Your Own Program to Load a Data Set" on page 29.)
- How to stage a cluster or component that is stored on a mass storage volume, as specified in the DESTAGEWAIT parameter. (This applies only to a system with the IBM 3850 Mass Storage System.)
- For an indexed cluster, additional information for performance includes:
  - Whether to replicate index records in order to reduce rotational delay, as specified in the REPLICATE parameter.
  - Whether to place the sequence set of an index adjacent to the data control area, as specified in the IMBED parameter.
  - Whether to place the cluster's index on a separate volume from data, as specified in the VOLUMES parameter for the index component.
  - The amount of free space to remain in the data component's control intervals and control areas when the data records are loaded, as specified in the FREESPACE parameter.
  - How data should be spread over multiple volumes, as specified in the KEYRANGES parameter.

All these performance options are discussed in Chapter 6, "Optimizing VSAM Performance" on page 65.

Security and Integrity Information

Information for security and integrity options includes:

- Passwords and related information.
- Identity of your own authorization routine to verify that a requester has the right to gain access to data, as specified in the AUTHORIZATION parameter.
- Identity of an I/O error-handling routine (the exception exit routine) that is entered if the program does not specify a SYNAD exit. This routine is specified in the EXCEPTIONEXIT parameter. For information on user exit routines, refer to *Data Facility Product: Customization*.
- Whether to verify that write operations have completed and that the data may be read, as specified in the WRITECHECK parameter.
- Whether and to what extent data is to be shared among systems, and jobs, as specified in the SHAREOPTIONS parameter.
- Whether to erase the information a data set contains when you delete the data set, as specified in the ERASE parameter.

Note: To control the erasing of data set information for a VSAM component whose cluster is RACF-protected and is cataloged in an integrated catalog facility catalog you can use an ERASE attribute that is kept in a generic or discrete profile. For information about how to specify and use this option, refer to *RACF General Information Manual* and associated RACF publications. (See also "Erasing Residual Data" on page 100.)

- Whether to destage a cluster or component that is stored on a mass storage volume before VSAM returns control to the program that issued the CLOSE macro. This is specified in the DESTAGEWAIT parameter. (This applies only to a system with the IBM 3850 Mass Storage System.)

See Chapter 8, "Data Security and Integrity" on page 99 for more information on the types of data protection available.

Allocating Space for a Data Set

When you define your data set, you must specify the amount of space to be allocated for it. Space for VSAM data sets may be allocated in units of cylinders, tracks, or records. The amount of space you allocate depends on the size of your data set and the index options you have chosen. "Index Options" on page 81 explains the index options you might choose.

When you allocate space for your data set, you can specify both a primary and a secondary allocation. A primary space allocation is the initial amount of space that is to be allocated. A secondary allocation is the amount of space that is to be allocated each time the cluster extends. Space for a data set defined in an integrated catalog facility catalog, or defined with the SUBALLOCATION parameter in a VSAM catalog, can be expanded to a maximum of 123 extents or to a maximum size of 2^{32} (approximately 4290000000) bytes. The last four extents are reserved for extending a data set when the last extent cannot be allocated in one piece. For data sets defined in VSAM catalogs defined with the UNIQUE parameter, the space for a data set can be expanded to a maximum of 16 extents per volume.

You can specify space allocation at the cluster or alternate index level, at the data level only, or at both the data and index levels. 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 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 (when it is not specified at the cluster level).

VSAM acquires space in increments of control areas. The control area size is based on primary and secondary space allocations.

- If either primary *or* secondary allocation is smaller than one cylinder, the smaller value is used as the control area size. (If RECORDS is specified, the allocation is rounded up to full tracks.)
- If both primary *and* secondary allocation are equal to or larger than one cylinder, the control area size is one cylinder. (A control area is never larger than one cylinder.)

Small Data Sets

If you allocate space for a data set in a unit smaller than one cylinder, VSAM allocates space in tracks when defining the data set. For data sets less than 1 cylinder in size, it is advantageous to specify the maximum number of tracks required in the primary allocation for the data component, 1 track for the sequence set index (which should not be embedded), and no secondary allocation for either data or index.

VSAM checks the smaller of primary and secondary space values against the specified device's cylinder size. If the smaller quantity is greater than or equal to the device's cylinder size, the control area is set equal to the cylinder size. If the smaller quantity is less than the device's cylinder size, the size of the control area is set equal to the smaller space quantity.

For example:

CYLINDERS(5,10)	Results in a 1-cylinder control area size
TRACKS(100,3)	Results in a 3-track control area size
RECORDS(2000,5)	Assuming 10 records would fit on a track, results in a 1-track control area size (minimum control area size is 1 track)
TRACKS(3,100)	Results in a 3-track control area size

A spanned record cannot be larger than a control area less the control information (10 bytes per control interval), so do not specify large spanned records and a small primary or secondary allocation not large enough to contain the largest spanned record.

VSAM data sets cataloged in an integrated catalog facility catalog are allocated with the CONTIG attribute if the allocation unit is TRACKS. Therefore, the primary and secondary allocations are in contiguous tracks.

Multiple Cylinder Data Sets

To define a data set larger than one cylinder, calculate the number of cylinders needed for data in a newly defined data set, and specify this amount in cylinders for the primary allocation. Make the secondary allocation equal to or greater than one cylinder, but less than the primary allocation. "Sample Calculation of Space Allocation for a VSAM Key-Sequenced Data Set" on page 25 demonstrates how to calculate the size of a data set.

When you allocate space for your cluster, you must consider the index options you have chosen:

- If IMBED is specified (to place the sequence set with the data), the data allocation includes the sequence set. This means that more space must be given for data allocation when IMBED is specified.

- If REPLICATE is specified and IMBED is not specified, an allocation of 3 primary and 1 secondary track for the index set is a valid choice.
- If the REPLICATE option is not used, specify 1 primary and 1 secondary track for the index set.

For information about index options, refer to "Index Options" on page 81.

Linear Data Sets

Space may be allocated either in tracks, cylinders, or records. For allocation, a record is equal to a control interval. Refer to Figure 15 on page 26 for DASD capacity on various devices.

A linear data set is created by access method services. When a linear data set is defined, the catalog will force the block size to 4096 bytes. If the specified BUFFERSPACE is greater than 8192 bytes, it will be decremented to a multiple of 4096. If BUFFERSPACE is less than 8192, access method services will issue a message and fail the command.

Sample Calculation of Space Allocation for a VSAM Key-Sequenced Data Set

This example shows how to calculate the size of the data component for a key-sequenced data set.

The following are assumed for the calculations:

Device type	3380
Unit of space allocation	cylinders
Data control interval size	1024 bytes
Physical block size (calculated by VSAM)	1024 bytes
Record size	200 bytes
Free space definition—control interval	20%
Free space definition—control area	10%
Number of records to be loaded	3000
Index options defined	IMBED/REPLICATE

Figure 15 provides information on DASD capacity in terms of the number of physical blocks per track and the number of tracks per cylinder.

Direct Access Device Characteristics

Figure 15 lists the physical characteristics and information about DASD capacity in terms of the number of physical blocks per track and the number of tracks per cylinder.

Figure 15. DASD Physical Characteristics

DASD Characteristics			**Block Size**			
			512	**1024**	**2048**	**4096**
Type	**Trk/Cyl**	**Cyl/Vol**	**Number of Physical Blocks/Track**			
2305-2	8	96	20	12	6	3
3330-1	19	404	20	11	6	3
3330-11	19	808	20	11	6	3
3340/3344	12	348	12	7	*	2
3350	30	555	27	15	8	4
3375	12	959	40	25	14	8
3380[1]	15	885	46	31	18	10
3380[2]	15	1770	46	31	18	10
3380[3]	15	2655	46	31	18	10

Note:

* Not selected for this device.

[1] 3380 single capacity Models A04,AA4,B04,AD4,BD4,AJ4,BJ4,CJ2.
[2] 3380 double capacity Models AE4,BE4.
[3] 3380 triple capacity Models AK4,BK4.

Using the specifications in Figure 15, you can calculate space for the data component as follows:

1. Maximum number of records per control interval[1] ((1024−10)/200) = 5
2. Subtract bytes of free space (20% * 1024) = 204
3. Number of loaded records per control interval (1024−10−204)/200 = 4
4. Number of physical blocks per track[2] = 31
5. Number of control intervals per track = 31
6. Maximum number of control intervals per control area[3] (31 * 14) = 434
   (1 cylinder = 15 tracks − 1 track for index sequential set)
7. Number of loaded control intervals per control area = 391
   (434 − 10% * 434)
8. Number of loaded records per cylinder (4 * 391) = 1564
9. Total space for data component (3000/1564) (rounded) = 2 cylinders

Notes:

[1] The value (1024 − 10) is the control interval length minus 10 bytes for 2 RDFs and 1 CIDF.

[2] On an IBM 3380, 31 physical blocks with 1024 bytes can be stored on one track. (See the physical block sizes in Figure 15 on page 26.)

[3] The value (31 * 14) is the number of physical blocks per track multiplied by the number of data tracks per cylinder (15 minus 1 for the embedded sequence set control interval). (See the information on tracks per cylinder in Figure 15 on page 26.)

Modifying Attributes of a Component

After a data set has been defined, you can change some of its attributes using the access method services command, ALTER. You identify the component by name, and specify the new attributes. ALTER can also be used as a migration path to change an entry-sequenced data set, with the proper attributes, into a linear data set. For an example of altering an entry-sequenced data set into a linear data set, see "Example 3: Alter the Cataloged Attributes of VSAM Data Sets" on page 166.

Loading a Data Set

After a data set is defined, data can be loaded into it. This entails moving records from a source data set to the VSAM data set you have defined. After the data records are written in the VSAM data set and a CLOSE operation is performed, the loading is complete, and the VSAM data set can be accessed.

When loading a key-sequenced data set, the records to be loaded must be in ascending order, with no duplicates in the input data set. With an entry-sequenced or relative record data set, the records to be loaded can be in any order. Data is loaded as single logical records in either key order or physical order. As a result, reorganization takes place. This reorganization can cause any of the following:

- Physical relocation of logical records
- Alteration of a record's relative position within the data set
- Redistribution of free space throughout the data set
- Reconstruction of the prime index

You can load all the records in one job or in several jobs. In subsequent jobs, VSAM stores records as before, extending the data set as required.

Using REPRO to Load a Data Set

The REPRO command causes access method services to retrieve records from a sequential, indexed-sequential, or VSAM data set and store them in VSAM format in a key-sequenced data set, a relative record data set, an entry-sequenced data set, or a sequential data set. This command is also used to load data from one linear data set into another linear data set.

When records are to be stored in key sequence, index entries are created and loaded into an index component as data control intervals and control areas are filled. Free space is left as indicated in the cluster definition in the catalog, and, if indicated in the definition, records are stored on particular volumes according to key ranges.

VSAM data sets used for either input or output must be cataloged. Sequential and indexed-sequential data sets need not be cataloged.

If a sequential or indexed-sequential data set is not cataloged, include the appropriate volume and unit parameters on your DD statement. Also, supply a minimum set of DCB parameters when the input data set is sequential or indexed sequential, and/or the output data set is sequential. The following table shows the four key parameters.

Parameters	User Must Supply	Default if Not Supplied
DSORG	IS	PS
RECFM	F, FB, V, VB, VS, VBS	U
BLKSIZE	block size	none
LRECL	logical record length	BLKSIZE for F or FB BLKSIZE-4 for V, VB, VS, VBS

The one parameter not supplied by default is BLKSIZE; you must supply this value. The DCB parameter DSORG must be supplied via the DD statement. The DCB parameters RECFM, BLKSIZE, and LRECL can be supplied via the DSCB or header label of a standard labeled tape, or by the DD statement.

If you are loading a VSAM data set into a sequential or indexed-sequential data set, you must remember that the 3-byte VSAM record definition field (RDF) is not included in the VSAM record length. When REPRO attempts to copy a VSAM record whose length is within 4 bytes of LRECL, a recoverable error occurs and the record is not copied.

Access method services does not support records greater than 32760 bytes for non-VSAM data sets (LRECL=X is not supported). If the logical record length of a non-VSAM input data set is greater than 32760 bytes, or if a VSAM data set defined with a record length greater than 32760 is to be copied to a sequential data set, the REPRO command terminates with an error message.

Records in an indexed-sequential data set that have a fixed-length, unblocked format with a relative-key-position of zero are preceded by the key string when used as input. The records in the output data set must have a record length defined that includes the extended length caused by the key string. To copy "dummy" indexed-sequential records (records with X'FF' in the first byte) specify the DUMMY option in the ENVIRONMENT parameter.

The REPRO operation is terminated if:

- One physical I/O error is encountered while writing to the output data set
- A total of four errors is encountered in any combination of the following:
  - Logical error while writing to the output data set
  - Logical error while reading the input data set
  - Physical error while reading the input data set

For information on physical and logical errors, refer to *Data Facility Product: Customization.*

Using Your Own Program to Load a Data Set

To use your own program to load a key-sequenced data set, first sort the records (or build them) in key sequence, then store them by sequential access (using the PUT macro). When you are initially loading a data set, direct access is not permitted. For more information on inserting records into a data set, refer to "Inserting a Record" on page 46.

VSAM uses the high-used RBA field to determine if a data set is empty or not. An implicit verify will update the high-used RBA. Immediately after a data set is defined, the high-used RBA value is zero.

The terms "create mode," "load mode," and "initial data set load" are synonyms for the process of inserting records into an empty VSAM data set. This processing is started by calling the VSAM OPEN macro. It continues while records are added following the (successful) open and concludes when the data set is closed. After loading and closing the data set, the high used RBA is equal to the offset of the first byte of the first unused control interval in the data set.

Note: If create mode loading of an entry-sequenced data set abends before the data set is loaded, a verify will not show the data set as being empty. An entry-sequenced data set must be defined with the RECOVERY option or it cannot be verified. You must specify the REUSE attribute for this data set and reset it to make the data set empty.

Certain restrictions apply during load mode processing:

- PUT and CHECK are the only macros you can use.
- Do not use improved control interval processing.
- You cannot do update or input processing until the data set has been loaded and closed.
- Specify only one string in the ACB (STRNO>1 is not permitted).
- Specify only one RPL for a relative record data set defined with the SPEED parameter.
- Do not specify local shared resources (LSR) or global shared resources (GSR).
- You may not open a data set for input processing.
- You cannot share data sets.
- Direct processing is not permitted (except relative record keyed direct).

If your application calls for direct processing during load mode, you can avoid this restriction by doing the following:

1. Open the empty data set for load mode processing.
2. Sequentially write one or more records. (These may be "dummy" records.)
3. Close the data set to terminate load mode processing.
4. Reopen the data set for normal processing. (You can now resume loading.)

For information on how to use exit routine macros when loading records into a data set, see *Data Facility Product: Customization*.

During load mode, each control area can be preformatted as records are loaded into it. Preformatting is useful for recovery if an error occurs during loading; however, performance is better during initial data set load without preformatting. The RECOVERY parameter of the access method services DEFINE command is used to indicate that VSAM is to preformat control areas during load mode.

Preformatting clears all previous information from the direct access storage area and writes end-of-file indicators. For VSAM, an end-of-file indicator consists of a control interval with a CIDF equal to zeros.

- For an entry-sequenced data set, VSAM writes an end-of-file indicator in every control interval in the control area.
- For a key-sequenced data set, VSAM writes an end-of-file indicator in the first control interval in the control area following the preformatted control area. (The preformatted control area contains free control intervals.)
- For a relative record data set, VSAM writes an end-of-file indicator in the first control interval in the control area following the preformatted control area. All RDFs in the preformatted control area are marked "slot empty."

As records are loaded into a preformatted control area of an entry-sequenced data set, an end-of-file indicator following the records indicates how far loading has progressed. If an error occurs that prevents loading from continuing, you can identify the last successfully loaded record by reading to end of file. You can then resume loading at that point, after verifying the data set.

Without preformatting (SPEED parameter), an end-of-file indicator is written only after the last record is loaded. If an error occurs that prevents loading from continuing, you may not be able to identify the last successfully loaded record and you may have to reload the records from the beginning.

Note: Remember that, if you specify SPEED, it will be in effect for load mode processing. After load mode processing, RECOVERY will be in effect, regardless of the DEFINE specification.

Reusing a VSAM Data Set as a Work File

VSAM allows you to create reusable data sets that you can use as work files. To do this, you must define the data set as reusable and specify that it be reset when you open it.

A data set that is not reusable may be loaded once and only once. After the data set is loaded, it can be read, written into, and the data in it can be modified. However, the only way to remove the set of data is to use the access method services command DELETE, which deletes the entire data set. If the data set is to be used again, it must be re-created with the access method services command DEFINE.

Instead of using the DELETE - DEFINE sequence, you can specify the REUSE parameter in the DEFINE CLUSTER|ALTERNATEINDEX command. This allows you to treat a filled data set as if it were empty and load it again and again regardless of its previous contents.

A reusable data set may be a key-sequenced data set, an entry-sequenced data set, a linear data set, or a relative record data set that resides on one or more volumes. A reusable base cluster cannot have an alternate index, and it cannot be associated with key ranges. Reusable data sets on an IBM 3850 Mass Storage System must begin on a cylinder boundary to prevent staging in RESET mode.

VSAM uses a *high-used RBA* (relative byte address) field to determine if a data set is empty or not. Immediately after a data set is defined, the high-used RBA value is zero. After loading and closing the data set, the high-used RBA is equal to the offset of the last byte in the data set. In a reusable data set, this high-used RBA field can be reset to zero at OPEN, and VSAM can use this data set like a newly-defined data set.

Copying a Data Set

You may want to copy a data set or merge two data sets for a variety of reasons. For example, you may want to create a test copy, you may want two copies to use for two different purposes, or you may want to keep a copy of back records before updating a data set. You can use the access method services REPRO command to copy data sets.

Using REPRO to Copy a Data Set

You can use the REPRO command to do any of the following:

- Copy or merge a VSAM data set into another VSAM data set.
- Copy or merge a sequential data set into another sequential data set.
- Copy an alternate index as a key-sequenced VSAM data set.
- Copy a VSAM data set whose records are fixed length into an empty VSAM relative record data set.
- Convert a sequential or indexed-sequential data set into a VSAM data set.
- Copy a data set (other than a catalog) to reorganize it. Data set reorganization is an automatic feature.

When copying to a key-sequenced data set, the records to be copied must be in ascending order, with no duplicates in the input data set. With an entry-sequenced data set, the records to be copied can be in any order.

Because data is copied as single logical records in either key order or physical order, automatic reorganization takes place. The reorganization can cause any of the following:

- Physical relocation of logical records
- Alteration of a record's physical position within the data set
- Redistribution of free space throughout the data set
- Reconstruction of the VSAM indexes

If you are copying to or from a sequential or indexed-sequential data set that is not cataloged, you must include the appropriate volume and unit parameters on your DD statements. For more information on these parameters, see "Using REPRO to Load a Data Set" on page 27.

Figure 16 describes how the data from the input data set is added to the output data set when the output data set is an empty or nonempty entry-sequenced, sequential, key-sequenced, linear or relative record data set.

Figure 16. Adding Data to Various Types of Output Data Sets

Type of Data Set	Empty	Nonempty
Entry-Sequenced/ Sequential	Creates new data set in sequential order.	Adds records in sequential order to the end of the data set.
Key-Sequenced	Creates new data set in key sequence and builds an index.	Merges records by key and updates the index. Unless the REPLACE option is specified, records whose key duplicates a key in the output data set are lost.
Linear	Creates new linear data set in relative byte order.	Adds data to control intervals in sequential order to the end of the data set.
Relative Record	Creates a new data set in relative record sequence, beginning with 1.	Records from another relative record data set are merged, keeping their old record numbers. Unless the REPLACE option is specified, a new record whose number duplicates an existing record number is lost. Records from any other type of organization cannot be copied into a nonempty relative record data set.

The REPRO operation is terminated if:

- One physical I/O error is encountered while writing to the output data set
- A total of four errors is encountered in any combination of the following:
  - Logical error while writing to the output data set
  - Logical error while reading the input data set
  - Physical error while reading the input data set

Creating an Alternate Index

An alternate index can be defined over a key-sequenced cluster or an entry-sequenced cluster. An alternate index cannot be defined to support a reusable cluster, a relative record cluster, a catalog, another alternate index, a linear cluster or a non-VSAM data set. For information on the structure of an alternate index, see "Accessing Records through an Alternate Index" on page 15.

The sequence for building an alternate index is as follows:

1. Define the base cluster, using the DEFINE CLUSTER command.
2. Load the base cluster either by using the REPRO command or by writing your own program to load the data set.
3. Define the alternate index, using the DEFINE ALTERNATEINDEX command.
4. Relate the alternate index to the base cluster, using the DEFINE PATH command. The base cluster and alternate index are described by entries in the same catalog.
5. Build the alternate index, using the BLDINDEX command.

VSAM uses three catalog entries to describe an alternate index:

- An alternate index entry describes the alternate index as a key-sequenced cluster.
- A data entry describes the alternate index's data component.
- An index entry describes the alternate index's index component.

Attributes of the alternate index's components can be specified separately from the attributes of the alternate index. If attributes are specified for the alternate index as a whole and not for the components, these attributes (except for password and USVR security attributes) apply to the components. If the attributes are specified for the components, they override any attributes specified for the entire alternate index.

Naming an Alternate Index

You specify an entry name for the alternate index when defining it. The entry name is the JCL DD statement's *dsname*. You can also name the alternate index's components so a user program can open and process the alternate index's data or index component as a data set. For more details on this kind of processing, see Chapter 7, "Processing Control Intervals" on page 87.

Specifying Alternate Index Information

When you define an alternate index, you specify descriptive information and performance, security, and data integrity options. The information can apply to the alternate index's data component, its index component, or the alternate index as a whole. Information for the entire alternate index is specified with the ALTERNATEINDEX parameter and its subparameters. Information for the data component or the index component is specified with the parameter DATA or INDEX.

Descriptive information includes:

- The name and password of the base cluster related to the alternate index, as specified in the RELATE parameter.
- Whether the alternate index entries are re-created from information in the VVDS, or defined for the first time, as specified in the RECATALOG parameter.
- Whether the alternate index is reusable, as specified in the REUSE parameter.
- Average and maximum lengths of alternate index records, as specified in the RECORDSIZE parameter.
- Length and position of the alternate key field in data records of the base cluster, as specified in the KEYS parameter.
- Name and password of the catalog that will contain the alternate index's entries, as specified in the CATALOG parameter. (This must be the same catalog that contains the base cluster's entries.)
- Volume serial number(s) of the volume(s) on which space is allocated for the alternate index, as specified in the VOLUMES parameter.
- Amount of space to allocate for the alternate index, as specified in the CYLINDERS|RECORDS|TRACKS parameter.

- The minimum amount of I/O buffer space that OPEN must provide when the program processes the alternate index's data, as specified in the BUFFERSPACE parameter.

The performance options and the security and integrity information for the alternate index are the same as that for the cluster. See "Specifying Cluster Information" on page 21.

How an Alternate Index Is Built

When an alternate index is built by BLDINDEX processing, the alternate index's volume and the base cluster's volume must be mounted. Any volumes identified with the WORKFILES parameter must also be mounted. The base cluster cannot be empty (that is, its high-used RBA value cannot be zero). Each record's alternate key value must be unique, unless the alternate index was defined with the NONUNIQUEKEY attribute.

Access method services opens the base cluster to read the data records sequentially, sorts the information obtained from the data records, and builds the alternate index data records.

The base cluster's data records are read and information is extracted to form the key-pointer pair:

- When the base cluster is entry-sequenced, the alternate key value and the data record's RBA form the key-pointer pair.
- When the base cluster is key-sequenced, the alternate key value and the data record's prime key value form the key-pointer pair.

The key-pointer pairs are sorted in ascending alternate key order.

If your program provides enough virtual storage, access method services performs an internal sort. (The sorting of key-pointer pairs takes place entirely within virtual storage.) If you do not provide enough virtual storage for an internal sort, or if you specify the EXTERNALSORT parameter, access method services defines and uses two sort work files and sorts the key-pointer pairs externally. If you have the available virtual storage, an internal sort is faster. An external sort is slower because of I/O to the work files, but, if you have a large data base, an external sort resolves the virtual storage problem.

For information on calculating the amount of virtual storage required to sort records, see Appendix E, "Calculating Virtual Storage Space for an Alternate Index" on page 201.

After the key-pointer pairs are sorted into ascending alternate key order, access method services builds alternate index records for key-pointer pairs. When all alternate index records are built and loaded into the alternate index, the alternate index and its base cluster are closed.

Alternate Index Maintenance

VSAM assumes alternate indexes are synchronized with the base cluster at all times and makes no synchronization checks during open processing; therefore, all structural changes made to a base cluster must be reflected in its alternate index or indexes. This is called *index upgrade.*

You can maintain your own alternate indexes or you can have VSAM maintain them. When the alternate index is defined with the UPGRADE attribute of the DEFINE command, VSAM updates the alternate index whenever there is a change to the associated base cluster. VSAM opens all upgrade alternate indexes for a base cluster whenever the base cluster is opened for output. If you are using control interval processing, you cannot use UPGRADE. (See Chapter 7, "Processing Control Intervals" on page 87.)

All the alternate indexes of a given base cluster that have the UPGRADE attribute belong to the *upgrade set.* The upgrade set is updated whenever a base data record is inserted, erased, or updated. The upgrading is part of a request and VSAM completes it before returning control to your program. If upgrade processing is interrupted due to a machine or program error such that a record is missing from the base cluster but its pointer still exists in the alternate index, record management will synchronize the alternate index with the base cluster by allowing you to reinsert the missing base record. However, if the pointer is missing from the alternate index, i.e., the alternate index does not reflect all the base cluster data records, you must rebuild your alternate index to resolve this discrepancy.

If you specify NOUPGRADE in the DEFINE command when the alternate index is defined, insertions, deletions, and changes made to the base cluster will not be reflected in the associated alternate index.

When a path is opened for update, the base cluster and all the alternate indexes in the upgrade set are allocated. If updating the alternate indexes is unnecessary, you can specify NOUPDATE in the DEFINE PATH command and only the base cluster will be allocated. In that case, VSAM does not automatically upgrade the alternate index. If two paths are opened with MACRF=DSN specified in the ACB macro, the NOUPDATE specification of one may be nullified if the other path is opened with UPDATE specified.

Defining a Path

| Before an alternate index is built (by BLDINDEX processing), you need to establish the relationship between an alternate index and its base cluster, using the access method services command, DEFINE PATH.

When your program opens a path for processing, both the alternate index and its base cluster are opened. When data in a key-sequenced base cluster is read or written using the path's alternate index, keyed processing is used. RBA processing is allowed only for reading or writing an entry-sequenced data set's base cluster.

Defining a Page Space

A page space is a system data set that contains pages of virtual storage. The pages are stored into and retrieved from the page space by the auxiliary storage manager. A page space is a nonindexed data set (an entry-sequenced cluster) that is entirely preformatted before it is used, and is contained on a single volume. In a system with the Mass Storage System, a page space cannot be defined on a mass storage volume. A page space cannot be opened as a user data set.

A page space has a maximum usable size equal to 65535 paging slots (records). For further details, see the description for space size declarations (CYLINDERS, RECORDS, and TRACKS for "DEFINE PAGESPACE") in *Access Method Services Reference.*

You can define a page space in a user catalog, then move the catalog to a new system and establish it as the system's master catalog. When you define a page space in a user catalog, code a STEPCAT or JOBCAT statement to identify and allocate the catalog. A page space cannot be used if its entry is in a user catalog.

When you issue a DEFINE PAGESPACE command, the system creates an entry in the catalog for the page space, then preformats the page space. If an error occurs during the preformatting process (for example, an I/O error or an allocation error), the page space entry remains in the catalog even though no space for it exists. Issue a DELETE command to remove the page space entry before you redefine the page space.

Each page space is represented by two entries in the catalog: a cluster entry and a data entry. (A page space is conceptually an entry-sequenced cluster.) Both of these entries should be password protected if the page space is to be password protected.

Note: The passwords you specify with the DEFINE PAGESPACE command are put in both the page space's cluster entry and its data entry. When you define page spaces during system generation (sysgen), use the ALTER command to add passwords to each entry, because passwords cannot be specified during system generation. Unless you ensure that the catalog containing the page space entry is password protected, a user can list the catalog's contents and find out each entry's passwords.

A page space is made known to the system as a system data set at sysgen or through members of a partitioned data set: SYS1.PARMLIB. To be used as a page space, it must be defined in a master catalog.

Checking for Problems in Catalogs and Data Sets

VSAM provides you with several means of locating problems in your catalogs and data sets. Procedures for listing catalog entries and printing out data sets are described below. You can also use the access method services REPRO command to copy a data set to an output device. For more information on this option, refer to "Copying a Data Set" on page 31. The access method services VERIFY command provides a means of checking and restoring end-of-data-set values after system failure. This is described in "Using VERIFY to Synchronize

Values" on page 63. Information on using the DIAGNOSE command to indicate the presence of invalid data or relationships in the BCS and VVDS is found in *Catalog Administration Guide*. The access method services EXAMINE command allows the user to analyze and report on the structural inconsistencies of key-sequenced data set clusters. This is described in Chapter 12, "Checking a VSAM Key-Sequenced Data Set Cluster for Errors" on page 139.

Listing Catalog Entries

After you define a catalog or data set, use the access method services command LISTCAT to list all or part of a catalog's entries. This listing will show information about objects defined in the catalog, such as:

- Attributes of the object
- Creation and expiration dates
- Protection specification
- Statistics regarding the dynamic usage or accessing of the data set represented by the entry
- Space allocation
- Volume information

The listing can be customized by limiting the number of entries, and the information about each entry, that is printed.

Printing a Data Set

In the event of a problem, you can use the access method services command PRINT to list part or all of a key-sequenced, relative record, a linear or entry-sequenced VSAM data set, an alternate index, or a catalog. If you use the relative byte address, you may print part of a linear data set. Partial printing will be rounded up to 4096 byte boundaries. The components of a key-sequenced data set or an alternate index can be listed individually by specifying the component name as the data set name. An alternate index is printed as though it were a key-sequenced cluster.

Entry-sequenced and linear data sets are listed in physical sequential order. Key-sequenced data sets can be listed in key order or in physical sequential order. Relative record data sets are listed in relative record number sequence. A base cluster can be listed in alternate key sequence by specifying a path name as the data set name for the cluster.

Only the data content of logical records is listed. System-defined control fields are not listed. Each record listed is identified by one of the following:

- The relative byte address (RBA) for entry-sequenced data sets
- The key for indexed-sequential and key-sequenced data sets, and for alternate indexes
- The record number for relative record data sets

Note: If four logical and/or physical errors are encountered while trying to read the input, printing is terminated.

To use the PRINT command to print a catalog, access method services must be authorized. For information about program authorization, see "Using the Authorized Program Facility (APF)" in *System Macros and Facilities*.

Chapter 4. Processing a Data Set

General-Use Programming Interface

This chapter is intended to help you process a data set. It primarily contains general-use programming interfaces, which allow you to write programs that use the services of MVS/XA Data Facility Product.

VSAM data sets are processed using VSAM macro instructions. You can process a VSAM data set to read, update, add, or delete data by following this procedure:

1. Create an access method control block to identify the data set to be opened. Use the ACB or GENCB macro instruction to do this.
2. Create an exit list to specify the optional exit routines that you supply, using the EXLST or GENCB macro instruction.
3. Optionally, create a resource pool, using the BLDVRP macro. (See Chapter 10, "Sharing Resources among Data Sets" on page 127.)
4. Connect your program to the data set you want to process, using the OPEN macro instruction.
5. Create a request parameter list to define your request for access, using the RPL or GENCB macro instruction.
6. Manipulate the control block contents using the GENCB, TESTCB, MODCB and SHOWCB macros.
7. Request access to the data set, using one or more of the VSAM request macro instructions (GET, PUT, POINT, ERASE, CHECK, and ENDREQ).
8. Disconnect your program from the data set, using the CLOSE macro instruction.

For information on the syntax of each macro, and for coded examples of the macros, see *VSAM Administration: Macro Instruction Reference*.

Creating an Access Method Control Block

Before you can open a data set for processing, you must create an access method control block (ACB) that identifies the data set to be opened, specifies the type of processing, specifies the basic options, and indicates whether exit routines are to be used while the data set is being processed.

Include the following information in your ACB for OPEN to prepare the kind of processing required by your program:

- The address of an exit list for your exit routines. You use the EXLST macro to construct the list.
- If you are processing concurrent requests, the number of requests (STRNO) defined for processing the data set. For more information on concurrent requests, refer to "Concurrent Requests" on page 53.

- The size of the I/O buffer virtual storage space and/or the number of I/O buffers that you are supplying for VSAM to process data and index records.
- The password required for the type of processing desired.
- The processing options which you plan to use:

End of General-Use Programming Interface

Product-Sensitive Programming Interface

  - Keyed, addressed, control interval, or a combination

End of Product-Sensitive Programming Interface

General-Use Programming Interface

  - Sequential, direct, or skip sequential access, or a combination
  - Retrieval, storage, or update (including deletion), or a combination
  - Shared or nonshared resources

- The address and length of an area for error messages from VSAM.

You can use the ACB macro to build an access method control block when the program is assembled, or the GENCB macro to build a control block when the program is executed. For information on the advantages and disadvantages of using GENCB, see "Manipulating Control Block Contents" on page 44.

Creating an Exit List

In order to access exit routines during data set processing, you must specify the addresses of your exit routines using the EXLST macro. Any number of ACB macros in a program can indicate the same exit list for the same exit routines to do all the special processing for them, or they can indicate different exit lists. You can use exit routines for:

Analyzing physical errors: When VSAM encounters an error in an I/O operation that the operating system's error routine cannot correct, the error routine formats a message for your physical error analysis routine (the SYNAD exit) to act on.

Analyzing logical errors: Errors not directly associated with an I/O operation, such as an invalid request, cause VSAM to exit to your logical error analysis routine (the LERAD exit).

End-of-data-set processing: When your program requests a record beyond the last record in the data set, your end-of-data-set routine (the EODAD exit) is given control. The end of the data set is beyond either the highest addressed or the highest keyed record, depending on whether your program is using addressed or keyed access.

Journalizing transactions: To journalize the transactions against a data set, you might specify a journal routine (the JRNAD exit). To process a key-sequenced data set by way of addressed access, you need to know whether any RBAs changed during keyed processing. When you're processing by key, VSAM exits to your routine for noting RBA changes before writing a control interval in which there is an RBA change.

User processing: User processing (UPAD) exits are available to assist subsystems which need to dispatch new units of work. The UPAD wait exit is given control before VSAM issues any WAIT SVCs. The UPAD post exit may be used to facilitate the use of cross memory processing.

The EXLST macro is coordinated with the EXLST parameter of an ACB or GENCB macro used to generate an ACB. To make use of the exit list, you must code the EXLST parameter in the ACB.

Data Facility Product: Customization provides a more detailed description of exit routines.

You can use the EXLST macro to build an exit list when the program is assembled, or the GENCB macro to build an exit list when the program is executed. For information on the advantages and disadvantages of using GENCB, see "Manipulating Control Block Contents" on page 44.

Opening a Data Set

Before your program can access a data set, it must issue the OPEN macro to open the data set for processing. Opening a data set causes VSAM to do the following:

- Mount the volumes on which the data set is stored, if necessary. VSAM calls for the required volumes to be mounted by examining the DD statement indicated by the ACB macro and the volume information in the catalog.

  Note: For multivolume data sets defined in integrated catalog facility catalogs, OPEN requires all primary volumes to be parallel mounted.

- 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 a key-sequenced data set).

- Construct the internal control blocks that VSAM needs to process your requests for access to the data set.

  VSAM determines what processing options are to be used by merging the information in the DD statement and the catalog definition of the data set with the information in the access method control block and the exit list.

  The order of precedence is:

  1. The DD statement AMP parameters
  2. The ACB, EXLST, or GENCB parameters
  3. The catalog entry for the data set

  For example, if information about buffer space is specified both in the DD statement and in the ACB or GENCB macro, the values in the DD statement override those in the macro. The catalog entry acts as a default when buffer space specified in the DD statement or in the macro is either less than the minimum specified when the data set was defined, or when buffer space is not specified in either the DD statement or the macro.

- Check for consistency of updates to the prime index and data components if you are opening a key-sequenced data set, an alternate index or a path. If a data set and its index have been updated separately, VSAM issues a warning message to indicate a time stamp discrepancy.

- Check the password your program specified in the ACB PASSWD parameter against the appropriate password (if any) in the catalog definition of the data. The password required depends on the kind of access specified in the access method control block (for example, access for retrieval or for update), as follows:

  - Full access allows you to perform all operations (retrieving, updating, inserting, and deleting) on a data set and any index or catalog record associated with it. The master password allows you to delete or alter the catalog entry for the data set or catalog it protects.

  - Control interval access requires the control password. This password allows you to use control interval access and to retrieve, update, insert, or delete records in the data set it protects. For information on the use of control interval access, see Chapter 7, "Processing Control Intervals" on page 87.

  - Update access requires the update password, which allows you to retrieve, update, insert, or delete records in the data set it protects.

  - Read access requires the read password, which allows you to examine records in the data set it protects; the read password does not allow you to add, change, or delete records.

  A password of one level authorizes you to do everything authorized by a password of a lower level.

  In addition to passwords, you can have protection provided by the Resource Access Control Facility (RACF), an IBM program product. When RACF protection and password protection are both applied to a data set, password protection is bypassed, and use is authorized solely through the RACF checking system. Password and RACF protection are further described in Chapter 8, "Data Security and Integrity" on page 99.

- If an error occurs during open, a component opened for update processing may improperly close (leaving the open-for-output indicator on). At OPEN, VSAM issues an implicit 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 CLOSE. If the data set is opened for input, however, the open-for-output indicator is left on.

Creating a Request Parameter List

After you have connected your program to the data set, you can issue requests for access. A request parameter list defines a request. It identifies the data set to which the request is directed by naming the ACB macro that defines the data set. Each request macro (GET, PUT, ERASE, POINT, CHECK, and ENDREQ) gives the address of the request parameter list that defines the request.

You can use the RPL macro to generate a request parameter list when your program is assembled, or the GENCB macro to build a request parameter list when your program is executed. For information on the advantages and disadvantages of using GENCB, see "Manipulating Control Block Contents" on page 44.

When you define your request, specify only the processing options appropriate to that particular request. Parameters not required for a request are ignored. For example, if you switch from direct to sequential retrieval with a request parameter list, you don't have to zero out the address of the field containing the search argument (ARG=address).

The following information defines your request:

- Access by address (RBA), key, or relative record number. Address access can be sequential or direct; key access can be sequential, skip sequential, or direct. Access can be forward (next sequential record) or backward (previous sequential record). Access can be for updating or not updating. A nonupdate direct request to retrieve a record can, optionally, cause VSAM to position to the following record for subsequent sequential access. For more information on VSAM positioning, see "Pointing VSAM to a Record" on page 49.
- RPLs (including RPLs defined by a chain), either synchronous, so that VSAM does not give control back to your program until the request completes, or asynchronous, so that your program can continue to process or issue other requests while the request is active. With asynchronous requests, your program must use the CHECK macro to suspend its processing until the request completes. For more information on synchronous and asynchronous processing, see "Asynchronous Requests" on page 54.
- For a keyed request, either a generic key (a leading portion of the key field), or a full key to which the key field of the record is to be compared.
- For retrieval, either a data record to be placed in a work area in your program or the address of the record within VSAM's buffer to be passed to your program. For requests that involve updating or inserting, the work area in your program contains the data record.
- For a request to directly access a control interval, specify the RBA of the control interval. With control interval access, you are responsible for maintaining the control information in the control interval. If VSAM's buffers are used, VSAM allows control interval and stored record operations simultaneously. If your program provides its own buffers, only control interval processing is allowed. For information on control interval access, see Chapter 7, "Processing Control Intervals" on page 87.

Chaining Request Parameter Lists

You can chain request parameter lists together to define a series of actions for a single GET or PUT. For example, each parameter list in the chain could contain a unique search argument and point to a unique work area. A single GET macro would retrieve a record for each request parameter list in the chain.

A chain of request parameter lists is processed serially as a single request. (Chaining request parameter lists is not the same as processing concurrent requests in parallel.) Processing in parallel requires that VSAM keep track of many positions in a data set.

Each request parameter list in a chain should have the same OPTCD subparameters. Having different subparameters may cause logical errors. You can't chain request parameter lists for updating or deleting records—only for retrieving records or storing new records. You can't process records in the I/O buffer with chained request parameter lists. (RPL OPTCD=UPD and RPL OPTCD=LOC are invalid for a chained request parameter list.)

With chained request parameter lists, a POINT, a sequential or skip-sequential GET, or a direct GET with positioning requested (RPL OPTCD=NSP) causes VSAM to position itself at the record following the record identified by the last request parameter list in the chain.

When you are using chained RPLs, if an error occurs anywhere in the chain, the RPLs following the one in error are made available without being processed and are posted complete with a feedback code of zero.

Manipulating Control Block Contents

VSAM provides a set of macros, GENCB, TESTCB, MODCB, and SHOWCB, to allow you to manipulate the contents of control blocks at execution time. You can use these macros to generate, test, modify, and display the contents of fields in the access method control block, the exit list, and the request parameter list. You don't have to know the format of the control block when you use these macros.

The GENCB, MODCB, TESTCB, and SHOWCB macros build a parameter list that describes, in codes, the actions indicated by the parameters you specify. The parameter list is passed to VSAM to take the indicated actions. An error can occur if you specify the parameters incorrectly.

Generating a Control Block

The GENCB macro can be used to generate an access method control block, an exit list, or a request parameter list when your program is executed. Generating the control block at execution time with GENCB has the advantage of requiring no reassembly of the program when you adopt a new version of VSAM in which control block formats might have changed. If you use the ACB, EXLST, and RPL macros to build control blocks, and then adopt a subsequent release of VSAM in which the control block format has changed, you will have to reassemble your program. GENCB also gives you the ability to generate multiple copies of the ACB, EXLST, or RPL to be used for concurrent requests. The disadvantage of using GENCB is that the path length is longer. It takes more instructions to build a control block using GENCB than to code the control block directly.

You may use the WAREA parameter to provide an area of storage in which to generate the control block. This work area has a 65K (X'FFFF') size limit. If you do not provide storage when you generate control blocks, the ACB, RPL, and EXLST will reside below 16 megabytes unless LOC=ANY is specified.

Testing Contents of Fields in the ACB, EXLST, and RPL

With the TESTCB macro, VSAM compares the contents of a field that you specify with a value that you specify. To show the result of this comparison, VSAM sets the condition code in the PSW (program status word). Only one keyword can be specified each time TESTCB is issued. This is useful to:

- Find out whether an action has been done by VSAM or your program (for example, opening a data set or activating an exit).
- Find out what kind of a data set is being processed in order to alter your program logic as a result of the test.

After issuing a TESTCB macro, you examine the PSW condition code. If the TESTCB is not successful, register 15 contains an error code and VSAM passes control to an error routine, if one has been specified. For a keyword specified as an option or a name, you test for an equal or unequal comparison; for a keyword specified as an address or a number, you test for an equal, unequal, high, low, not-high, or not-low condition.

VSAM compares A to B, where A is the contents of the field and B is the value to which it is to be compared. A low condition means, for example, that A is lower than B—that is, that the value in the control block is lower than the value you specified. If you specify a list of option codes for a keyword (for example, MACRF=(ADR,DIR)), each of them must equal the corresponding value in the control block for you to get an equal condition.

Some of the fields can be tested at any time: others, only after a data set is opened. The ones that can be tested only after a data set is opened can, in the case of a key-sequenced data set, pertain either to the data or to the index, as specified in the OBJECT parameter.

You can display fields using the SHOWCB maco at the same time that you test the fields.

Modifying Contents of the ACB, EXLST, and RPL

The MODCB macro allows you to customize the control blocks generated with the GENCB macro. The MODCB macro can be used to modify the contents of an access method control block, an exit list, or a request parameter list. Typical reasons to modify to a request parameter list are to change the indication of length of a record (RECLEN) when you're processing a data set whose records are not all the same length, and to change the type of request (OPTCD), such as from direct to sequential access or from full-key search argument to generic key search argument.

Displaying Contents of Fields in the ACB, EXLST, and RPL

The SHOWCB macro causes VSAM to move the contents of various fields in an access method control block, an exit list, or a request parameter list into your work area. You might want to learn the reason for an error or to collect statistics about a data set to allow your program to print a message or keep records of transactions.

Requesting Access to the Data Set

After your program is opened and a request parameter list is built, you can use the action request macros GET, PUT, ERASE, POINT, CHECK, and ENDREQ. Each request macro uses a request parameter list that defines the action to be taken. For example, when a GET macro points to a request parameter list that specifies synchronous, sequential retrieval, the next record in sequence is retrieved. When an ENDREQ macro points to a request parameter list, any current request (for example, a PUT) for that request parameter list ends immediately.

The action request macros allow you to do the following:

- Insert new records
- Retrieve existing records
- Point to existing records
- Update existing records
- Delete existing records
- Write buffers
- Retain buffers
- Perform multistring processing
- Perform concurrent requests
- Access records using a path
- Check for completion of asynchronous requests
- End request processing

Inserting a Record

Record insertions in VSAM data sets occur in several ways:

PUT RPL OPTCD=DIR,NSP
Allows you to insert records directly; VSAM remembers its position for subsequent sequential access.

PUT RPL OPTCD=DIR,NUP
Allows you to insert a record directly; VSAM does not remember its position.

PUT RPL OPTCD=SEQ,NUP or NSP
Allows you to insert records sequentially; VSAM remembers its position for subsequent sequential access.

PUT RPL OPTCD=SKP,NUP or NSP
Allows you to insert records in skip sequential order; VSAM remembers its position for subsequent sequential access.

Insertions into a Key-Sequenced Data Set

Insertions into a key-sequenced data set use the free space provided during the definition of the data set or the free space that develops as a result of control interval and control area splits. To create a data set or make mass insertions, use RPL OPTCD=SEQ,NUP,or NSP. This type of insertion uses the sequential insert strategy, and maintains free space during load mode and during mass insertions. All the other types use the direct insert strategy. If MACRF=SIS is specified in the ACB, all inserts use sequential insert strategy.

With addressed access of a key-sequenced data set, VSAM does not insert or add new records.

Sequential Insertion

- If the new record belongs after the last record of the control interval and the free space limit has not been reached, the new record goes into the existing control interval. If the control interval does not contain sufficient free space the new record is inserted into a new control interval without a true split.
- If the new record does not belong at the end of the control interval and there is free space in the control interval, it is placed in sequence into the existing control interval. If adequate free space does not exist in the control interval, a control interval split occurs at the point of insertion. The new record is inserted into the original control interval and the following records are inserted into a new control interval.

Mass Sequential Insertion

When VSAM detects that two or more records are to be inserted in sequence into a collating position (between two records) in a data set, VSAM uses a technique called mass sequential insertion to buffer the records being inserted. This reduces I/O operations. Using sequential instead of direct access in this case enables you to take advantage of this technique. You can also extend your data set (resume loading) by using sequential insertion to add records beyond the highest key or relative record number. There are possible restrictions to extending a data set into a new control area depending on the share options you specify. See Chapter 9, "Sharing a VSAM Data Set" on page 113.

Mass sequential insertion observes control interval and control area free space specifications when the new records are a logical extension of the control interval or control area (that is, when the new records are added beyond the highest key or relative record number used in the control interval or control area).

When several groups of records in sequence are to be mass inserted, each group may be preceded by a POINT with RPL OPTCD=KGE to establish positioning. KGE specifies that the key you provide for a search argument must be equal to the key or relative record number of a record.

Direct Insertion

A new record is inserted into an existing control interval if enough free space exists in the control interval. If there is not enough free space, and the insertion is not to the end of the control interval, the control interval is split, and approximately half of the records are moved to a new control interval. If there is still not enough free space, the control interval is split again, and approximately half of the records are moved to a new control interval. This process continues until there is enough free space to insert the new record.

Insertions into a Relative Record Data Set

Insertions into a relative record data set go into empty slots. When a record is inserted sequentially into a relative record data set, it is assigned the next relative record number in sequence. If the slot is not empty, VSAM sets an error return code, indicating a duplicate record. The assigned number is returned in the argument field of the RPL.

Direct or skip-sequential insertion of a record into a relative record data set causes the record to be placed as specified by the relative record number in the argument field of the RPL. You must insert the record into a slot that does not contain a record. If the slot specified does contain a record, VSAM sets an error return code in the RPL and rejects the request.

Insertions into an Entry-Sequenced Data Set

VSAM does not insert new records into an entry-sequenced data set. All records are added at the end of the data set.

Insertions into a Linear Data Set

Linear data sets cannot be processed at the record level. Use of the GET, PUT and POINT macros is not allowed at the record level. You must use the DIV macro to process a linear data set. See *Supervisor Services and Macro Instructions* for information on how to use Data-In-Virtual (DIV).

Retrieving Records

The GET macro is used to retrieve records. To retrieve records for update, use the GET macro with the PUT macro. You can retrieve records sequentially or directly. In either case, VSAM returns the length of the retrieved record to the RECLEN field of the RPL.

Sequential Retrieval

Records can be retrieved sequentially using keyed access or addressed access.

- **Keyed sequential retrieval.** The first time your program accesses a data set for keyed sequential access (RPL OPTCD=(KEY,SEQ)), VSAM is positioned at the first record in the data set in key sequence if nonshared resources are being used. With shared resources, you must use a POINT macro to establish position, or a direct request which retains position. A GET macro retrieves the record. The GET then positions VSAM at the next record in key sequence. VSAM checks positioning when processing modes are changed between requests.

  For keyed sequential retrieval of a relative record data set, the relative record number is treated as a full key. If a deleted record is encountered during sequential retrieval, it is skipped and the next record is retrieved. The relative record number of the retrieved record is returned in the argument field of the RPL.

- **Addressed sequential retrieval.** Retrieval by address is identical to retrieval by key, except that the search argument is an RBA, which must be matched to the RBA of a record in the data set. When a processing program opens a data set with nonshared resources for addressed access, VSAM is positioned at the record with RBA of zero to begin addressed sequential processing. A sequential GET request causes VSAM to retrieve the data record at which it is positioned and then positions VSAM at the next record. The address specified for a GET or a POINT must correspond to the beginning of a data record; otherwise the request is invalid. Spanned records stored in a key-sequenced data set cannot be retrieved using addressed retrieval.

GET-previous (backward-sequential) processing is a variation of normal keyed or addressed-sequential processing. Instead of retrieving the next record in ascending sequence (relative to current positioning in the data set), GET-previous processing retrieves the next record in descending sequence. To process records in descending sequence, specify BWD in the RPL OPTCD parameter. You can select GET-previous processing for POINT, GET, PUT (update only), and ERASE operations. The initial positioning by POINT, other than POINT LRD, requires that you specify a key. The following GET-previous processing does not need any specified key to retrieve the next record in ascending sequence.

GET-previous processing is not permitted with control interval or skip-sequential processing.

Pointing VSAM to a Record

You can use the POINT macro to begin retrieving records sequentially at a place other than the beginning of the data set. This macro places VSAM at the record with the specified key or relative byte address. However, it does not provide data access to the record. If you specify a generic key (a leading portion of the key field), the record pointed to is the first of the records having the same generic key. The POINT macro can position VSAM for either forward or backward processing, depending on whether FWD or BWD was specified in the RPL OPTCD parameter.

If, after positioning, you issue a direct request by way of the same request parameter list, VSAM drops positioning unless NSP or UPD was specified in the RPL OPTCD parameter.

When a POINT is followed by a VSAM GET/PUT request, both the POINT and the subsequent request must be in the same processing mode. For example, a POINT with RPL OPTCD=(KEY,SEQ,FWD) must be followed by GET/PUT with RPL OPTCD=(KEY,SEQ,FWD); otherwise, the GET/PUT request is rejected.

For skip-sequential retrieval, you must indicate the key of the next record to be retrieved. VSAM skips to the next record's index entry by using horizontal pointers in the sequence set to find the appropriate sequence-set index record and scan its entries. The key of the next record to be retrieved must always be higher in sequence than the key of the preceding record retrieved.

If your request fails, with an error code, positioning may not be maintained. To determine whether positioning is maintained in the case of a logical error, see "Macro Instruction Return Codes and Reason Codes" in *VSAM Administration: Macro Instruction Reference* Positioning is always released when you specify the ENDREQ macro.

Direct Retrieval

Records can also be retrieved directly using keyed access or addressed access.

- **Keyed direct retrieval** for a key-sequenced data set does not depend on prior positioning; VSAM searches the index from the highest level down to the sequence set to retrieve a record. You can specify the record to be retrieved by supplying one of the following:

  - The exact key of the record
  - An approximate key, less than or equal to the key field of the record
  - A generic key

  You can use an approximate specification when you do not know the exact key. If a record actually has the key specified, VSAM retrieves it; otherwise, it retrieves the record with the next higher key. Generic key specification for direct processing causes VSAM to retrieve the first record having that generic key. If you want to retrieve all the records with the generic key, specify RPL OPTCD=NSP in your direct request. That causes VSAM to position itself at the next record in key sequence. You can then retrieve the remaining records sequentially.

To use direct or skip-sequential access to process a relative record data set, you must supply the relative record number of the record you want in the argument field of the RPL macro. If you request a deleted record, the request will cause a no-record-found logical error.

A relative record data set has no index; VSAM takes the number of the record to be retrieved and calculates the control interval that contains it and its position within the control interval.

- **Addressed direct retrieval** requires that the RBA of each individual record be specified; previous positioning is not applicable.

With direct processing, you can optionally specify RPL OPTCD=NSP to indicate that position be maintained following the GET. Your program can then process the following records sequentially in either a forward or backward direction.

Updating a Record

The GET and PUT macros are used to update records. A GET for update retrieves the record and the following PUT for update stores the record that the GET retrieved.

When you update a record in a key-sequenced data set, you cannot alter the prime key field.

You can update the contents of a record with addressed access, but you cannot alter the record's length. To change the length of a record in an entry-sequenced data set, you must store it either at the end of the data set (as a new record) or in the place of an inactive record of the same length. You are responsible for marking the old version of the record as inactive.

Deleting a Record

After a GET for update retrieves a record, an ERASE macro instruction can delete the record. The ERASE macro can be used only with a key-sequenced data set or a relative record data set.

When you delete a record in a key-sequenced data set, the record is physically erased. The space the record occupied is then available as free space.

You can erase a record from the base cluster of a path only if the base cluster is a key-sequenced data set. If the alternate index is in the upgrade set (that is, UPGRADE was specified when the alternate index was defined), it is modified automatically when you erase a record. If the alternate key of the erased record is unique, the alternate index data record with that alternate key is also deleted.

When you erase a record from a relative record data set, the record is set to binary zeros and the control information for the record is updated to indicate an empty slot. You can reuse the slot by inserting another record of the same length into it.

With an entry-sequenced data set, you are responsible for marking a record you consider to be deleted. As far as VSAM is concerned, the record is not deleted. You can reuse the space occupied by a record marked as deleted by retrieving the record for update and storing in its place a new record of the same length.

Deferred and Forced Writing of Buffers

For integrity reasons, it is sometimes desirable to force the data buffer to be written after a PUT operation. At other times, it is desirable to defer the writing of a buffer as long as possible to improve performance. At the time the PUT is issued, if the RPL OPTCD specifies direct processing (DIR), and NSP is not specified, forced writing of the buffer occurs. Otherwise, writing is deferred. An ERASE request follows the same buffer writing rules as the PUT request. If LSR and GSR deferred writes are not specified, an ENDREQ macro always forces the current modified data buffer to be written.

Retaining Data Buffers and Positioning

Some operations retain positioning while others release it. In a similar way, some operations hold onto a buffer and others release it with its contents. Figure 17 shows which RPL options result in the *retention* of data buffers and positioning, and which options result in the *release* of data buffers and positioning:

	Buffer and Positioning	
RPL Options	**Retained**	**Released**
SEQ	•	
SKP	•	
DIR NSP	•	
DIR (no NSP)		•
DIR LOC	•	
UPD (with GET)	•	

Figure 17. Effect of RPL Options on Data Buffers and Positioning

Notes to Figure 17:

1. A sequential GET request for new control intervals releases the previous buffer.

2. The ENDREQ macro and the ERASE macro with RPL OPTCD=DIR releases data buffers and positioning.

3. Certain options that retain positioning and buffers upon normal completion may not do so if the request fails with an error code. To determine whether or not positioning is maintained in the case of a logical error, see "Macro Instruction Return Codes and Reason Codes" in *VSAM Administration: Macro Instruction Reference.*

The operation that uses but immediately releases a buffer and does not retain positioning is:

```
GET  RPL OPTCD=(DIR,NUP,MVE)
```

Multistring Processing

In multiple string processing, there may be multiple independent RPLs within a region for the same data set. The data set may 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 region 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.

If you are doing multiple string update processing, you must take into account VSAM look-aside processing and the rules surrounding exclusive use. Look-aside means VSAM checks its buffers to see if the control interval is already present when requesting an index or data control interval.

For GET nonupdate requests, an attempt is made to locate a buffer already in storage. As a result, a down-level copy of the data may be obtained either from buffers attached to this string or from secondary storage.

For GET to update requests, the buffer is obtained in exclusive control, and then read from the device for the latest copy of the data. If the buffer is already in exclusive control of another string, the request fails with an exclusive control feedback code. If you are using shared resources, the request may be queued, or may return an exclusive control error.

The exclusive use rules are as follows:

1. If a given string obtains a record with a GET for update request, the control interval is not available for update or insert processing by another string.
2. If a given string is in the process of a control area split caused by an update with length change or an insert, that string obtains exclusive control of the entire control area being split. Other strings cannot process insert or update requests against this control area until the split is complete.

If you are using nonshared resources, VSAM does not queue requests that have exclusive control conflicts, and you are required to clear the conflict. If a conflict is encountered, VSAM returns a logical error return code, and you must stop activity and clear the conflict. If the RPL that caused the conflict had exclusive control of a control interval from a previous request, you issue an ENDREQ before you attempt to clear the problem. You can clear the conflict in one of three ways:

- Queue until the RPL holding exclusive control of the control interval releases that control and then reissue the request

- Issue an ENDREQ against the RPL holding exclusive control to force it to release control immediately
- Use shared resources and issue MRKBFR MARK=RLS

Note: If your RPL has provided a correctly specified MSGAREA and MSGLGN, the address of the RPL holding exclusive control is provided in the first word of the MSGAREA. Your RPL field, RPLDDDD, contains the RBA of the requested control interval.

Concurrent Requests

With VSAM, you can maintain concurrent positioning for many requests to a data set. Thus you can process many portions of a data set concurrently, even within the same program, merely by providing many RPLs for that data set. The same ACB is used for all requests, and the data set needs to be opened only once. This means, for example, you could be processing a data set sequentially using one RPL, and at the same time, using another RPL, directly access selected records from the same data set.

For concurrent requests that require VSAM to keep track of more than one position in a data set, you can issue up to 255 request macros to process the data set at any one time.

For each request, a string defines the set of control blocks for the exclusive use of one user. If the number of strings you specify is not sufficient, the operating system dynamically extends the number of strings as needed by concurrent requests for the ACB. Strings allocated by dynamic string addition are not necessarily in contiguous storage. Dynamic string addition does not occur with LSR and GSR—it occurs only with NSR buffering.

Accessing Records Using a Path

When you are processing records sequentially using a path, records from the base cluster are returned according to ascending or, if you are retrieving the previous record, descending alternate key values. If there are several records with a non-unique alternate key, those records are returned in the order in which they were entered into the alternate index. VSAM sets a return code in the RPL when there is at least one more record with the same alternate key to be processed. For example, if there are three data records with the alternate key 1234, the return code would be set during the retrieval of records one and two and would be reset during retrieval of the third record.

When you use direct or skip-sequential access to process a path, a record from the base data set is returned according to the alternate key you have specified in the argument field of the RPL macro. If the alternate key is not unique, the record which was first entered with that alternate key is returned and a feedback code (duplicate key) is set in the RPL. To retrieve the remaining records with the same alternate key, specify RPL OPTCD=NSP when retrieving the first record with a direct request, and then switch to sequential processing.

You can insert and update data records in the base cluster using a path if:

- The PUT request does not result in non-unique alternate keys in an alternate index, which you have defined with the UNIQUEKEY attribute. However, if a nonunique alternate key is generated and you have specified the NONUNIQUEKEY attribute, updating can occur.

- You do not change the key of reference between the time the record was retrieved for update and the PUT is issued.
- You do not change the prime key.

When the alternate index is in the upgrade set, the alternate index is modified automatically by inserting or updating a data record in the base cluster. If the updating of the alternate index results in an alternate-index record with no pointers to the base cluster, the alternate-index record is erased.

Asynchronous Requests

In *synchronous mode*, VSAM does not return to your program from a PUT or GET operation until it has completed the operation. In *asynchronous mode*, VSAM returns control to your program *before* completing a PUT or a GET. This means that a program in asynchronous mode can perform other useful work while a VSAM PUT or GET is completed.

Asynchronous mode can improve throughput with direct processing because it permits processing to overlap with accesses from and to the direct access device. When reading records directly, each read often involves a seek on the direct access device, a relatively slow operation. In synchronous mode, this seek time does not overlap with other processing.

In order to specify asynchronous mode, you must specify OPTCD=ASY rather than OPTCD=SYN in the RPL.

Checking for Completion of Asynchronous Requests

Suppose your program is ready to process the next record, but VSAM is still trying to obtain that record. (The next record is not yet read in from the direct access device.) It may be necessary to stop execution of the program and wait for VSAM to complete reading in the record. The CHECK instruction stops execution of the program until the operation in progress is completed. You must issue a CHECK macro after each request for an RPL. If you attempt another request without an intervening CHECK, that request will be rejected.

Once the request is completed, CHECK releases control to the next instruction in your program, and frees up the RPL for use by another request.

Ending a Request

Suppose you determine, after initiating a request but before it is completed, that you don't want to complete the request. For example, suppose you determine during the processing immediately following a GET that you don't want the record you just requested. You can use the ENDREQ macro to cancel the request. ENDREQ has the following advantages:

- It allows you to avoid checking an unwanted asynchronous request.
- It writes any unwritten data or index buffers in use by the string.
- It cancels the VSAM positioning on the data set for the RPL.

Note: If you issue the ENDREQ macro, it is important that you check the ENDREQ return code to make sure it completes successfully. If an asynchronous request does not complete ENDREQ successfully, you must issue the CHECK macro. The data set cannot be closed until all asynchronous requests successfully complete either ENDREQ or CHECK.

Closing a Data Set

The CLOSE macro disconnects your program from the data set. It causes VSAM to do the following:

- Write any unwritten data or index records whose contents have changed.
- Update the catalog entry for the data set if necessary (if the location of the end-of-file indicator has changed, for example).
- Write SMF records if SMF is being used.
- Restore control blocks to the status they had before the data set was opened.
- Release virtual storage that was obtained during OPEN processing for additional VSAM control blocks and VSAM routines.

If a record management error occurs while CLOSE is flushing buffers, the data set's catalog information is not updated. The catalog may not properly reflect the data set's status and the index may not accurately reflect some of the data records. If the program enters an abnormal termination routine (abend), all open data sets are closed. The VSAM CLOSE invoked by abend does not update the data set's catalog information, it does not complete outstanding I/O requests, and buffers are not flushed. The catalog may not properly reflect the cluster's status, and the index may not accurately reference some of the data records. You can use the access method services VERIFY command to correct catalog information. The use of VERIFY is described in "Using VERIFY to Synchronize Values" on page 63.

When processing asynchronous VSAM requests, all strings must be quiesced by issuing the CHECK macro or the ENDREQ macro before issuing CLOSE or CLOSE TYPE=T (temporary CLOSE).

You can avoid an incomplete write to a direct access device by doing synchronous direct inserts or by using abnormal termination exits in which you issue a CLOSE or CLOSE TYPE=T to properly close the data set.

CLOSE TYPE=T causes VSAM to complete any outstanding I/O operations, update the catalog if necessary, and write any required SMF records. Processing can continue after a temporary CLOSE without issuing an OPEN macro.

If a VSAM data set is closed and CLOSE TYPE=T is not specified, you must reopen the data set before performing any additional processing on it.

When you issue a temporary or a permanent CLOSE macro, VSAM updates the data set's catalog records. If VSAM was invoked by an abend, the data set's catalog records are not updated, and contain inaccurate statistics.

It is the user's responsibility to ensure that shared DD statements are not dynamically unallocated until all ACBs sharing these DD statements have been closed. For an explanation of dynamic allocation, see *JCL User's Guide*.

VSAM Macro Relationships

At this point it is important to see how all of these macros work together in a program. Figure 18 shows the relationship between JCL and the VSAM macros in a program.

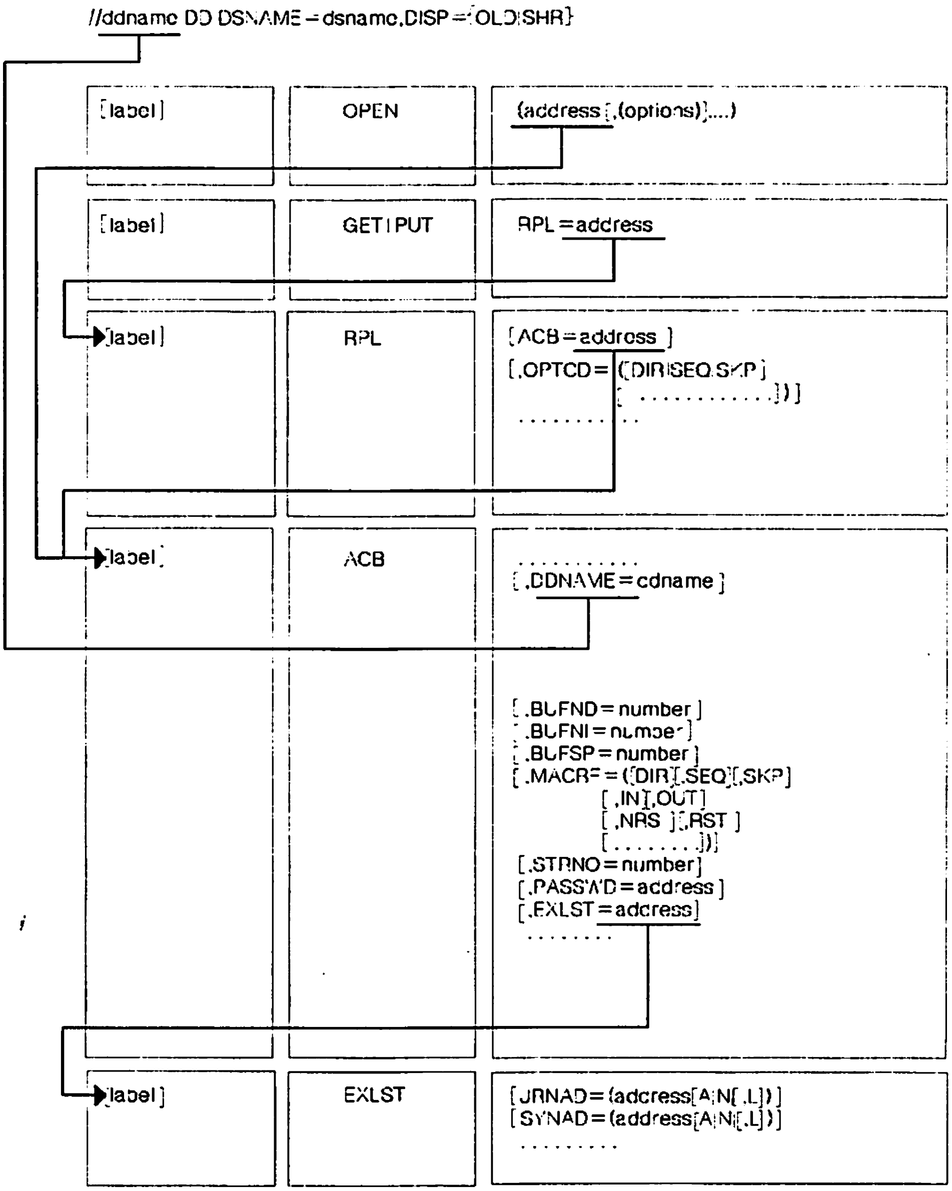


Figure 18. VSAM Macro Relations

Figure 19 on page 57 is a skeleton program that shows the relationship of VSAM macros to each other and to the rest of the program.

```
START  CSECT
         SAVE(14,12)                        Standard entry code
         .
         .
         .
         B      INIT                        Branch around file specs

MASACB   ACB    DDNAME=MASDS,AM=VSAM,       File specs                 X
                MACRF=(KEY,SEQ,OUT),                                   X
                EXLST=EXITS,                                           X
                RMODE31=ALL

MASRPL   RPL    ACB=MASACB,                                            X
                OPTCD=(KEY,SEQ,NUP,MVE,SYN),                           X
                AREA=WA,                                               X
                AREALEN=80,                                            X
                RECLEN=80

EXITS    EXLST  LERAD=LOGER,                                           X
                JRNAD=JOURN

TRANDCB  DCB    DDNAME=TRANDS,                                         X
                DSORG=PS,                                              X
                MACRF=GM,                                              X
                EODAD=EOTRF,                                           X
                LRECL=80,                                              X
                BLKSIZE=80,                                            X
                RECFM=F
INIT     .                                  Program initialization
         .
         .
         OPEN   (MASACB,,TRANDCB)           Connect data sets
         .
         .
         .
         GET    TRANDCB,WA                  Processing loop
         .
         .
         .
         PUT    RPL=MASRPL
         .
         .
         .
EOTRF    CLOSE  (MASACB,,TRANDCB)           Disconnect data sets
         .
         .
         .
         RETURN (14,12)                     Return to calling routine
LOGER    .Exit routines
         .
JOURN    .
         .
WA       DS     CL80                        Work area
         END
```

Figure 19. Skeleton VSAM Program

Operating in SRB or Cross Memory Mode

Operating in service request block (SRB) or cross memory mode allows you to use structures in other memories to increase the amount of space available. SRB and cross memory mode are privileged modes of operation reserved for authorized users. Cross memory is a complex concept, and there are a number of warnings and restrictions associated with it. For information, see *System Macros and Facilities*.

You can only operate in cross memory or SRB mode for synchronous, supervisor state requests. An attempt to invoke VSAM asynchronously in either mode results in a logical error. Also, VSAM makes no attempt to synchronize cross memory mode requests. This means that the RPL must specify WAITX, and a UPAD exit (user processing exit routine) must be provided in an exit list to handle cross memory request synchronization. You must provide an UPAD routine that ensures the ECB is posted before returning to VSAM.

In order to function in cross memory or SRB mode, record management does not issue any supervisor call instructions (SVCs) or take any user exits as they were entered. Instead of issuing SVCs, RPL return codes are set to indicate that an SVC (such as an end of volume) is required, in order to complete the request.

Whenever VSAM cannot avoid the SVC, it sets an RPL return code to indicate that you must change processing mode so that you are running under a TCB (task control block) in the address space in which the data set was opened. You cannot be in cross memory mode. You may then reissue the request to allow the SVC to be issued by VSAM. The requirement for VSAM to issue an SVC is kept to a minimum. Areas identified as requiring a TCB not in cross memory mode are EXCEPTIONEXIT, loaded exits, EOV (end of volume), dynamic string addition, AIX (alternate index) processing, and MSS-related macros.

If a logical error or an end-of-data condition occurs during cross memory or SRB processing, VSAM attempts to enter the LERAD (logical error exit) or EODAD (end of data exit) routine. If the routine must be loaded, it cannot be taken; VSAM sets the RPL feedback to indicate "invalid TCB." If an I/O error occurs during cross memory or SRB processing and an EXCEPTIONEXIT or loaded SYNAD (physical error exit) routine is specified, these routines cannot be taken; the RPL feedback indicates an I/O error condition.

See *Data Facility Product: Customization* for more information on exit routines.

End of General-Use Programming Interface

Chapter 5. Establishing Backup and Recovery Procedures

It is important to establish backup and recovery procedures for your data sets so that you can replace a destroyed or damaged data set with its backup copy. There are several methods of backing up and recovering VSAM data sets:

- Using the access method services REPRO command.
- Using the access method services EXPORT and IMPORT commands.
- Writing your own program for backup and recovery.
- Using the Data Facility Data Set Services (DFDSS) DUMP and RESTORE commands. This option is available only if DFDSS Release 2 or later is installed on your system, and if your data sets are cataloged in an integrated catalog facility catalog.

Each of these methods of backup and recovery has its advantages. You will need to decide which method is best for the particular data that you need to back up.

Using REPRO for Backup and Recovery

The REPRO command is used to create a duplicate VSAM data set for backup. Using REPRO for backup and recovery has the following advantages:

- **Backup Copy Is Accessible.** The backup copy obtained by using REPRO is accessible for processing. It can be a VSAM data set or a sequential (SAM) data set.
- **Type of Data Set Can Be Changed.** The backup copy obtained by using REPRO can be a different type of VSAM data set than the original. For example, you could back up a VSAM key-sequenced data set by copying it to a VSAM entry-sequenced data set.
- **Data Is Reorganized.** Using REPRO for backup results in data reorganization and the re-creation of an index for a key-sequenced data set. The data records are rearranged physically in ascending key sequence (control interval and control area splits may have placed them physically out of order) and free-space quantities are restored. Because the data is reorganized, you must remember that any absolute references by way of RBA become invalid.

Options Using REPRO

REPRO provides you with several options for creating backup copies and using them for data set recovery. The following are suggested ways to use REPRO.

1. Create a backup copy on another catalog, then use the backup copy to replace the original.
   a. To create a backup copy, define a data set on another catalog, and use REPRO to copy the original data set into the new data set you have defined. Because the two data sets are defined on separate catalogs, they may have the same name.
   b. Because a backup copy created by REPRO is accessible for processing, when you want to replace the original with the backup copy, you can

leave the backup copy on the catalog it was copied to. If you do this, the JOBCAT and STEPCAT statements in the JCL must be changed to reflect the name of the catalog containing the backup copy.

2. Create a copy of a nonreusable data set on the same catalog, then delete the original data set, define a new data set, and load the backup copy into the newly defined data set.

   a. To create a backup copy, define a data set, and use REPRO to copy the original data set into the newly defined data set. If you define the backup data set on the same catalog as the original data set, the backup data set must have a different name.

   b. To recover the data set, delete the original data set if it still exists, using the DELETE command. Next, redefine the data set using the DEFINE command, then restore it with the backup copy using the REPRO command.

3. Create a copy of a reusable data set, then load the backup copy into the original data set. When using REPRO, the REUSE attribute allows repeated backups to the same VSAM reusable target data set.

   a. To create a backup copy, define a data set, and use REPRO to copy the original reusable data set into the newly defined data set.

   b. To recover the data set, load the backup copy into the original reusable data set.

4. Create a backup copy of a data set, then merge the backup copy with the damaged data set. When using REPRO, the REPLACE parameter allows you to merge a backup copy into the damaged data set. This option is not available for entry-sequenced data sets, since records are always added to the end of an entry-sequenced data set.

   a. To create a backup copy, define a data set, and use REPRO to copy the original data set into the newly defined data set.

   b. To recover the data set, use the REPRO command with the REPLACE parameter to merge the backup copy with the destroyed data set. With a key-sequenced data set, each source record whose key matches a target record's key replaces the target record. Otherwise, the source record is inserted into its appropriate place in the target cluster. With a relative record data set, each source record, whose relative record number identifies a data record in the target data set, replaces the target record. Otherwise, the source record is inserted into the empty slot its relative record number identifies.

      When only part of a data set is damaged, you can replace only the records in the damaged part of the data set. The REPRO command allows you to specify a location at which copying is to begin and a location at which copying is to end.

Using EXPORT/IMPORT for Backup and Recovery

Using EXPORT/IMPORT for backup and recovery has the following advantages:

- **Data Is Reorganized.** Using EXPORT for backup results in data reorganization and the re-creation of an index for a key-sequenced data set. The data records are rearranged physically in ascending key sequence (control interval and control area splits may have placed them physically out of order) and free-space quantities are balanced. Because the data is reorganized, you must remember that any absolute references by way of RBA become invalid.
- **Redefinition Is Easy.** Because most catalog information is exported along with the data set, you are not required to define a data set before importing the copy. The IMPORT command deletes the original copy, defines the new object, and copies the data from the exported copy into the newly defined data set.
- **Attributes Can Be Changed or Added.** When you IMPORT a data set for recovery, you can specify the OBJECTS parameter to indicate new or changed attributes for the data set. This allows you to change the name of the data set, the key ranges, and the volumes on which the data set is to reside.

Structure of an Exported Data Set

An exported data set is an unloaded copy of the data set. The backup copy can only be a sequential (SAM) data set.

Most catalog information is exported along with the data set, easing the problem of redefinition. The backup copy contains all of the information necessary to redefine the VSAM cluster or alternate index when you IMPORT the copy.

Procedure for Using EXPORT/IMPORT

When you export a copy of a data set for backup, specify the TEMPORARY attribute. This indicates that the data set is not to be deleted from the original system.

You can export ESDS or LDS base clusters in control interval by specifying the CIMODE parameter. When CIMODE is forced for an LDS, a RECORDMODE specification is overridden.

Use the IMPORT command to totally replace a VSAM cluster whose backup copy was built using the EXPORT command. The IMPORT command uses the backup copy to replace the cluster's contents and catalog information.

You can protect an exported data set by specifying the INHIBITSOURCE or INHIBITTARGET parameters. These parameters indicate that the source or target data set cannot be accessed for any operation other than retrieval.

Data Facility Data Set Services (DFDSS)

The IBM Data Facility Data Set Services, program product (5740-UT3) Release 2 or later, can be used to back up and recover VSAM data sets cataloged in an integrated catalog facility catalog. With the DUMP operation, you can dump DASD data to a sequential data set. With the RESTORE operation, you can restore a data set to a DASD volume from a DFDSS-produced dump volume.

The DUMP procedure produces an image copy of the data set; the data set is not reorganized when it is restored.

For more information on using DFDSS for backup and recovery, refer to *DFDSS: User's Guide and Reference*.

Writing Your Own Program for Backup and Recovery

There are two methods of creating your own program for backup and recovery.

- If you periodically process a data set sequentially, you can easily create a backup copy as a by-product of normal processing. This backup copy can be used like one made by REPRO.
- You can write your own program to back up your data sets. Whenever possible, this program should be integrated into the regular processing procedures.

  The JRNAD exit routine is one way to write your own backup program. When you request a record for update, your program can call the JRNAD exit routine to copy the record you are going to update, and write it to a different data set. When you return to VSAM, VSAM completes the requested update. If something goes wrong, you have a backup copy. For more information on the JRNAD exit routine, see *Data Facility Product: Customization*.

Updating after Data Set Recovery

After replacing a damaged data set with its backup copy, rerun the jobs that updated the original between the time it was backed up and the time it became inaccessible. This updates the backup copy.

Synchronizing Data Set and Catalog Information

Because the physical and logical description of a data set is contained in its catalog entries, VSAM requires up-to-date catalog entries to access data sets.

Synchronizing Values following Data Set or Catalog Damage

If either your data set or your catalog is damaged, your recovery procedure must match both data set and catalog entry status. Recovery by way of reloading the data set automatically takes care of this problem. A new catalog entry is built when the data set is reloaded.

Backing up the data sets in a user catalog allows you to recover from damage to the catalog. You can import the backup copy of a data set whose entry is lost or redefine the entry and reload the backup copy.

If the catalog is completely lost, you can redefine it, then import or redefine and reload all the data sets that were defined in the catalog.

Synchronizing Values following an Abnormal Termination

When a data set is closed, its end-of-data and end-of-key-range information is used to update the data set's cataloged information. If a system failure occurs before the data set is closed (before the user's program issues CLOSE), the data set's cataloged information 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.

Using VERIFY to Synchronize Values

The VERIFY command is used to compare the end-of-data and end-of-key range information in a catalog with the true end-of-data and end-of-key range. If the information in the catalog does not agree with the true end-of-data or end-of-key range, the catalog information is corrected.

The VERIFY command should be used following a system failure that caused a component opened for update processing to be improperly closed. Clusters, alternate indexes, and catalogs can be verified. Paths over an alternate index cannot be verified. Paths defined directly over a base cluster can be verified. 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. Therefore, use the cluster or alternate index name as the target of your VERIFY command.

To use the VERIFY command to verify a catalog, access method services must be authorized. For information about program authorization, see "Using the Authorized Program Facility (APF)" in *System Macros and Facilities.*

You cannot use VERIFY to correct catalog records for a key-sequenced data set or a relative record data set after load mode failure. An entry-sequenced data set defined with the RECOVERY attribute may be verified after a create (load) mode failure; however, you cannot run VERIFY against an empty data set or a linear data set. Any attempt to do either will result in a VSAM logical error. See "Opening a Data Set" on page 41 for information about VSAM issuing the implicit VERIFY command.

Chapter 6. Optimizing VSAM Performance

This chapter describes many of the options and factors that either influence or, in some cases, determine VSAM's performance as well as the performance of the operating system. The main topics include control interval and control area size, free space, key ranges, buffer management, index options, and staging VSAM data sets for MSS.

Most of the options are specified in the access method services DEFINE command when a data set is created. In some cases, options can be specified in the ACB and GENCB macro instructions and in the DD AMP parameter.

Optimizing Control Interval Size

You can let VSAM select the size of a control interval for a data set, or you can request a particular control interval size in the DEFINE command. You may be able to improve VSAM's performance by specifying a control interval size in the DEFINE command, depending on the particular storage and access requirements for your data set. For information on the structure and contents of control intervals, see "Control Intervals" on page 3.

Control interval size affects record processing speed and storage requirements in these ways:

- **Buffer space.** Data sets with large control interval sizes require more buffer space in virtual storage. For information on how much buffer space is required, see "Determining I/O Buffer Space for Nonshared Resources" on page 76.
- **I/O operations.** Data sets with large control interval sizes require fewer I/O operations to bring a given number of records into virtual storage; fewer index records must be read. This is significant primarily for sequential and skip-sequential access. Large control intervals are not beneficial for keyed direct processing of a key-sequenced data set.
- **Free space.** Free space is used more efficiently (fewer control interval splits and less wasted space) as control interval size increases relative to data record size. For more information on efficient use of free space, see "Optimizing Free Space Distribution" on page 70.

Control Interval Size Limitations

When you request a control interval size, you must take into account the length of your records and whether or not the SPANNED attribute has been specified.

The valid control interval sizes are from 512 to 8192 bytes in increments of 512 bytes and from 8K to 32K bytes in increments of 2K bytes for objects cataloged in integrated catalog facility catalogs and for the integrated catalog facility catalogs themselves. The valid sizes for index components cataloged in VSAM catalogs are 512, 1024, 2048 and 4096 bytes.

Unless the data set was defined with the SPANNED attribute, the control interval must be large enough to hold a data record of the maximum size specified in the RECORDSIZE parameter. Because the minimum amount of control information in a control interval is 7 bytes, a control interval is normally at least 7 bytes larger than the largest record in the component. If the control interval size you specify is not large enough to hold the maximum size record, VSAM increases the control interval size to a multiple of the minimum physical block size. The control interval size VSAM provides is large enough to contain the record plus the overhead.

The use of the SPANNED attribute removes this constraint by allowing data records to be continued across control intervals. The maximum record size is then equal to the number of control intervals per control area multiplied by control interval size minus 10. The use of the SPANNED attribute places certain restrictions on the processing options that can be used with a data set. For example, records of a data set with the SPANNED attribute cannot be read or written in locate mode. For more information on spanned records, see "Spanned Records" on page 6.

Control interval size is limited by the requirement that it be a whole number of physical blocks. The information recorded on a DASD track is divided into physical blocks. Block sizes the same as the supported control interval sizes are supported for objects cataloged in integrated catalog facility catalogs as well as the integrated catalog facility catalogs themselves. Block sizes of 512, 1024, 2048 and 4096 bytes are supported for index components cataloged in VSAM catalogs. If you specify a control interval that is not a proper multiple for the supported block size, VSAM increases it to the next multiple. For example, 2050 is increased to 2560.

Physical Block Size and Track Capacity

Figure 20 illustrates the relationship between control interval size, physical block size, and track capacity. The information on a track is divided into physical blocks. Control interval size must be a whole number of physical blocks. Control intervals may span tracks. However, poor performance results if a control interval spans a cylinder boundary, because the read/write head must move between cylinders.

<table>
<tr><td colspan="4">CI1</td><td colspan="4">CI2</td><td colspan="4">CI3</td></tr>
<tr><td>PB</td><td>PB</td><td>PB</td><td>PB</td><td>PB</td><td>PB</td><td>PB</td><td>PB</td><td>PB</td><td>PB</td><td>PB</td><td>PB</td></tr>
<tr><td colspan="3">Track 1</td><td colspan="3">Track 2</td><td colspan="3">Track 3</td><td colspan="3">Track 4</td></tr>
</table>

PB - Physical Block

Figure 20. Control Interval Size, Physical Block Size, and Track Capacity

The physical block size is always selected by VSAM. VSAM chooses the largest physical block size that is an integral multiple of the control interval size. The block size is also based on device characteristics.

Data Control Interval Size

You can either specify a data control interval size or default to a system-calculated control interval size. If you don't specify a size, the system calculates a default value that will best use the space on the track for the average record size of spanned records or the maximum record size of nonspanned records.

If you specify control interval size at the cluster level only, the size applies to the data component, and VSAM calculates the index control interval size.

Normally, a 4096-byte data control interval is reasonably good regardless of the DASD device used, processing patterns, or the processor. A linear data set, in fact, requires a 4096-byte control interval. However, there are some special considerations that might affect this choice:

- If you have very large control intervals, more pages are required to be fixed during I/O operations. This could adversely affect the operation of the system.
- Small records in a data control interval can result in a large amount of control information. Often free space cannot be used.
- The type of processing you use may also affect your choice of control interval size.
  - *Direct processing.* When direct processing is predominant, a small control interval is preferable, because you are only retrieving one record at a time. In general, select the smallest data control interval that yields a reasonable space utilization.
  - *Sequential processing.* When sequential processing is predominant, larger data control intervals may be good choices. For example, given a 16K-byte data buffer space, it is better to read two 8K-byte control intervals with one I/O operation than four 4K-byte control intervals with two I/O operations.
  - *Mixed processing.* If the processing is a mixture of direct and sequential, a small data control interval with multiple buffers for sequential processing may be a good choice.

If you specify free space for a key-sequenced data set, the system determines the number of bytes to be reserved for free space. For example, if control interval size is 4096, and the percentage of free space in a control interval has been defined as 20%, 819 bytes are reserved.

To find out what values are actually set in a defined data set, you can issue the access method services LISTCAT command.

Index Control Interval Size

For a key sequenced data set, you can either specify an index control interval size or default to a system calculated size. If you don't specify a size, VSAM uses 512. After VSAM determines the number of control intervals in a control area (see "How VSAM Adjusts Control Interval Size" on page 68), it estimates whether an index record is large enough to handle all the control intervals in a control area. If the index record is too small, either the size of an index control interval is increased or the size of the control area is reduced by decreasing the number of control intervals.

The size of the index control interval must be compatible with the size of the data control interval. If you select too small a data control interval size, the number of data control intervals in a control area may be large enough to cause the index control interval size to exceed the maximum. VSAM first tries to increase the index control interval size and, if not possible, then starts decreasing the number of control intervals per control area. If this does not solve the problem, the DEFINE fails.

A 512-byte index control interval is normally chosen, because this is the smallest amount of data that can be transferred to virtual storage using the least amount of device and channel resources. A larger control interval size may be needed, depending on the allocation unit, the data control interval size, the key length, and the key content as it affects compression. (It is rare to have the entire key represented in the index, because of key compression.) A catalog listing will show the number of control intervals in a control area. This number plus the key length, can be used to estimate the size of index record necessary to avoid control area splits, which occur when the index control interval size is too small. To make a general estimate of the index control interval size, you can multiply one half of the key length by the number of data control intervals per control area.

How VSAM Adjusts Control Interval Size

The control interval sizes you specify when the data set is defined are not necessarily the ones that appear in the catalog. VSAM makes adjustments, if possible, so that control interval size conforms to proper size limits, minimum buffer space, adequate index-to-data size, and record size. This happens when your data set is defined.

For example:

1. You specify data and index control interval size. After VSAM determines the number of control intervals in a control area, it estimates whether one index record is large enough to handle all control intervals in the control area. If not, the size of the index control interval is increased, if possible. If the size cannot be increased, VSAM decreases the number of control intervals in the control area.
2. You specify maximum record size as 2560 and data control interval size as 2560, and have no spanned records. VSAM adjusts the data control interval size to 3072 to allow space for control information in the data control interval.
3. You specify buffer space as 4K, index control interval size as 512, and data control interval size as 2K. VSAM decreases the data control interval to 1536. Buffer space must include space for two data control intervals and one index control interval at DEFINE time. For more information on buffer space requirements, see "Determining I/O Buffer Space for Nonshared Resources" on page 76.

Optimizing Control Area Size

There is no way to explicitly specify control area size. The control area size is determined on the basis of the space allocation requested. For a key-sequenced data set, data and index control interval size, and the amount of buffer space available also affect control area size. When extents are allocated, they are composed of a whole number of control areas. By calculating the size of a control area as it does, VSAM is able to meet the primary and secondary space requirements without overcommitting space for the data set. For information on the structure and contents of control areas, see "Control Areas" on page 6.

A control area is never larger than one cylinder. If you allocate space in a DEFINE command using the CYLINDERS parameter, VSAM sets the control area size to one cylinder. If you allocate space in tracks or records, VSAM checks the smaller of the primary and secondary space allocation against the specified device's cylinder size. If the smaller space allocation is less than the device's cylinder size, the size of the control area is set equal to the smaller space quantity. If the smaller quantity is greater than or equal to the device's cylinder size, the control area size is set equal to cylinder size.

If the control area is smaller than a cylinder, its size will be an integral multiple of tracks, and it can span cylinders. However, a control area can never span an extent of a data set, which is always composed of a whole number of control areas. For more information on allocating space for a data set, see "Allocating Space for a Data Set" on page 23.

Advantages of a Large Control Area

Control area size has significant performance implications. One-cylinder control areas have the following advantages:

- There is a smaller probability of control area splits.
- The index is more consolidated. One index record addresses all the control intervals in a control area. If the control area is large, fewer index records and index levels are required. For sequential access, a large control area decreases the number of reads of index records.
- There are fewer sequence set records. The sequence set record for a control area is always read for you. Fewer records means less time spent reading them.
- If the sequence set of the index is embedded on the first track of the control area to reduce disk-arm movement and rotational delay, a large control area reduces the storage space needed for sequence sets.

  For example, the IMBED option requires one track per control area for sequence set information. If the control area on a 3380 is three tracks, 1/3 of the DASD space is required for sequence sets. If the control area on a 3380 is one cylinder, 1/15 the DASD space is required.
- If you have allocated enough buffers, a large control area allows you to read more buffers into storage at one time. This is useful if you are accessing records sequentially.

Disadvantages of a Large Control Area Size

The following disadvantages of a one-cylinder control area must also be considered:

- If there is a control area split, more data is moved.
- During sequential I/O, a large control area ties up more real storage and more buffers.

Optimizing Free Space Distribution

With the DEFINE command, you can specify the percentage of free space in each control interval and the percentage of free control intervals per control area. Free space improves performance by reducing the likelihood of control interval and control area splits. This, in turn, reduces the likelihood of VSAM moving a set of records to a different cylinder away from other records in the key sequence. When there is a direct insert or a mass sequential insert that does not result in a split, VSAM inserts the records into available free space.

The amount of free space you need depends on the number and location of records to be inserted, lengthened, or deleted. Too much free space may result in the following:

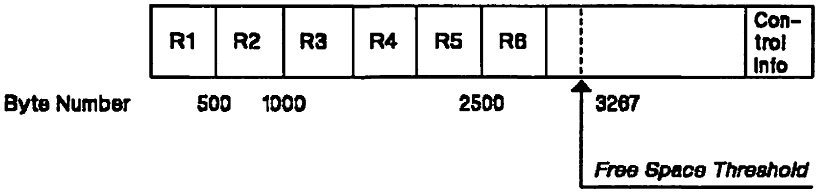
- Increased number of index levels, which affects run times for direct processing.
- More direct access storage required to contain the data set.
- More I/O operations required to sequentially process the same number of records.

Too little free space may result in an excessive number of control interval and control area splits. These splits are time consuming, and have the following additional effects:

- More time is required for sequential processing because the data set is not in physical sequence.
- More seek time is required during processing because of control area splits.

Consider using LISTCAT or the ACB JRNAD exit to monitor control area splits. When splits become prevalent, you can reorganize the data set. For information about the JRNAD exit, refer to *Data Facility Product: Customization.*

Figure 21 illustrates how free space is determined for a control interval.



```
FREESPACE(20 10)
CONTROLINTERVALSIZE(4096)
RECORDSIZE(500 500)
```

Figure 21. Determining Free Space

For this data set, each control interval is 4096 bytes. In each control interval, 10 bytes are reserved for control information. Because control interval free space is specified as 20%, 819 bytes are reserved as free space. (4096 * .20 = 819). This puts the *free space threshold* at 3267 bytes. The space between the threshold and the control information is reserved as free space.

Because the records loaded in the data set are 500-byte records, there is not enough space for another record between byte 3000 and the free space threshold at byte 3267. These 267 bytes of unused space are also used as free space. This leaves 1086 bytes of free space; enough to insert two five hundred byte records. Only 86 bytes are left unusable.

When you specify free space, ensure that the percentages of free space you specify yield full records and full control intervals with a minimum amount of unusable space.

Choosing the Optimal Percentage of Free Space

The amount of control interval free space you specify should be consistent with expected record insertion activity. Determine the amount of 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 control areas are created to provide room for additional insertions.

  If the few records to be added are fairly evenly distributed, control interval 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, control area 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 control intervals 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. For more information on mass insertion, refer to "Inserting a Record" on page 46.
- **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 parts of the data set that will receive additions. The example in "Altering the Free Space Specification When Loading a Data Set" demonstrates this.

Altering the Free Space Specification When Loading a Data Set

The following example uses the ALTER command to change the FREESPACE specification when loading a data set.

Assume that a large key-sequenced data set is to contain records with keys from 1 through 300000. It is expected to have no inserts in key range 1 through 100000, some inserts in key range 100001 through 200000, and heavy inserts in key range 200001 through 300000.

An ideal data structure at loading time would be:

Key Range	**Free Space**
1 through 100000	none
100001 through 200000	5% control area
200001 through 300000	5% control interval and 20% control area

This structure can be built as follows:

1. DEFINE CLUSTER with FREESPACE (0 0), or without any FREESPACE parameter.
2. Load records 1 through 100000 with REPRO or any user program using a sequential insertion technique.
3. CLOSE the data set.
4. Change the FREESPACE value of the cluster with the access method services command ALTER *clustername* FREESPACE (0 5).
5. Load records 100001 through 200000 with REPRO or any user program using a sequential insertion technique.
6. CLOSE the data set.
7. Change the FREESPACE value of the cluster with the access method services command ALTER *clustername* FREESPACE (5 20).
8. Load records 200001 through 300000 with REPRO or any user program using a sequential insertion technique.

This procedure has the following advantages:

- It prevents wasting space. For example, if FREESPACE (0 10) were defined for the whole data set, the free space in the first key range would all be wasted.

- It minimizes control interval and control area splits. If FREESPACE (0 0) were defined for the whole data set, there would be a very large number of control interval and control area splits for the first inserts.

Key Ranges

With a key-sequenced data set, you may specify specific key ranges to be stored on specific volumes. For example, if you have three volumes, you might assign records with keys A through E to the first volume, F through M to the second, and N through Z to the third. Key range data sets cannot be defined as reusable.

All the volumes specified for the cluster's data component or for the index component must be of the same device type. However, the data component and the index component can be on different device types. When you define a key range data set, all volumes to contain key ranges must be mounted.

When you define a key range data set, the primary space allocation is acquired from each key range. Secondary allocations are supplied as the data set is extended within a particular key range.

Using key ranges can optimize performance in the following ways:

- To access records in a specific key range, you only have to mount the volume containing that key range, if you assign data to various volumes according to ranges of key values.
- Having a primary space allocation for each key range in a data set provides an implied amount of free space in each portion of the data set.

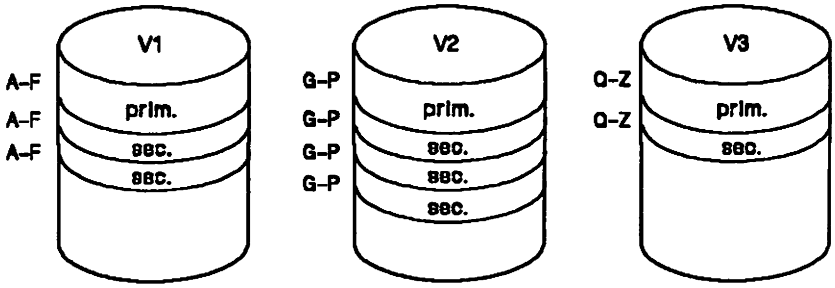
Examples of How Key Ranges Are Assigned

The following examples demonstrate how key ranges are assigned for VSAM data sets defined in integrated catalog facility catalogs.

Example 1

This example illustrates how space is allocated for a key range data set with three key ranges on three volumes.

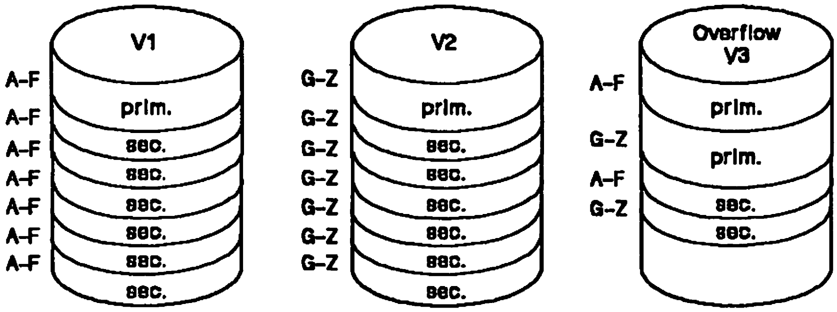
```
DEFINE CLUSTER ... VOLUMES (V1,V2,V3)   -
       KEYRANGES ((A,F) (G,P) (Q,Z))   -
       CYL (100,50)
```



Example 2

This example illustrates how space is allocated for a key range data set with two key ranges on three volumes. When there are more volumes than there are key ranges, the excess volumes are marked as candidates and are not required to be mounted. The candidate volumes are used for overflow records from any key range.

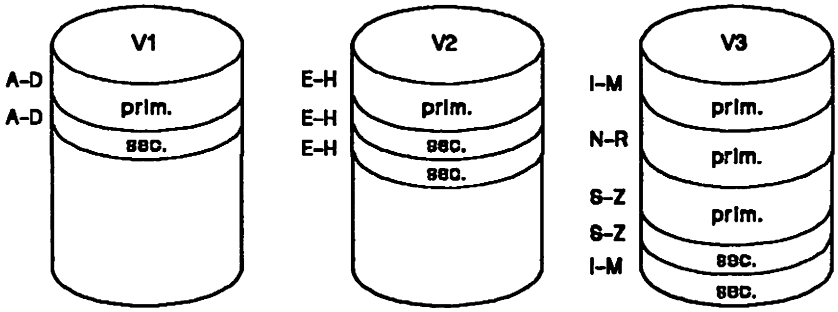
```
DEFINE CLUSTER ... VOLUMES (V1,V2,V3)   -
       KEYRANGES ((A,F) (G,Z))   -
       CYL (100,50)
```



Example 3

This example illustrates how space is allocated for a key range data set with five key ranges on three volumes.

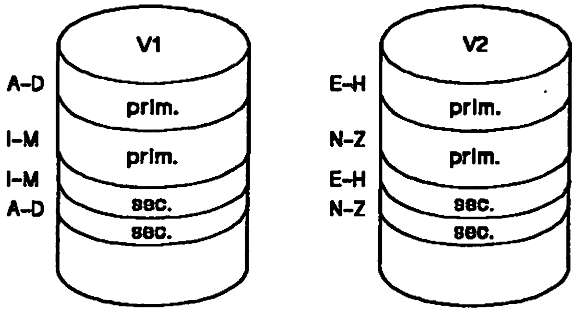
```
DEFINE CLUSTER ... VOLUMES (V1,V2,V3)   -
       KEYRANGES ((A,D) (E,H) (I,M) (N,R) (S,Z))   -
       CYL (100,50)
```



Example 4

This example illustrates how space is allocated for a key range data set with four key ranges on two volumes when each volume is specified twice. When a volume serial number is duplicated in the VOLUMES parameter, more than one key range is allocated space on the volume.

```
DEFINE CLUSTER ... VOLUMES (V1,V2,V1,V2)   -
      KEYRANGES ((A,D) (E,H) (I,M) (N,Z))   -
      CYL (100,50)
```



Naming Key Ranges

If you name the data component of a cluster, a generated name is used for each additional key range on a volume. If you do not name the data component, then all names on all volumes are generated. The names of user-named data components appear once on each volume in the Format-1 DSCB. Generated names are used on all volumes for the Format-1 DSCBs, but each name has a unique key range qualifier so that you can correlate the Format-1 DSCB and the key range.

Format-1 DSCB names are used for data sets defined in an integrated catalog facility catalog or defined with the UNIQUE attribute in a VSAM catalog.

For a multivolume key range cluster, the name specified on the data component is used for the first key range on each volume. If more than one key range resides on a volume, a special key range qualifier is appended to a generated name.

A key range qualifier is a unique qualifier appended to the generated name to form a unique name for a particular key range. A key range qualifier is 4-characters, starting with an alphabetic "A" followed by three digits that start at 001. The first generated key range name would therefore be: **CLUSTER/AIX high-level qualifier.Taaaaaaa.VDDyyddd.Tbbbbbbb.A001.**

If the data component name you specify is longer than 39 characters, the first 37 characters are joined to the last 2 characters, and the 4-character key range qualifier is appended.

If a duplicate generated name is found on a volume, the key range qualifier starts with the letter "B" or "C" and so on, until a unique name is found.

Determining I/O Buffer Space for Nonshared Resources

I/O buffers are used by VSAM to read and write control intervals from DASD to virtual storage. VSAM requires a minimum of three buffers, two for data control intervals and one for an index control interval. (One of the data buffers is used only for formatting control areas and splitting control intervals and control areas.) The VSAM default is enough space for these three buffers.

To increase performance, there are parameters to override the VSAM default values. There are three places where these parameters may be specified:

- BUFFERSPACE, specified in the access method services DEFINE command. This is the least amount of storage ever provided for I/O buffers.
- BUFSP, BUFNI, and BUFND, specified in the VSAM ACB macro instruction, is the maximum amount of storage to be used for a data set's I/O buffers. If the value specified in the ACB macro is greater than the value specified in DEFINE, the ACB value overrides the DEFINE value.
- BUFSP, BUFNI, and BUFND, specified in the JCL DD AMP parameter, is the maximum amount of storage to be used for a data set's I/O buffers. A value specified in JCL overrides DEFINE and ACB values if it is greater than the value specified in DEFINE.

VSAM must always have sufficient space available to process the data set as directed by the specified processing options.

Obtaining Buffers above 16 Megabytes

To increase the storage area available below 16 megabytes for your problem program, you can request VSAM data buffers and VSAM control blocks from virtual storage above 16 megabytes. To do this, specify the RMODE31 parameter on the ACB macro. See Appendix A, "Using 31-Bit Support" on page 147 for more information.

Buffer Allocation for Concurrent Data Set Positioning

To calculate the number of buffers you need, you must determine the number of *strings* you will use. A string is a request to a VSAM data set requiring data set positioning. If different concurrent accesses to the same data set are necessary, *multiple strings* are used. If multiple strings are used, each string requires exclusive control of an index I/O buffer. Therefore, the value specified for the STRNO parameter (in the ACB or GENCB macro, or AMP parameter) is the minimum number of index I/O buffers required when requests that require concurrent positioning are issued.

Buffers for Direct Access

Generally, you can increase performance for direct processing by increasing the number of index buffers, because direct processing always requires a top-down search through the index. A large number of data buffers does not increase performance, because only one data buffer is used for each access.

Allocating Data Buffers for Direct Access

Because VSAM does not read ahead buffers for direct processing, only the minimum number of data buffers are needed. Only one data buffer is used for each access. If you specify more data buffers than the minimum, this has little beneficial effect.

When processing a data set directly, VSAM reads only one data control interval at a time. For output processing (PUT for update), VSAM immediately writes the updated control interval, if OPTCD=NSP is not specified in the RPL macro.

Allocating Index Buffers for Direct Access

If the number of I/O buffers provided for index records is greater than the number of requests that require concurrent positioning (STRNO), one buffer is used for the highest-level index record. Any additional buffers are used, as required, for other index set index records. 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.

Unused index buffers do not degrade performance, so you should always specify an adequate number. For optimum performance, the number of index buffers should at least equal the number of high-level index set control intervals plus one per string to contain the entire high-level index set and one sequence set control interval per string in virtual storage. Note that additional index buffers will not be used for more than one sequence set buffer per string unless shared resource pools are used. For large data sets, specify the number of index buffers equal to the number of index levels.

VSAM reads index buffers one at a time, and if you are using shared resources, you can keep your entire index set in storage. 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 control interval is not in storage.

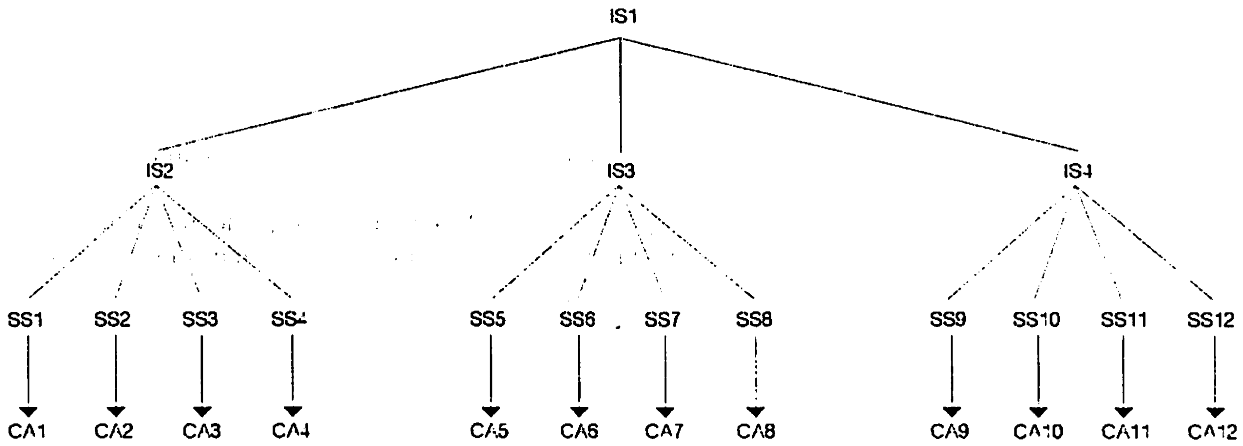
VSAM keeps as many index set records as the buffer space will allow in virtual storage. Ideally, the index would be small enough to allow the entire index set to remain in virtual storage. Because the characteristics of the data set may not allow a small index, you should be aware of how index I/O buffers are used so you can determine how many to provide.

The following example, illustrated in Figure 22, demonstrates how buffers are scheduled for direct access.

Assume the following:

- Two strings
- Three-level index structure as shown
- Three data buffers (one for each string, and one for splits)
- Four index buffers (one for highest level of index, one for second level, and one for each string)

	Index Buffers				Data Buffers	
Direct Get from CA	Pool		String 1 SS	String 2 SS	String 1	String 2
CA1	IS1	IS2	SS2		CA2-CI	
CA3				SS3		CA3-CI
CA6-1st CI		IS3	SS6		CA6-1st CI	
CA6-1st CI				SS6		CA6-1st CI



IS - Index Set
SS - Sequence Set
CA - Control Area

Figure 22. Scheduling Buffers for Direct Access

Here's what happens:

- **Request 1:** A control interval from CA2 is requested by string 1.
  - The highest level index set, IS1, is read into an index buffer. IS1 remains in this buffer for all requests.
  - IS1 points to IS2, which is read into a second index buffer.
  - IS2 points to the sequence set, SS2, which is read into an index buffer for string 1.
  - SS2 points to a control interval in CA2. This control interval is read into a data buffer for string 1.

- **Request 2:** A control interval from CA3 is requested by string 2.
  - IS1 and IS2 remain in their respective buffers.
  - SS3 is read into an index buffer for string 2.
  - SS3 points to a control interval in CA3. This control interval is read into a data buffer for string 2.
- **Request 3:** The first control interval in CA6 is requested by string 1.
  - IS1 remains in its buffer.
  - Since IS1 now points to IS3, IS3 is read into the second index buffer, replacing IS2.
  - SS6 is read into an index buffer for string 1.
  - SS6 points to the first control interval in CA6. This control interval is read into a data buffer for string 1.
- **Request 4:** The first control interval in CA6 is now requested by string 2.
  - IS1 and IS3 remain in their respective buffers.
  - SS6 is read into an index buffer for string 2.
  - SS6 points to the first control interval in CA6. This control interval is read into a data buffer for string 2.
  - If the string 1 request for this control interval was a GET for update, the control interval would be held in exclusive control, and string 2 would not be able to access it.

Suggested number of buffers for direct processing:

- Index Buffers

  Minimum = STRNO
  Maximum = Number of Index Set Records + STRNO

- Data Buffers

  STRNO + 1

Buffers for Sequential Access

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 before they are needed. Having only one index I/O buffer doesn't hinder performance, because VSAM gets to the next control interval 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.

Data Buffers for Sequential Access Using Nonshared Resources

For straight sequential processing environments, start with four data buffers per string. One buffer is used only for formatting control areas and splitting control intervals and control areas. The other three are used to support the read-ahead function, so that sequential control intervals are placed in buffers before any records from the control interval are requested. By specifying a sufficient number of data buffers, you can access the same amount of data per I/O operation with small data control intervals as with large data control intervals.

When SHAREOPTIONS 4 is specified for the data set, the read-ahead function can be ineffective because the buffers are refreshed when each control interval is read. Therefore, for SHAREOPTIONS 4, keeping data buffers at a minimum can actually improve performance.

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. This allows you to do more read-ahead processing. An excessive number of buffers, however, can cause performance problems, because of excessive paging.

For mixed processing situations (sequential and direct), start with two data buffers per string and increase BUFND to three per string, if paging is not a problem.

When processing the data set sequentially, VSAM reads ahead as buffers become available. For output processing (PUT for update), VSAM does not immediately write the updated control interval from the buffer unless a control interval split is required. The POINT macro does not cause read-ahead processing unless RPL OPTCD=SEQ is specified; its purpose is to position the data set for subsequent sequential retrieval.

Suggested number of buffers for sequential access:

- Index buffers = STRNO
- Data buffers = 3 + STRNO (minimum)

Buffer Allocation for a Path

Processing data sets using a path can increase the number of buffers that need to be allocated, since buffers are needed for the alternate index, the base cluster, and any alternate indexes in the upgrade set.

The BUFSP, BUFND, BUFNI, and STRNO parameters apply only to the path's alternate index when the base cluster is opened for processing with its 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.

Two data buffers and one index buffer are always allocated to each alternate index in the upgrade set. If the path's alternate index is a member of the upgrade set, the minimum buffer increase for each allocation is one for data buffers and one for index buffers. Buffers are allocated to the alternate index as though it were a key-sequenced data set. When a path is opened for output and the path alternate index is in the upgrade set, you can specify ACB MACRF=DSN and the path alternate index shares buffers with the upgrade alternate index.

Acquiring Buffers

Data and index buffers are acquired and allocated only when the data set is opened. VSAM dynamically allocates buffers based on parameters in effect when the program opens the data set. Parameters that influence the buffer allocation are in the program's ACB: MACRF=(IN|OUT, SEQ|SKP, DIR), STRNO=n, BUFSP=n, BUFND=n, and BUFNI=n. Other parameters that influence buffer allocation are in the DD statement's AMP specification for BUFSP, BUFND, and BUFNI, and the BUFFERSPACE value in the data set's catalog record.

If you open a data set whose ACB includes MACRF=(SEQ,DIR), buffers are allocated according to the rules for sequential processing. If the RPL is modified later in the program, the buffers allocated when the data set was opened do not change.

Data and index buffer allocation (BUFND and BUFNI) can be specified only by the user with access to modify the ACB parameters, or via the AMP parameter of the DD statement. Any program can be assigned additional buffer space by modifying the data set's BUFFERSPACE value, or by specifying a larger BUFSP value with the AMP parameter in the data set's DD statement.

When a buffer's contents are written, 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, when your program processes records in a limited key range, you might increase throughput by providing extra data buffers. Buffer space is released when the data set is closed.

Note: 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 will not help. 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 may be paged out.

Index Options

Five options influence performance through the use of the index of a key-sequenced data set. Each option improves performance, but some require that you provide additional virtual storage or auxiliary storage space. The options are:

- Specifying enough virtual storage to contain all index-set records (if you are using shared resources).
- Ensuring that the index control interval is large enough to contain the key of each control interval in the control area.
- Placing the index and the data set on separate volumes.
- Replicating index records (REPL option).

- Embedding sequence-set records adjacent to control areas (IMBED option).

Index Set Records in Virtual Storage

To retrieve a record from a key-sequenced data set or store a record using keyed access, VSAM needs to examine the index of that data set. Before your processing program begins to process the data set, it must specify the amount of virtual storage it is providing for VSAM to buffer index records. The minimum is enough space for one I/O buffer for index records, but a serious performance problem would occur if an index record were continually deleted from virtual storage to make room for another and then retrieved again later when it is required. Ample buffer space for index records can improve performance.

You ensure virtual storage for index-set records by specifying enough virtual storage for index I/O buffers when you begin to process a key-sequenced data set. VSAM keeps as many index-set records in virtual storage as possible. Whenever an index record must be retrieved to locate a data record, VSAM makes room for it by deleting the index record that VSAM judges to be least useful under the prevailing circumstances. It is generally the index record that belongs to the lowest index level or that has been used the least. VSAM does not keep more than one sequence set index record per string unless shared resource pools are used.

Avoiding Control Area Splits

The second option you might consider is to ensure that the index-set control interval is large enough to contain the key of each control interval in the control area. This reduces the number of control area splits. This option also keeps to a minimum the number of index levels required, thereby reducing search time and improving performance. However, this option may increase rotational delay and data transfer time for the index-set control intervals. It also increases virtual storage requirements for index records.

Index and Data on Separate Volumes

When a key-sequenced data set is defined, the entire index or the high-level index set alone can be placed on a volume separate from the data, either on the same or on a different type of device.

Using different volumes enables VSAM to gain access to an index and to data at the same time. Additionally, the smaller amount of space required for an index makes it economical to use a faster storage device for it.

Replicating Index Records

You can specify that each index record be replicated (written on a track of a direct access volume) as many times as it will fit. Replication reduces rotational delay in retrieving the index record. Average rotational delay is half the time it takes for the volume to complete one revolution. Replication of a record reduces this time. For example, if 10 copies of an index record fit on a track, average rotational delay is only 1/20 of the time it takes for the volume to complete one revolution.

Because there are usually few control intervals in the index set, the cost in terms of direct access storage space is small. If the entire index set is not being held in storage and there is significant random processing, then replication is a good choice. If not, replication does very little. Because replication requires little direct access storage, and it is an attribute that cannot be altered, it may be desirable to choose this option.

Sequence-Set Records Adjacent to Control Areas

When the data set is defined, you can specify that the sequence-set index record for each control area is to be embedded on the first track of the control area. This allows you to separate the sequence set from the index set and place them on separate devices. You can put the index set on a faster device to retrieve higher levels of the index. Sequence set records can be retrieved almost as fast because there is little rotational delay. (When you choose the IMBED option, sequence set records are replicated regardless of whether you also chose the REPL option.)

With the IMBED option, one track of each control area is used for sequence set records. In some situations, this may be too much space for index in relation to the data. For example, the space required for the sequence-set is 1/12 of the data space on a 3340, but only 1/15 of the data space on a 3380. IMBED must be specified explicitly to get the performance benefits of a replicated, embedded sequence-set.

Staging VSAM Data Sets on a Key or Key Range Basis for MSS

Certain Mass Storage System (MSS) applications with very large data bases require a capability for selective staging, since staging or binding of all of the data at OPEN time is not practical. Cylinder fault mode is also impractical since retrieval of data from large data sets characteristically requires a large number of queries. What you want is a mechanism to stage necessary data from multiple adjacent or nonadjacent cylinders with a single MSS cartridge load operation. This will improve performance by avoiding an inordinate amount of time spent in loading and unloading MSS cartridges for each cylinder required by the processing program.

VSAM provides a *stage-by-key range* function to support staging in this type of MSS environment. This function involves the following two interfaces:

- **Prestaging of discretely identified records** using the CNVTAD (convert address) and MNTACQ (mount acquire) macros.
- **Prestaging of a specified range of records** using the ACQRANGE (acquire range) macro.

The use of the stage-by-key range function enables you to prestage your data and thus to optimize your use of MSS in environments with large data bases. With prestaging, data extents can be acquired in advance of their use, thus reducing the number of cylinder faults incurred during processing.

Prestaging Discretely Identified Records

The CNVTAD/MNTACQ interface is typically used when your MSS application needs to refer to a small, noncontiguous subset of data contained within a large data set. The following three steps describe the processing procedure for a general application using the CNVTAD/MNTACQ interface:

STEP 1

Your application program queues a number of transactions for data stored on the MSS. On the basis of either time or the number of elements on the queue, STEP 2 is initiated.

STEP 2

The transactions for a given data set are processed to extract the keys to be used to access the data set. You supply these keys as input to the CNVTAD macro, which determines the volume and RBA on which each record resides in MSS. For a key-sequenced data set, you provide the full key of the record; for an entry-sequenced data set, you provide the RBA of the record; and for a relative record data set, you provide the relative record number of the record. The entries for a given volume may then be used by your application as input to the MNTACQ macro. MNTACQ causes the volume to be mounted (if it is not already mounted) and stages the cylinders corresponding to the RBAs provided. This results in acquiring the data from the corresponding MSS cartridges with a minimum number of cartridge loads.

STEP 3

Your application program can now process transactions for a given data set and volume, and, generally, will not encounter cylinder faults for the data acquired in STEP 2.

Prestaging of a Specified Range of Records

The ACQRANGE interface gives you the capability of prestaging a continuous subset of a data set rather than individual cylinders or the entire data set. You typically use ACQRANGE when you are processing within a range of keys and when you know the volume serial number of the virtual volume on which your range of keys resides. Input to ACQRANGE is a starting and ending pair of arguments (keys, RBAs, or relative record numbers) which delineate the continuous subset of data you want to stage. The data to be staged may cross volume boundaries, but the volumes must be mounted (via JCL or dynamic allocation) prior to execution of ACQRANGE. If not, your request is rejected. Mounting must be in parallel, which means you must have at least as many units as virtual volumes. Defining key ranges for key-sequenced data sets helps minimize the subset of volumes to be mounted.

Non-MSS Support

As an aid to MSS migration, installation, and testing, VSAM allows you to issue CNVTAD, MNTACQ, and ACQRANGE macros against non-MSS data sets. Such use against DASD that is not virtual results in register 15 being set to zero; however, the RPLERRCD field in the RPL will be nonzero for ACQRANGE and MNTACQ. For reason code information, see *VSAM Administration: Macro Instruction Reference.*

Restrictions and Limitations

The MSS function of staging VSAM data sets on a key or key range basis has the following restrictions and limitations:

- For keyed processing, key equal and key equal or greater than are supported, but the generic key facility is not supported. If you want to use generic keys, you must pad your generic key to the full key length.
- The RPL option, OPTCD=WAITX, is not supported.

- Chained RPLs are not supported.
- The data set must be opened in NCI (normal control interval access) mode.
- User buffers are not supported for LSR and GSR.
- You cannot use these macros against a data set in load mode.
- When used in path processing, alternate indexes are not supported. However, alternate indexes are supported when opened as end-use VSAM objects.
- You cannot use these macros against a key-sequenced data set with an embedded single-level index.
- Data staged by MNTACQ and ACQRANGE is not "bound." That is, it may be destaged prior to use of the data because of the requirements for MSS facilities, either by your own program or by another MSS user program.
- Prestaging data should be avoided if a data set is being shared by another task or region. Because the index is used well in advance of the actual reference to the data, concurrent insertions by other users can render the target addresses derived by ACQRANGE or CNVTAD obsolete and erroneous before or after data records have been prestaged. Prestaging, as described in this chapter, has no facility for detecting or enforcing various sharing situations. If your installation allows prestaging concurrent with data set sharing, you can expect unpredictable errors and inaccurate staging.
- MNTACQ and ACQRANGE macros obtain virtual storage dynamically via the GETMAIN macro. If the request for storage fails, a logical error code is set in the RPL.

Using Alternate Indexes with MSS Macros

Although MSS macros do not specifically support alternate indexes and paths, support for alternate indexes can be gained. The alternate index is a key-sequenced data set and can be processed as such. CNVTAD, MNTACQ, and ACQRANGE macros can be used to stage the data portion of the alternate index. After you have staged the alternate index, the RBA and key pointers can be extracted from the data component of the alternate index. These pointers can be used as input to CNVTAD, MNTACQ, and ACQRANGE for the base data.

Chapter 7. Processing Control Intervals

Product-Sensitive Programming Interface

This chapter is intended to help you process control intervals. It contains product-sensitive programming interfaces provided by MVS/XA Data Facility Product. Installation exits and other product-sensitive interfaces are provided to allow your installation to perform tasks such as product tailoring, monitoring, modification, or diagnosis. They are dependent on detailed design or implementation of the product. Such interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Control interval access gives you access to the contents of a control interval; keyed access and addressed access give you access to individual data records.

With control interval access, you have the option of letting VSAM manage I/O buffers or managing them yourself (user buffering). With keyed and addressed access, VSAM always manages I/O buffers. If you choose user buffering, you have the further option of using improved control interval access, which provides faster processing than normal control interval access. With user buffering, only control interval processing is allowed.

Control interval access allows you greater flexibility in processing entry-sequenced data sets. The definition of an entry-sequenced data set includes the unchangeability of RBAs and the permanence of records. With control interval access, you can change the RBAs of records in a control interval and delete records by modifying the RDFs and the CIDF.

When using control interval processing, you are responsible for maintaining alternate indexes. If you have specified keyed or addressed access (ACB MACRF={KEY|ADR},...) and control interval access, then those requests for keyed or addressed access (RPL OPTCD={ KEY|ADR},...) cause VSAM to upgrade the alternate indexes. Those requests specifying control interval access will not upgrade the alternate indexes. You are responsible for upgrading them.

When you process control intervals, you are responsible for how your processing affects indexes, RDFs, and CIDFs. Bypassing use of the control information in the CIDF and RDFs may make the control interval unusable for record level processing. For instance, key-sequenced data sets depend on the accuracy of their indexes and the RDFs and the CIDF in each control interval. You should not update key-sequenced data sets with control interval access. Upgrading an alternate index is described in "Alternate Index Maintenance" on page 35.

Gaining Access to a Control Interval

Control interval access is specified entirely by the ACB MACRF parameter and the RPL (or GENCB) OPTCD parameter. To prepare for opening a data set for control interval access with VSAM managing I/O buffers, specify:

```
ACB      MACRF=(CNV,...),...
```

With NUB (no user buffering) and NCI (normal control interval access), you may specify, in the MACRF parameter, that the data set is to be opened for keyed and addressed access as well as for control interval access. For example, MACRF=(CNV, KEY, SKP, DIR, SEQ, NUB, NCI, OUT) is a valid combination of subparameters.

You define a particular request for control interval access by coding:

```
RPL      OPTCD=(CNV,...),...
```

In general, control interval access with no user buffering has the same freedoms and limitations as keyed and addressed access have. Control interval access may be synchronous or asynchronous, may have the contents of a control interval moved to your work area (OPTCD=MVE) or left in VSAM's I/O buffer (OPTCD=LOC), and may be defined by a chain of request parameter lists (except with OPTCD=LOC specified).

With the exception of ERASE, all the request macros (GET, PUT, POINT, CHECK, and ENDREQ) can be used for normal control interval access. To update the contents of a control interval, you must (with no user buffering) previously have retrieved the contents for update. You cannot alter the contents of a control interval with OPTCD=LOC specified.

Both direct and sequential access may be used with control interval access, but skip sequential access may not. That is, you may specify OPTCD=(CNV,DIR) or (CNV,SEQ), but not OPTCD=(CNV,SKP).

With sequential access, VSAM takes an EODAD exit when you try to retrieve the control interval whose CIDF is filled with 0's or, if there is no such control interval, when you try to retrieve a control interval beyond the last one. A control interval with such a CIDF contains no data or unused space, and is used to represent the software end-of-file. However, VSAM control interval processing does not prevent you from using a direct GET or a POINT and a sequential GET to retrieve the software end-of-file. The search argument for a direct request with control interval access is the RBA of the control interval whose contents are desired.

The RPL (or GENCB) parameters AREA and AREALEN have the same use for control interval access in relation to OPTCD=MVE or LOC as they do for keyed and addressed access. With OPTCD=MVE, AREA gives the address of the area into which VSAM moves the contents of a control interval; with OPTCD=LOC, AREA gives the address of the area into which VSAM puts the address of the I/O buffer containing the contents of the control interval.

You may load an entry-sequenced data set with control interval access. If you open an empty entry-sequenced data set, VSAM allows you to use only sequential storage. That is, you may issue only PUTs, with OPTCD=(CNV,SEQ,NUP).

PUT with OPTCD=NUP stores information in the next available control interval (at the end of the data set).

You may not load or extend a data set with improved control interval access. VSAM also prohibits you from extending a relative record data set by way of normal control interval access.

You can update the contents of a control interval in one of two ways:

- Retrieve the contents with OPTCD=UPD and store them back. In this case, the RBA of the control interval is specified during the GET for the control interval.
- Without retrieving the contents, store new contents in the control interval with OPTCD=UPD. (This is only possible if you specify UBF for user buffering.) In this case, because no GET (or a GET with OPTCD=NUP) precedes the PUT, you have to specify the RBA of the control interval as the argument addressed by the RPL.

Structure of Control Information

With keyed access and addressed access, VSAM maintains the control information in a control interval. With control interval access, you are responsible for that information.

Note: A linear data set has no control information imbedded in the control interval. All of the bytes in the control interval are data bytes; there are no CIDFs or RDFs.

Figure 23 shows the relative positions of data, unused space, and control information in a control interval. For more information on the structure of a control interval, see "Control Intervals" on page 3.

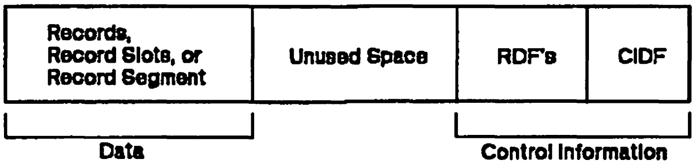


Figure 23. General Format of a Control Interval

Control information consists of a CIDF (control interval definition field) and, for a control interval containing at least one record, record slot, or record segment, one or more RDFs (record definition fields). The CIDF and RDFs are ordered from right to left.

CIDF—Control Interval Definition Field

The CIDF is a 4-byte field that contains two 2-byte binary numbers:

Offset	Length	Description
0(0)	2	The displacement from the beginning of the control interval to the beginning of the unused space, or, if there is no unused space, to the beginning of the control information. This number is equal to the length of the data (records, record slots, or record segment). In a control interval without data, the number is 0.
2(2)	2	The length of the unused space. This number is equal to the length of the control interval, minus the length of the control information, minus the 2-byte value at CIDF+0. In a control interval without data (records, record slots, or record segment), the number is the length of the control interval, minus 4 (the length of the CIDF; there are no RDFs). In a control interval without unused space, the number is 0.
2(2)	1...	Busy flag; set when the control interval is being split; reset when the split is complete.

In an entry-sequenced data set, when there are unused control intervals beyond the last one that contains data, the first of the unused control intervals contains a CIDF filled with 0's. In a key-sequenced or relative record data set or a key-range portion of a key-sequenced data set, the first control interval in the first unused control area (if any) contains a CIDF filled with 0's. This represents the software end-of-file.

RDF—Record Definition Field

The RBAs of records or relative record numbers of slots in a control interval ascend from left to right. RDFs from right to left describe these records or slots or a segment of a spanned record. RDFs describe records one way for key-sequenced and entry-sequenced data sets and another way for relative record data sets.

In a key-sequenced or entry-sequenced data set, records may vary in length and may span control intervals. This affects the contents of the RDF.

- A nonspanned record with no other records of the same length next to it is described by a single RDF that gives the length of the record.
- Two or more consecutive nonspanned records of the same length are described by a pair of RDFs. The RDF on the right gives the length of each record, and the RDF on the left gives the number of consecutive records of the same length.
- Each segment of a spanned record (one segment per control interval) is described by a pair of RDFs. The RDF on the right gives the length of the segment, and the RDF on the left gives its update number. (The update number in each segment is incremented by one each time a spanned record is updated. A difference among update numbers within a spanned record indicates a possible error in the record.)

In a relative record data set, records do not vary in length or span control intervals. Each record slot is described by a single RDF that gives its length and indicates whether it contains a record.

An RDF is a 3-byte field that contains a 1-byte control field and a 2-byte binary number:

Offset	Length and Bit Pattern	Description
0(0)	1	Control Field
	x... ..xx	Reserved
	.x..	Indicates whether there is (1) or is not (0) a paired RDF to the left of this RDF.
	..xx	Indicates whether the record spans control intervals: 00 No. 01 Yes; this is the first segment. 10 Yes; this is the last segment. 11 Yes; this is an intermediate segment.
	.... x...	Indicates what the 2-byte binary number gives: 0 The length of the record, segment, or slot described by this RDF. 1 The number of consecutive nonspanned records of the same length, or the update number of the segment of a spanned record.
	.... .x..	For a relative record data set, indicates whether the slot described by this RDF does (0) or does not (1) contain a record.
1(1)	2	Binary number: When bit 4 of byte 0 is 0, gives the length of the record, segment, or slot described by this RDF. When bit 4 of byte 0 is 1 and bits 2 and 3 of byte 0 are 0, gives the number of consecutive records of the same length. When bit 4 of byte 0 is 1 and bits 2 and 3 of byte 0 are not 0, gives the update number of the segment described by this RDF.

Control Field Values for Nonspanned Key-Sequenced and Entry-Sequenced Data Sets

In a key-sequenced or entry-sequenced data set with nonspanned records, the possible hexadecimal values in the control field of an RDF are:

Left RDF	Right RDF	Description
	X'00'	This RDF gives the length of a single nonspanned record.
X'08'	X'40'	The right RDF gives the length of each of two or more consecutive nonspanned records of the same length. The left RDF gives the number of consecutive nonspanned records of the same length.

Figure 24 shows the contents of the CIDF and RDFs of a 512-byte control interval containing nonspanned records of different lengths.

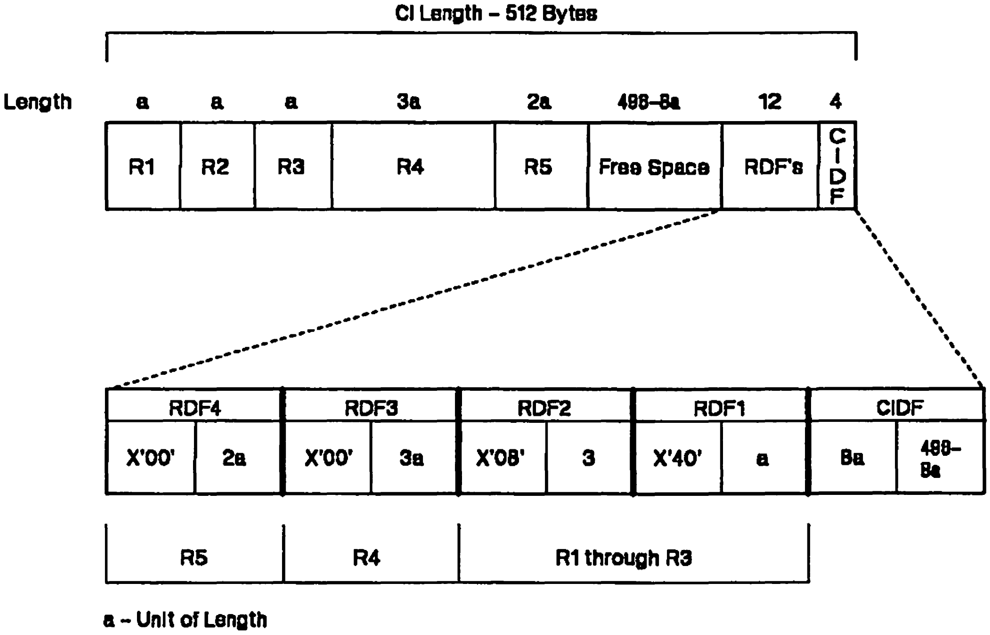


Figure 24. Format of Control Information for Nonspanned Records

The four RDFs and the CIDF comprise 16 bytes of control information as follows:

- RDF4 describes the fifth record.
- RDF3 describes the fourth record.
- RDF2 and RDF1 describe the first three records.
- The first 2-byte field in the CIDF gives the total length of the five records—8a, which is the displacement from the beginning of the control interval to the free space.
- The second 2-byte field gives the length of the free space, which is the length of the control interval minus the total length of the records and the control information—512 minus 8a minus 16, or 496 minus 8a.

Control Field Values for Spanned Key-Sequenced and Entry-Sequenced Data Sets

A control interval that contains the record segment of a spanned record contains no other data; it always has two RDFs. The possible hexadecimal values in their control fields are:

Left RDF	Right RDF	Description
X'18'	X'50'	The right RDF gives the length of the first segment of a spanned record. The left RDF gives the update number of the segment.
X'28'	X'60'	The right RDF gives the length of the last segment of a spanned record. The left RDF gives the update number of the segment.
X'38'	X'70'	The right RDF gives the length of an intermediate segment of a spanned record. The left RDF gives the update number of the segment.

Figure 25 shows contents of the CIDF and RDFs for a spanned record with a length of 1306 bytes. There are three 512-byte control intervals that contain the segments of the record. The number "n" in RDF2 is the update number. Only the control interval that contains the last segment of a spanned record can have free space. Each of the other segments uses all but the last 10 bytes of a control interval.

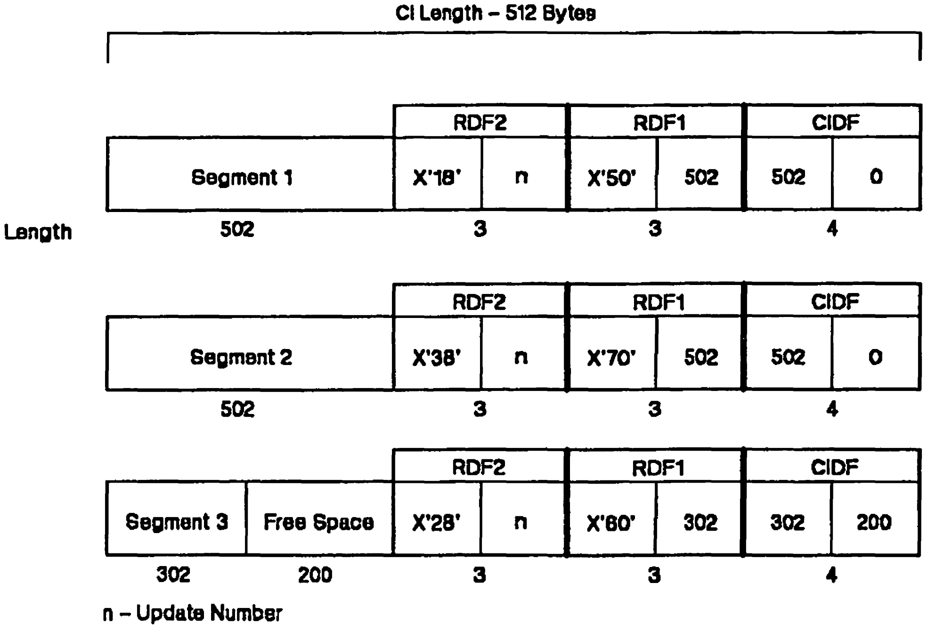


Figure 25. Format of Control Information for Spanned Records

In a key-sequenced data set, the control intervals might not be contiguous or in the same order as the segments (that is, for example, the RBA of the second segment can be lower than the RBA of the first segment).

All the segments of a spanned record must be in the same control area. When a control area does not have enough control intervals available for a spanned record, the entire record is stored in a new control area.

Control Field Values for Relative Record Data Sets

In a relative record data set, the possible hexadecimal values in the control field of an RDF are:

X'04' This RDF gives the length of an empty slot.

X'00' This RDF gives the length of a slot that contains a record.

Every control interval in a relative record data set contains the same number of slots and the same number of RDFs, one for each slot. The first slot is described by the rightmost RDF; the second slot is described by the next RDF to the left, and so on.

User Buffering

With control interval access, you have the option of *user buffering*. This means that you provide buffers in your own area of storage for use by VSAM.

User buffering is required for improved control interval access (ICI) and for PUT with OPTCD=NUP.

With ACB MACRF=(CNV,UBF) specified (control interval access with user buffering), the work area specified by the RPL (or GENCB) AREA parameter is. in effect, the I/O buffer. VSAM transmits the contents of a control interval directly between the work area and direct access storage.

If you specify user buffering, you cannot specify KEY or ADR in the MACRF parameter; you can specify only CNV. That is, you cannot intermix keyed and addressed requests with requests for control interval access.

OPTCD=LOC is inconsistent with user buffering and is not allowed.

Improved Control Interval Access

With user buffering you have the option of specifying improved control interval access:

```
ACB      MACRF=(CNV,UBF,ICI,...),...
```

Improved control interval access is faster than normal control interval access because the path length is shorter. However, with user buffering there is no read-ahead buffering, so you can only have one control interval scheduled at a time. This means that improved control interval access performs well for direct processing.

You cannot load or extend a data set using improved control interval processing.

A processing program can achieve the best performance with improved control interval access by combining it with SRB dispatching. SRB mode with fixed control blocks provides the fastest path. (SRB dispatching is described in *System Macros and Facilities*.)

Opening an Object for Improved Control Interval Access

Improved control interval processing is faster because functions have been removed from the path. However, this causes several restrictions:

- The object must not be empty.
- The object must be one of the following:
  - An entry-sequenced or relative record cluster
  - The data component of an entry-sequenced, key-sequenced, linear, or relative record cluster
  - The index component of a key-sequenced cluster (index records must not be replicated)
- Control intervals must be the same size as physical records. When you use the access method services DEFINE command to define the object, you can specify control interval size equal to a physical record size used for the device on which the object is stored. VSAM uses physical record sizes of (n x 512) and (n x 2048), where n is a positive integer from 1 to 16.

  The following table identifies the direct access devices for which the physical record size equal to the control interval size is selected for a data component. The physical record size is always equal to the control interval size for an index component.

	Control Interval Size (Bytes)			
Device	**512**	**1024**	**2048**	**4096**
2305-2	X	X	X	X
3330	X	X	X	X
3330-1	X	X	X	X
3340	X	X		X
3344	X	X		X
3350	X	X	X	X
3375	X	X	X	X
3380[1]	X	X	X	X

Processing a Data Set with Improved Control Interval Access

To process a data set with improved control interval access, a request must be:

- Defined by a single RPL (VSAM ignores the NXTRPL parameter).
- A direct GET, GET for update, or PUT for update (no POINT, no processing empty data sets). A relative record data set with slots formatted is considered not to be empty, even if no slot contains a record.
- Synchronous (no CHECK, no ENDREQ).

[1] 3380, all models

In order to release exclusive control after a GET for update, you must issue a PUT for update, a GET without update, or a GET for update for a different control interval.

With improved control interval access, VSAM assumes (without checking) that an RPL whose ACB has MACRF=ICI has OPTCD=(CNV, DIR, SYN); that a PUT is for update (RPL OPTCD=UPD); and that your buffer length (specified in RPL AREALEN=number) is correct. Because VSAM does not check these parameters, you should debug your program with ACB MACRF=NCI, then change to ICI.

With improved control interval access, VSAM does not take JRNAD exits and does not keep statistics (which are normally available by way of SHOWCB).

Fixing Control Blocks and Buffers in Real Storage

With improved control interval access, you can specify that control blocks are to be fixed in real storage (ACB MACRF=(CFX,...)). If you so specify, your I/O buffers must also be fixed in real storage. Having your control blocks fixed in real storage, but not your I/O buffers, may cause physical errors or unpredictable results. If you specify MACRF=CFX without ICI, VSAM ignores CFX. NFX is the default; it indicates that buffers are not fixed in real storage, except for an I/O operation. A program must be authorized to fix pages in real storage, either in supervisor state with protection key 0 to 7, or link-edited with authorization. (The authorized program facility is described in *System Macros and Facilities*.) An unauthorized request is ignored.

Control Blocks in Common (CBIC) Option

When you are using improved control interval processing, the CBIC option allows you to have multiple address spaces that address the same data and use the same control block structure. The VSAM control blocks associated with a VSAM data set are placed into the common service area (CSA). The control block structure and VSAM I/O operations are essentially the same whether or not the CBIC option is invoked, except for the location of the control block structure. The user-related control blocks are generated in the protect key (0 through 7); the system-related control blocks are generated in protect key 0. The VSAM control block structure generated when the CBIC option is invoked retains normal interfaces to the region that opened the VSAM data set (for example, the DEB is chained to the region's TCB).

The CBIC option is invoked when a VSAM data set is opened. To invoke the CBIC option, you set the CBIC flag (located at offset X'33' (ACBINFL2) in the ACB, bit 2 (ACBCBIC)) to one. When your program opens the ACB with the CBIC option set, your program must be in supervisor state with a protect key from 0 to 7; otherwise, VSAM will not open the data set.

The following restrictions apply when using the CBIC option:

- The CBIC option must be used only when the ICI option is also specified.
- You cannot also specify LSR or GSR.
- You cannot use the following types of data sets with the CBIC option: catalogs, catalog recovery areas, swap data sets, or system data sets.

- If an address space has opened a VSAM data set with the CBIC option, your program cannot take a checkpoint for that address space.

If another region accesses the data set's control block structure in the CSA via VSAM record management, the following conditions should be observed:

- An OPEN macro should not be issued against the data set.
- The ACB of the user who opened the data set with the CBIC option must be used.
- CLOSE and temporary CLOSE cannot be issued for the data set (only the user who opened the data set with the CBIC option can close the data set).
- The region accessing the data set control block structure must have the same storage protect key as the user who opened the data set with the CBIC option.
- User exit routines should be accessible from all regions accessing the data set with the CBIC option.

End of Product-Sensitive Programming Interface

Chapter 8. Data Security and Integrity

The protection of data includes:

- Data security, or the safety of data from theft or intentional destruction
- Data integrity, or the safety of data from accidental loss or destruction

The following sections describe the available data protection:

- Resource Access Control Facility (RACF)
- Authorized program facility (APF)
- Access method services password protection
- User-security-verification routine (USVR)
- Access method services cryptographic option

Resource Access Control Facility (RACF)

Resource Access Control Facility (RACF) provides an optional software access control measure you may use in addition to, or instead of, passwords. Password protection and RACF protection can coexist for the same data set. When RACF protection is applied to a data set that is already password-protected and the catalog containing it is protected by RACF, password protection is bypassed and access is controlled solely through the RACF authorization mechanism. If a user-security-verification routine (USVR) exists, it is not invoked for RACF-defined data sets.

To apply RACF protection to a data set in a VSAM catalog. the catalog containing it must be RACF-protected. An integrated catalog facility catalog, however, does not have to be RACF-protected in order for its data sets to be RACF-protected.

To have password protection take effect for a data set, the catalog containing it must be either RACF-protected or password-protected and the data set itself must not be defined to RACF. Although passwords are ignored for a RACF-protected data set, they can still provide protection if the data set is moved to a system that does not have RACF protection.

Using a Generic Profile

RACF Release 5 and later releases provide a *generic profile checking facility*. With the always call capability of integrated catalog facility catalogs, you can consolidate the access authorization requirements of several similarly named and similarly used data sets under a single generic profile definition. A generic profile is used to protect one cluster or a group of clusters that require similar access authority. For example, you could build a generic profile with a high level qualifier of *userid*.*. In a TSO environment, this profile would protect all your data sets cataloged in integrated catalog facility catalogs.

VSAM data sets that are generically protected are not RACF-indicated in the catalog. Therefore, whether or not a data set is RACF-indicated or password-protected, RACF is always called for access to data sets cataloged in integrated catalog facility catalogs. If the data set is not protected by either a discrete profile or a generic profile, password protection is in effect.

For clusters cataloged in an integrated catalog facility catalog, a generic profile is used to verify access to the entire cluster, or any of its components. Discrete profiles for the individual components may exist, but **only** the cluster's profile (generic or discrete) is used to protect the components in the cluster.

Note: Profiles defined by ADSP processing during a data set define operation will be cluster profiles only.

Data sets protected with discrete profiles are flagged as RACF-indicated. If a data set protected by a discrete profile is moved to a system where RACF is not installed, no user is given authority to access the data set. However, if the data set is protected with a generic profile, it is not flagged as RACF-indicated; therefore, access authority is determined by normal VSAM password protection.

Checking Authorization

RACF authorization checking is generally compatible with the password authorization checking scheme. The compatibility includes the time the authorization check is made and the sources of authorization. The RACF authorization levels of alter, control, update, and read correspond to the VSAM password levels of master, control, update, and read.

Deleting any type of RACF-protected entry from a RACF-protected catalog requires alter-level authorization to the catalog or the entry being deleted.

Altering the passwords in a RACF-protected catalog entry requires RACF alter authority to the entry being altered, or the operations attribute. Alter authority to the catalog itself is not sufficient for this operation.

Erasing Residual Data

You can use RACF to control the erasure of residual data for integrated catalog facility cataloged VSAM clusters or alternate indexes by specifying an erase indicator in generic or discrete profiles. To control the erasure of sensitive data with RACF:

- The erase feature must be activated by the RACF SETROPTS command,
- The cluster must be RACF-protected, with the RACF ERASE option specified in the generic or discrete profile, and
- Specifying NOERASE using the access methods services commands will not override erasure if the RACF profile has the ERASE option set.

For more information about RACF security features see *RACF General Information Manual* and associated RACF publications.

Authorized Program Facility (APF)

The authorized program facility (APF) limits the use of sensitive system services and resources to authorized system and user programs. For information about program authorization, see "Using the Authorized Program Facility (APF)" in *System Macros and Facilities.*

All access method services load modules are contained in SYS1.LINKLIB, and the root segment load module (IDCAMS) is link-edited with the SETCODE AC(1) attribute. These two characteristics ensure that access method services executes with APF authorization.

APF authorization is established at the job step level. If, during the execution of an APF-authorized job step, a load request is satisfied from an unauthorized library, the task is abnormally terminated. It is the installation's responsibility to ensure that a load request cannot be satisfied from an unauthorized library during access method services processing.

The following situations could cause the invalidation of APF authorization for access method services:

- An access method services module is loaded from an unauthorized library.
- A user-security-verification routine (USVR) is loaded from an unauthorized library during access method services processing.
- An exception exit routine is loaded from an unauthorized library during access method services processing.
- A user-supplied special graphics table is loaded from an unauthorized library during access method services processing.

Because APF authorization is established at the job step task level, access method services is not authorized if invoked by an unauthorized problem program or an unauthorized terminal monitor program (TMP).

You must enter the names of those access method services commands requiring APF authorization to execute under TSO in the authorized command list.

The restricted functions performed by access method services that cannot be requested in an unauthorized state are:

CNVTCAT	when converting to an integrated catalog facility catalog
DEFINE	when the RECATALOG parameter is specified
DELETE	when the RECOVERY parameter is specified
EXPORT	when the object to be exported is an integrated catalog facility catalog
IMPORT	when the object to be imported is an integrated catalog facility catalog
PRINT	when the object to be printed is a catalog
REPRO	when copying an integrated catalog facility catalog, or the integrated catalog facility catalog unload/reload is to be used
VERIFY	when a catalog is to be verified

If the above functions are required and access method services is invoked from a problem program or a TSO terminal monitor program, the invoking program must be authorized.

Access Method Services Password Protection

Access method services provides options to protect data sets against unauthorized use and loss of data. To effectively use the protection features, you must understand the difference between operations on a catalog and operations on data sets represented by a catalog entry:

- Referring to a catalog entry when new entries are defined (DEFINE), or existing entries are altered (ALTER), deleted (DELETE), or listed (LISTCAT).
- Using the data set represented by a catalog entry when it is connected to a user's program (OPEN), or disconnected (CLOSE).

Different passwords may be needed for each type of operation.

Operations on a catalog may be authorized by the catalog's password or, in some cases, by the password of the data set defined in the catalog. The *Access Method Services Reference* describes which level of password is required for each operation.

The following are examples of passwords required for defining, listing, and deleting catalog entries.

- Defining a data set in a password-protected catalog requires the catalog's update (or higher) password.
- Listing, altering, or deleting a data set's catalog entry requires the appropriate password of either the catalog or the data set. However, if the catalog, but not the data set, is protected, no password is needed to list, alter, or delete the data set's catalog entry.

OPEN and CLOSE operations on a data set may be authorized by the password pointed to by the PASSWD parameter of the ACB macro. The "ACB Macro" section in *VSAM Administration: Macro Instruction Reference* describes which level of password is required for each type of operation.

Passwords to Authorize Access

You may, optionally, define passwords for access to clusters, cluster components (data and index), page spaces, alternate indexes, alternate index components (data and index), paths, master and user catalogs. Different passwords have various degrees of security, with higher levels providing greater protection than lower levels. The levels are:

- Full access. This is the master password, which allows you to perform all operations (retrieving, updating, inserting, and deleting) on an entire VSAM data set and any index and catalog record associated with it. The master password allows all operations and bypasses any additional verification checking by the user-security-verification routine.
- Control access. This password authorizes you to use control interval access. For further information, see Chapter 7, "Processing Control Intervals" on page 87.
- Update access. This password authorizes you to retrieve, update, insert, or delete records in a data set. The update password does not allow you to alter passwords or other security information.

- Read access. The read-only password allows you to examine data records and catalog records, but not to add, alter, or delete them, nor to see password information in a catalog record.

Each higher-level password allows all operations permitted by lower levels. Any level may be null (not specified), but if a low-level password is specified, the DEFINE and ALTER commands give the higher passwords the value of the *highest* password specified. For example, if only a read-level password is specified, the read-level becomes the update-, control-, and master-level password as well. If you specify a read password and a control password, the control password value becomes the master-level password as well. However, in this case, *the update-level password is null* because the value of the read-level password is not given to higher passwords.

Catalogs are themselves VSAM data sets, and may have passwords. For some operations (for example, listing all the catalog's entries with their passwords or deleting catalog entries), the catalog's passwords may be used instead of the entry's passwords. If the master catalog is protected, the update- or higher-level password is required when defining a user catalog, because all user catalogs have an entry in the master catalog. When deleting a protected user catalog, the user catalog's master password must be specified.

Some access method services operations may involve more than one password authorization. For example, importing a data set involves defining the data set and loading records into it. If the catalog into which the data set is being imported is password protected, its update-level (or higher-level) password is required for the definition; if the data set is password protected, its update-level (or higher-level) password is required for the load. The IMPORT command allows you to specify the password of the catalog; the password, if any, of the data set being imported is obtained by the commands from the exported data.

Every VSAM data set is represented in a catalog by two or more components: a cluster component and a data component, or, if the data set is a key-sequenced data set, a cluster component, a data component, and an index component. Of the two or three components, the cluster component is the controlling component. Each of the two or three components can have its own set of four passwords; the passwords you assign have no relationship to each other. For example, password protecting a cluster but not the cluster's data component, allows someone to issue LISTCAT to determine the name of your cluster's data component, open the data component, and access records in it, even though the cluster itself is password protected.

One reason for password protecting the components of a cluster is to prevent access to the index of a key-sequenced data set. (One way to gain access to an index is to open it independently of the cluster.) See Chapter 7, "Processing Control Intervals" on page 87 for a description of access to an index.

Password Protection Considerations and Precautions

For a Catalog

Observe the following precautions when using protection commands for the catalog:

- To create a catalog entry (with the DEFINE command), the update- or higher-level password of the catalog is required.
- To modify a catalog entry (with the ALTER command), the master password of the entry or the master password of the catalog which contains the entry is required. However, if the entry to be modified is a non-VSAM or generation data group entry, the update-level password of the catalog is sufficient.
- To gain access to passwords in a catalog (for example, to list or change passwords), specify the master-level password of either the entry or the catalog. A master-level password must be specified with the DEFINE command to model an entry's passwords.
- To delete a protected data set entry from a catalog requires the master-level password of the entry or the master-level password of the catalog containing the entry. However, if the entry in a VSAM catalog describes a VSAM data space, the update-level password of the catalog is sufficient.
- To delete a non-VSAM, generation data group, or alias entry, the update level password of the catalog is sufficient.
- To list catalog entries with the read-level passwords, specify the read password of the entry or the catalog's read level password. However, entries without passwords may be listed without specifying the catalog's read-level password. To list the passwords associated with a catalog entry, specify the master password of the entry or the catalog's master password.

  To avoid unnecessary prompts, specify the catalog's password, which allows access to all entries that the operation affects. A catalog's master-level password allows you to refer to all catalog entries. However, a protected cluster cannot be processed with the catalog's master password.
- Specification of a password where none is required is always ignored.

For a Data Set

Observe the following precautions when using protection commands for data sets:

- To access a VSAM data set using its cluster name, instead of data or index names, you must specify the proper level password for the cluster even if the data or index passwords are null.
- To access a VSAM data set using its data or index name, instead of its cluster name, you must specify the proper data or index password. However, if cluster passwords are defined, the master password of the cluster may be specified instead of the data or index password.
- If a cluster has only null (not specified) passwords, you may access the data set using the cluster name without specifying passwords. This is true even if the data and index entries of the cluster have defined passwords. This allows unrestricted access to the VSAM data set as a whole but protects against unauthorized modification of the data or index as separate components.

Relation of Data Set and Catalog Protection

If you define passwords for any data sets in a catalog, you must also protect the catalog by defining passwords for the catalog or by defining the catalog to RACF. If you do not protect the catalog, no password checking takes place during operations on the data set's catalog entries or during open processing of data sets cataloged in that catalog.

Password Prompting

Computer operators and TSO terminal users may supply a correct password if a processing program does not give the correct one when it tries to open a password-protected data set. When the data set is defined, you may use the CODE parameter to specify a code instead of the data set name to prompt the operator or terminal user for a password. The prompting code keeps your data secure by not allowing the operator or terminal user to know both the name of the data set and its password.

A data set's code is used for prompting for any operation against a password-protected data set. The catalog code is used for prompting when the catalog is opened as a data set, when an attempt is made to locate catalog entries that describe the catalog, and when an entry is to be defined in the catalog.

If you do not specify a prompting code, VSAM identifies the job for which a password is needed with the JOBNAME and DSNAME for background jobs or with the DSNAME alone for foreground (TSO) jobs.

When you define a data set, you may use the ATTEMPTS parameter to specify the number of times the computer operator or terminal user is allowed to give the password when a processing program is trying to open a data set.

If the ATTEMPTS parameter is coded with 0, no password prompting is done. If the allowed number of attempts is exceeded and you are using System Management Facilities, a record is written to the SMF data set to indicate a security violation.

Note: When logged onto TSO, VSAM tries the logon password before prompting at the user terminal. Using the TSO logon password counts as one attempt.

Passwords for Non-VSAM Data Sets

When you define a non-VSAM data set in an integrated catalog facility catalog, the data set is not protected with passwords in its catalog entry. To password protect a non-VSAM data set when it is created, specify LABEL=(PASSWORD|NOPWREAD) in the DD statement that describes the data set (for more details, see *JCL*). Use the PROTECT macro instruction to assign a password to the non-VSAM data set (for more details, see *System—Data Administration*).

If the catalog is update-protected, you must supply the catalog's update- or higher-level password to define, delete, or alter a non-VSAM data set. The password can be supplied as a subparameter of the command's CATALOG parameter, or as a response to the password-prompting message.

User-Security-Verification Routine (USVR)

In addition to password protection, VSAM allows you to protect data by specifying the program that verifies a user's authorization. Specific requirements of the user-security-verification routine are described in *Data Facility Product: Customization.* To use this routine, specify the name of the authorization routine you have written in the AUTHORIZATION parameter of the DEFINE or ALTER command.

If a password exists for the type of operation being performed, the password must be given, either in the command or in response to prompting. The user-security-verification routine is called only after the password specified is verified; it is bypassed whenever a correct master password is specified, whether or not the master password is required for the requested operation.

Access Method Services Cryptographic Option

Although you can provide security for online data by using such facilities as VSAM password protection and the IBM Resource Access Control Facility (RACF) program product, these facilities do not protect data when it is stored offline. Sensitive data stored offline is susceptible to misuse.

It is generally recognized that data cryptography is an effective means of protecting offline data, if the enciphering techniques are adequate. The enciphering function is available by using the ENCIPHER option of the access method services REPRO command. The REPRO command uses the services of the IBM Programmed Cryptographic Facility (5740-XY5) or the Cryptographic Unit Support (5740-XY6). The Programmed Cryptographic Facility conforms to the Data Encryption Standard (DES) of the United States National Bureau of Standards for enciphering data. The data remains protected until the REPRO DECIPHER option is used to decipher it with the correct key.

Note: The format and examples of the REPRO command are in *Access Method Services Reference.*

There are three types of offline environments in which the enciphering of sensitive data adds to its security:

- Data sets that are transported to another installation, where data security is required during transportation and while the data is stored at the other location
- Data sets that will be stored for long periods of time at a permanent storage location
- Data sets that are stored offline at the site at which they are normally used

You can use REPRO to copy a plaintext (not enciphered) data set to another data set in enciphered form. Enciphering converts data to an unintelligible form called a ciphertext. The enciphered data set can then be stored offline or sent to a remote location. When desired, the enciphered data set can be brought back online and you can use REPRO to recover the plaintext from the ciphertext by copying the enciphered data set to another data set in plaintext (deciphered) form.

Enciphering and deciphering are based on an 8-byte binary value called the key. Using the REPRO DECIPHER option, the data can be either deciphered on the system it was enciphered on, or deciphered on another system that has this functional capability and the required key to decipher the data. Given the same key, encipher and decipher are inverse operations. This option uses the services of the Programmed Cryptographic Facility or the Cryptographic Unit Support to encipher/decipher the data, and uses block chaining with ciphertext feedback during the encipher/decipher operation.

With the exception of catalogs, all data sets supported for copying by REPRO are supported as input (SAM, ISAM, VSAM) for enciphering and as output (SAM, VSAM) for deciphering. The input data set for the decipher operation must be an enciphered copy of a data set produced by REPRO. The output data set for the encipher operation can only be a VSAM entry-sequenced or sequential (SAM) data set. The target (output) data set of both an encipher and a decipher operation must be empty. If the target data set is a VSAM data set that has been defined with the reusable attribute, you can use the REUSE parameter of REPRO to reset it to an empty status.

The REPRO ENCIPHER parameter indicates that REPRO is to produce an enciphered copy of the data set, and to supply information needed for the encipherment. The INFILE or INDATASET parameter identifies and allocates the plaintext (not enciphered) source data set. The OUTFILE or OUTDATASET parameter identifies and allocates a target data set to contain the enciphered data.

The REPRO DECIPHER parameter indicates that REPRO is to produce a deciphered copy of the data set, and to supply information needed for deciphering. The INFILE or INDATASET parameter identifies and allocates the enciphered source data set. The OUTFILE or OUTDATASET parameter identifies and allocates a target data set to contain the plaintext data.

Figure 26 on page 108 is a graphic representation of the input and output data sets involved in REPRO ENCIPHER/DECIPHER operations.

You should not build an alternate index over a VSAM entry-sequenced data set that is the output of a REPRO ENCIPHER operation.

When a VSAM relative record data set is enciphered, the record size of the output data set must be at least 4 bytes greater than the record size of the relative record data set. (The extra 4 bytes are needed to prefix a relative record number to the output record.) You can specify the record size of an output VSAM entry-sequenced data set through the RECORDSIZE parameter of the DEFINE CLUSTER command. You can specify the record size of an output sequential (SAM) data set through the DCB LRECL parameter in the output data set's DD statement. When an enciphered VSAM relative record data set is subsequently deciphered with a relative record data set as the target, any empty slots in the original data set are reestablished.

If you specify the REPLACE parameter of the REPRO command together with either the ENCIPHER or the DECIPHER parameter, access method services ignore REPLACE, because the target VSAM data set must be empty and has no records to replace.

When you encipher a data set, you can specify any of the delimiter parameters available with the REPRO command (SKIP, COUNT, FROMADDRESS, FROMKEY, FROMNUMBER, TOADDRESS, TOKEY, TONUMBER) that are appropriate to the data set being enciphered. However, no delimiter parameter can be specified when a data set is deciphered. If DECIPHER is specified together with any REPRO delimiter parameter, your REPRO command terminates with a message.

When the REPRO command copies and enciphers a data set, it places one or more records of clear header data preceding the enciphered data records. This header data consists of information necessary for the deciphering of the enciphered data. Information a header may contain consists of:

- Number of header records
- Number of records to be ciphered as a unit
- Key verification data
- Enciphered data encrypting keys

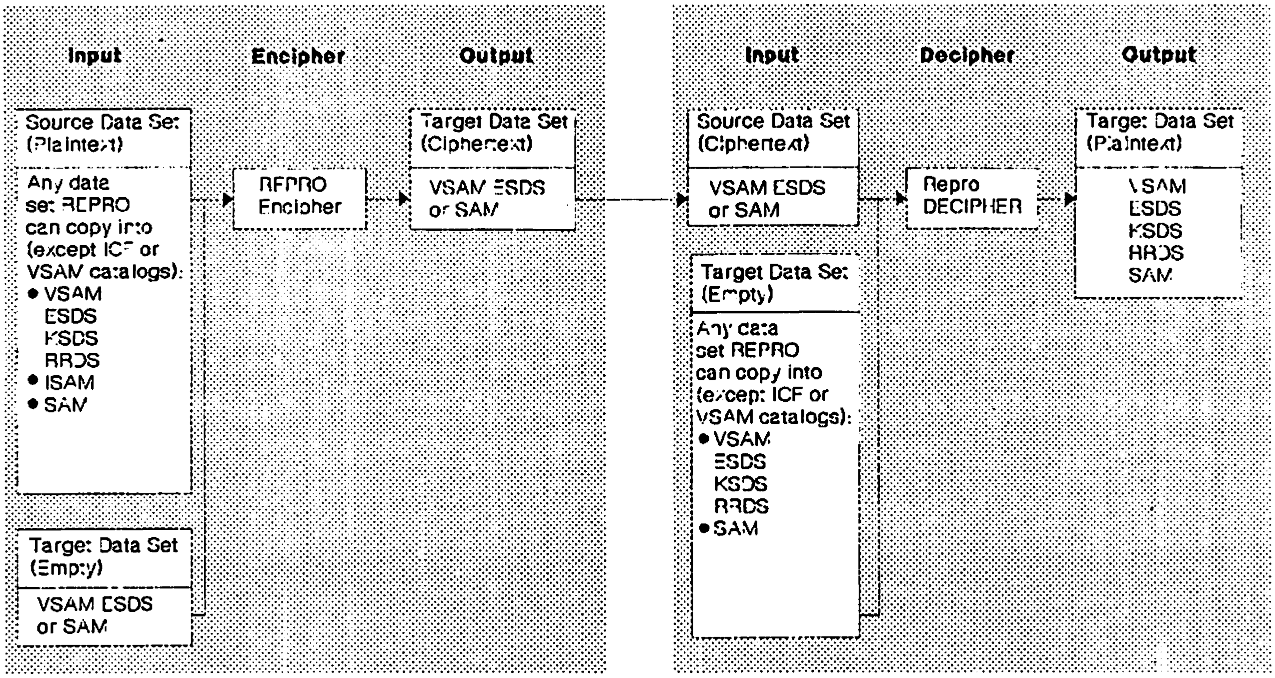

Input
Encipher
Output
Source Data Set (Plaintext)
Any data set REPRO can copy into (except ICF or VSAM catalogs):
• VSAM
ESDS
KSDS
RRDS
• ISAM
• SAM
Target Data Set (Empty)
VSAM ESDS or SAM
REPRO Encipher
Target Data Set (Ciphertext)
VSAM ESDS or SAM
Input
Decipher
Output
Source Data Set (Ciphertext)
VSAM ESDS or SAM
Target Data Set (Empty)
Any data set REPRO can copy into (except ICF or VSAM catalogs):
• VSAM
ESDS
KSDS
RRDS
• SAM
Repro DECIPHER
Target Data Set (Plaintext)
VSAM
ESDS
KSDS
RRDS
SAM

Figure 26. REPRO Encipher/Decipher Operations

Key Management

This section is intended to give you an overview of key management used when enciphering or deciphering data via the REPRO command.

Data Encryption Keys

A "key" is defined as an 8-byte value. When you use the encipher/decipher function of the REPRO command, you may specify keys that the Programmed Cryptographic Facility or the Cryptographic Unit Support generates and manages for you (system-key management), or keys that you manually generate and privately manage (private-key management). In either case, REPRO invokes the appropriate Programmed Cryptographic Facility program product service.

For both private- and system-key management, REPRO allows you to supply an 8-byte value to be used as the plaintext data encrypting key. If you do not supply the data encrypting key, REPRO provides an 8-byte value to be used as the plaintext data encrypting key. The plaintext data encrypting key is used to encipher/decipher the data using the Data Encryption Standard.

If you supply your own plaintext data encrypting key on ENCIPHER or DECIPHER through the REPRO command, you risk exposing that key when the command is listed on SYSPRINT. To avoid this exposure, you may direct REPRO to a data encrypting key data set to obtain the plaintext data encrypting key. You identify the DD statement for this data set through the DATAKEYFILE parameter of the REPRO command. REPRO obtains the data encrypting key from this data set by locating the first nonblank column in the first record of the data set and processing 16 consecutive columns of data. These 16 columns contain the hexadecimal character representation for the 8-byte key. Each column of data must contain the EBCDIC character representation of a hexadecimal digit (that is, 0 through F). If a blank column is found before 16 nonblank columns have been processed, the data key is padded to the right with EBCDIC blanks. If the end of the first record is encountered before 16 columns have been processed, the data key is considered invalid. If a valid data encrypting key cannot be obtained from the first record of the data encrypting key data set, then REPRO either generates the data encrypting key (ENCIPHER) or terminates with a message (DECIPHER).

If you allow (or force, by an invalid data encrypting key data set record) REPRO to provide the data encrypting key on ENCIPHER, you risk exposing the data encrypting key only if you manage your keys privately; REPRO lists the generated data encrypting key only for privately managed keys.

Secondary File Keys

When you want to decipher the data, you must supply the data encrypting key that was used to encipher the data. However, as a security precaution, you may want to supply the data encrypting key in a disguised form. When enciphering the data set, you may supply the name of a system-managed key, called a secondary file key. REPRO uses the secondary file key indicated by the supplied name to disguise (encipher) the data encrypting key. When deciphering the data set, the name of the file key and the disguised data encrypting key may be supplied rather than the plaintext data encrypting key. In this way, the actual plaintext data encrypting key is not revealed. (Note that the secondary file key is used only in key communication, not when enciphering data.)

The Programmed Cryptographic Facility or the Cryptographic Unit Support offers the installation this secondary file key-management facility. To use this facility, your installation must first execute the Programmed Cryptographic Facility key generator utility. This utility is used to generate the secondary-file keys to be used by the installation.

The Programmed Cryptographic Facility key generator utility generates the secondary-file keys you request and stores the keys, in enciphered form, in the cryptographic key data set (CKDS). It lists the external name (key name) of each secondary key and the plaintext form of the secondary key. To access a particular secondary file key, you supply the keyname of the secondary file key.

If the secondary key is to be used on a system other than the system on which the keys were generated, the utility must also be executed at the other system to define the same plaintext secondary file keys. The plaintext secondary file keys may be defined in the CKDS of the other system with different keynames.

If you choose to manage your own private keys, no secondary file keys are used to encipher the data encrypting key; it is your responsibility to ensure the secure nature of your private data encrypting key.

If you choose to have the Programmed Cryptographic Facility or the Cryptographic Unit Support manage your keys, REPRO uses secondary file keys to encipher the data encrypting key. You specify a secondary file key to REPRO through the key name by which it is known in the CKDS. Two types of secondary file keys are recognized by REPRO:

- An internal file key, defined in the CKDS by using a remote or cross key 2 statement as input to the Programmed Cryptographic Facility key generator utility. Use of a cross key 2 statement for DECIPHER requires APF (Authorized Program Facility) authorization and can provide an additional level of security.
- For ENCIPHER, an external file key, defined in the CKDS by using a local or cross key 1 statement as input to the Programmed Cryptographic Facility key generator utility. An internal file key may be used for ENCIPHER but not for DECIPHER operations.

Although both internal and external secondary file keys may be used when doing a REPRO ENCIPHER, only internal secondary file keys may be used when doing a REPRO DECIPHER. Thus, external secondary file keys may not be used to do a REPRO ENCIPHER and REPRO DECIPHER on the same system. External secondary file keys are used when the enciphered data is to be deciphered on another system. The REPRO DECIPHER is done on the other system using the same secondary file key; however, in this other system's CKDS, the secondary file key must be defined as an internal secondary file key.

An internal secondary file key may be used to do a REPRO ENCIPHER and REPRO DECIPHER on the same system, and, if the same secondary file key is also defined as an internal secondary file key in another system's CKDS, then a REPRO ENCIPHER and REPRO DECIPHER may also be done on that system.

Requirements

In planning to use the ENCIPHER or DECIPHER functions of the REPRO command, you should be aware of the following requirements:

- Either the Programmed Cryptographic Facility Program Product, Program Number 5740-XY5, or the Cryptographic Unit Support Facility Program Product, Program Number 5740-XY6, and its prerequisites must be installed on your system.
- Prior to issuing the REPRO command (via a batch job or from a TSO terminal), the Programmed Cryptographic Facility program product must have already been started through a START command at the operator's console.

In planning to use REPRO DECIPHER with the SYSTEMKEYS parameter specifying a cross key 2 secondary file key, you should be aware of the following requirement:

- Access method services must be authorized. For information about program authorization, see "Using the Authorized Program Facility (APF)" in *System Macros and Facilities.*

Chapter 9. Sharing a VSAM Data Set

General-Use Programming Interface

This chapter is intended to help you share data within a single system and among multiple systems. It contains general-use programming interfaces, which allow you to write programs that use the services of MVS/XA Data Facility Product.

When you define VSAM data sets, you may specify how the data is to be shared within a single system or among multiple systems that may have access to your data and share the same direct access devices. This chapter provides guidelines for accessing shared data sets and for controlling data set sharing to prevent the loss of 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 may result when one or more of the data set's users do not adhere to guidelines recommended for accessing shared data sets.

The extent to which you want your data sets to be shared depends on the application. If your requirements are similar to those of an integrated catalog facility catalog, where there may be many users on more than one system, more than one user should be allowed to read and update the data set simultaneously. At the other end of the spectrum is an application where high security and data integrity require that only one user at a time have access to the data.

When your program issues a GET request, VSAM reads an entire control interval into virtual storage (or obtains a copy of the data from a control interval already in virtual storage). If your program modifies the control interval's data, VSAM ensures within a single control block structure that you have exclusive use of the information in the control interval until it is written back to the data set. If the data set is accessed by more than one program at a time, and more than one control block structure contains buffers for the data set's control intervals, VSAM cannot ensure that your program has exclusive use of the data. You must obtain exclusive control yourself, using facilities such as ENQ/RESERVE and DEQ.

Two ways to establish the extent of data set sharing are the data set disposition specified in the JCL and the share options specified in the access method services DEFINE or ALTER command. If the VSAM data set cannot be shared because of the disposition specified in the JCL, a scheduler allocation failure occurs. If your program attempts to open a data set that is in use and the share options specified do not allow concurrent use of the data, the open fails, and a return code is set in the ACB error field.

During load mode processing, you may not share data sets. Share options are overridden during load mode processing. 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.

Subtask Sharing

Subtask sharing is the ability to perform multiple OPENs to the same data set from different subtasks in a single region and still share a single control block structure. This allows many logical views of the data set while maintaining a single structure. With a single structure, you can ensure that you have exclusive control of the buffer when updating a data set.

If you share multiple control block structures within a task or region, VSAM treats this like cross-region sharing. You must adhere to the guidelines and restrictions specified in "Cross-Region Sharing" on page 118.

To share successfully within a task or between subtasks, you should assure that VSAM builds a single control block structure for the data set. This includes blocks for control information in addition to input/output buffers. All subtasks access the data set through this single control block structure, independent of the SHAREOPTION or DISP specifications. The three methods of achieving a single 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
- ddname sharing, with multiple ACBs (all from the same data set) pointing to a single DD statement. This is the default.

  For 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. The data set names are related with an ACB open specification (MACRF = DSN).

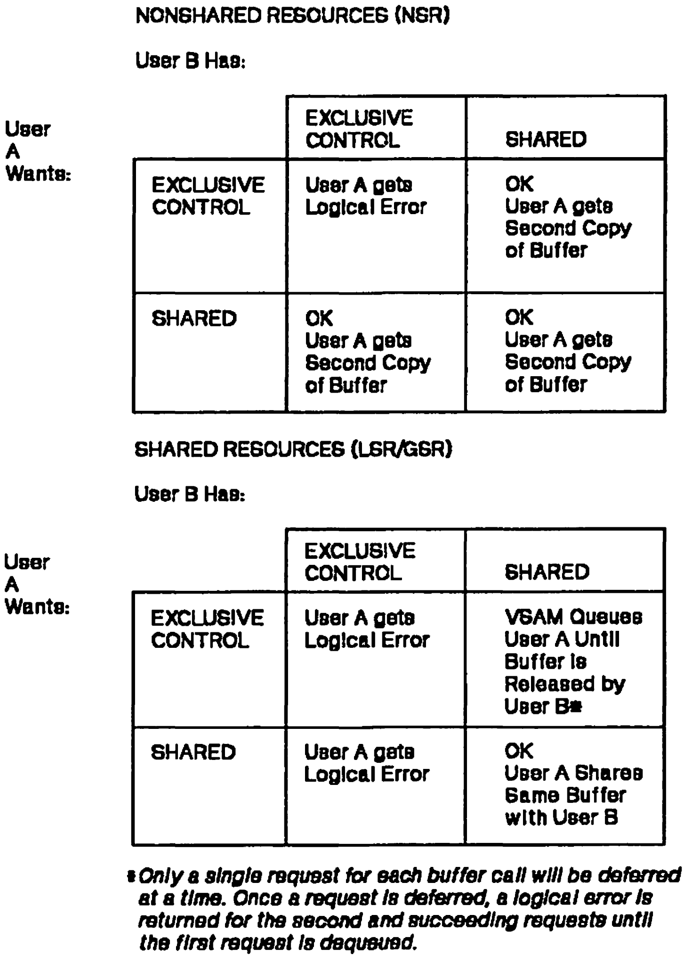
  For example:

  ```
  //DD1    DD    DSN=ABC
  //DD2    DD    DSN=ABC.PATH

          OPEN ACB1,DDN=DD1,DSN
          OPEN ACB2,DDN=DD2,DSN
  ```

Multiple ACBs must be in the same region, and they must be opening to the same base cluster. The connection occurs independently of the path selected to the base cluster. If the ATTACH macro is used to create a new task that will be processing a shared data set, allow the ATTACH keyword SZERO to default to YES or code SZERO = YES. This will cause subpool 0 to be shared with the subtasks. For more information on the ATTACH macro, see *System Macros and Facilities*. To ensure correct processing in the shared environment, all VSAM requests should be issued in the same key as the jobstep TCB key.

In this environment with a single control block, VSAM record management serializes updates to any single control interval and provides read and write integrity. When a control interval is not available for the type of user processing requested (shared or exclusive), VSAM record management returns a logical error code with an exclusive control error indicated in the RPL feedback code. When this occurs, you must decide whether to retry later or to free the resource causing the conflict. See Figure 27 for a diagram on exclusive control conflict feedback and results of different user requests.



NONSHARED RESOURCES (NSR)

User B Has:

User A Wants:	EXCLUSIVE CONTROL	SHARED
EXCLUSIVE CONTROL	User A gets Logical Error	OK User A gets Second Copy of Buffer
SHARED	OK User A gets Second Copy of Buffer	OK User A gets Second Copy of Buffer

SHARED RESOURCES (LSR/GSR)

User B Has:

User A Wants:	EXCLUSIVE CONTROL	SHARED
EXCLUSIVE CONTROL	User A gets Logical Error	VSAM Queues User A Until Buffer Is Released by User B*
SHARED	User A gets Logical Error	OK User A Shares Same Buffer with User B

****Only a single request for each buffer call will be deferred at a time. Once a request is deferred, a logical error is returned for the second and succeeding requests until the first request is dequeued.***

Figure 27. Exclusive Control Conflict Resolution

Preventing Deadlock in Exclusive Control

Contention for VSAM data (the contents of a control interval) can lead to deadlocks, in which a processing program is prevented from continuing because its request for data cannot be satisfied. A and B can engage as contenders in four distinct ways:

1. A wants exclusive control, but B has exclusive control. VSAM refuses A's request: A must either do without the data or retry the request.
2. A wants exclusive control, but B is only willing to share. VSAM queues A's request (without notifying A of a wait) and gives A use of the data when B releases it.
3. A wants to share, but B has exclusive control. VSAM refuses A's request: A must either do without the data or retry the request.
4. A wants to share, and B is willing to share. VSAM gives A use of the data, along with B.

VSAM's action in a contention for data rests on two assumptions:

- If a processing program has exclusive control of the data, it can update or delete it.
- If a processing program is updating or deleting the data, it has exclusive control. (The use of MRKBFR, MARK=OUT provides an exception to this assumption. A processing program can update the contents of a control interval without exclusive control of them.)

In 1 and 3 above, B is responsible for giving up exclusive control of a control interval by way of an ENDREQ, a MRKBFR with MARK=RLS, or a request for access to a different control interval. (The RPL that defines the ENDREQ, MRKBFR, or request is the one used to acquire exclusive control originally.)

Data Set Name Sharing

Data set name sharing is established by the ACB option (MACRF=DSN). To understand DSN sharing, you must understand a sphere and the base of the sphere and how they function.

A sphere is a VSAM cluster and its associated data sets. The cluster is originally defined with the access method services command DEFINE CLUSTER. The most common use of the sphere is to open a single cluster. The base of the sphere is the cluster itself. When opening a path (which is the relationship between an alternate index and base cluster) the base of the sphere is again the base cluster. Opening the alternate index as a data set results in the alternate index becoming the base of the sphere. In Figure 28, DSN is specified for each ACB and output processing is specified.

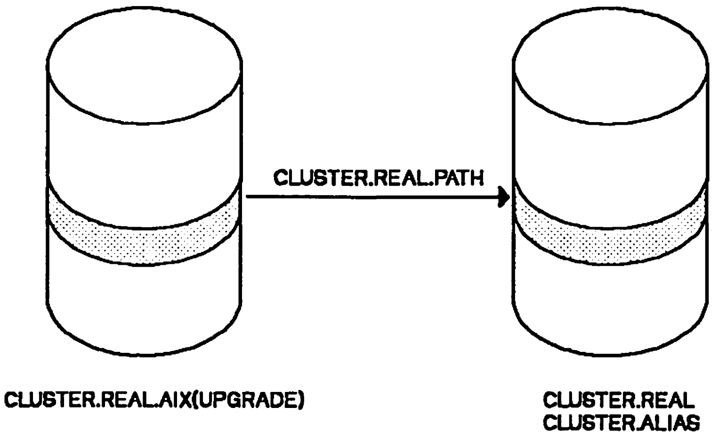


Figure 28. Relationship between the Base Cluster and the Alternate Index

Connecting Spheres: VSAM will connect an ACB to an existing control block structure for data set name sharing only when the base of the sphere is the same for both ACBs. The following three OPEN statements illustrate how information is added to a single control block structure, permitting data set name sharing.

1. OPEN ACB=(CLUSTER.REAL)
   - Builds control block structure for CLUSTER.REAL
   - Builds control block structure for CLUSTER.REAL.AIX

2. OPEN ACB=(CLUSTER.REAL.PATH)

   - Adds to existing structure for CLUSTER.REAL
   - Adds to existing structure for CLUSTER.REAL.AIX

3. OPEN ACB=(CLUSTER.ALIAS)

   Adds to existing structure for CLUSTER.REAL.

If you add this fourth statement, the base of the sphere changes, and multiple control block structures are created for the alternate index CLUSTER.REAL.AIX.

4. OPEN ACB=(CLUSTER.REAL.AIX)

   - Does not add to existing structure as the base of the sphere is not the same.

   - SHAREOPTIONS are enforced for CLUSTER.REAL.AIX since multiple control block structures exist.

To be compatible, both the new ACB and the existing control block structure must be consistent in their specification of the following processing options.

- The data set specification must be consistent in both the ACB and the existing control block structure. This means that an index of a KSDS, which is opened as an ESDS, does not share the same control block structure as the KSDS opened as a KSDS.

- The MACRF options DFR, UBF, ICI, CBIC, LSR, and GSR must be consistent. For example, if the new ACB and the existing structure both specify MACRF=DFR, the connection is made. If the new ACB specifies MACRF=DFR and the existing structure specifies MACRF=DFR,UBF, no connection is made.

If compatibility cannot be established, OPEN tries (within the limitations of the share options specified when the data set was defined) to build a new control block structure. If it cannot, OPEN fails.

When processing multiple subtasks sharing a single control block, concurrent GET and PUT requests are allowed. A control interval is protected for write operations using an exclusive control facility provided in VSAM record management. Other PUT requests to the same control interval are not allowed and a logical error is returned to the user issuing the request macro. Depending on the selected buffer option, nonshared (NSR) or shared (LSR/GSR) resources, GET requests to the same control interval as that being updated may or may not be allowed. Figure 27 on page 115 illustrates the exclusive control facility.

When a subtask issues OPEN to an ACB that will share a control block structure that may have been previously used, issue the POINT macro to obtain the position for the data set. In this case, it should not be assumed that positioning is at the beginning of the data set.

Cross-Region Sharing

The extent of data set sharing within one operating system depends on the data set disposition and the cross region share option specified when you define the data set. Independent job steps or subtasks in an MVS/XA operating system or multiple systems with global resource serialization (GRS) can access a VSAM data set simultaneously. To share a data set, each user must specify DISP=SHR in the data set's DD statement. The level of cross-region sharing allowed by VSAM is established (when the data set is defined) with the SHAREOPTIONS value:

- Cross-region SHAREOPTIONS 1: The data set can be shared by any number of users for read processing, *or* the data set can be accessed by only one user for read and write processing.
- Cross-region SHAREOPTIONS 2: The data set can be accessed by any number of users for read processing *and* it can also be accessed by one user for write processing.
- Cross-region SHAREOPTIONS 3: The data set can be fully shared by any number of users.
- Cross-region SHAREOPTIONS 4: The data set can be fully shared by any number of users, and buffers used for direct processing are refreshed for each request.

With options 3 and 4 you are responsible for maintaining both read and write integrity for the data the program accesses. These options require your program to use ENQ/DEQ to maintain data integrity while sharing the data set, including the OPEN and CLOSE processing. User programs that ignore the write integrity guidelines can cause VSAM program checks, lost or inaccessible records, uncorrectable data set failures, and other unpredictable results. These options place heavy responsibility on each user sharing the data set.

When your program requires that no updating from another control block structure occur before it completes processing of the requested data record, your program can issue an ENQ to obtain exclusive use of the VSAM data set. If your program completes processing, it can relinquish control of the data set with a DEQ. If your program is only reading data and not updating, it is probably a good practice to serialize the updates and have the readers wait while the update is occurring. If your program is updating, after the update has completed the ENQ/DEQ bracket, the reader must determine the required operations for control block refresh and buffer invalidation based on a communication mechanism or assume that everything is down-level and refresh each request.

The extent of cross-region sharing is affected by the use of DISP=SHR or DISP=OLD in the DD statement. If the data set's DD statement specifies DISP=OLD, only the dsname associated with the DD statement is exclusively controlled. In this case, only the cluster name is reserved for the OPEN routine's exclusive use. You can include DD statements with DISP=OLD for each of the cluster's components to reserve them as well. Doing this ensures that all resources needed to open the data set will be exclusively reserved before your task is initiated.

Protecting the cluster name with DISP processing and the components by VSAM OPEN SHAREOPTIONS is the normally accepted procedure. When a shared data set is opened with DISP=OLD, or is opened for create or reset processing, your program has exclusive control of the data set within your operating system.

Note: Scheduler "disposition" processing is the same for VSAM and non-VSAM data sets. This is the first level of share protection.

Read Integrity during Cross-Region Sharing

You are responsible for ensuring read integrity when the data set is opened for sharing with cross-region SHAREOPTIONS 2, 3, and 4. When your program issues a GET request, VSAM obtains a copy of the control interval containing the requested data record. Another program sharing the data set may also obtain a copy of the same control interval, and may update the data and write the control interval back into the data set. When this occurs, your program has lost read integrity. The control interval copy in your program's buffer is no longer the current copy.

The following should be considered when you are providing read integrity:

- Establish ENQ/DEQ procedures for all requests, read as well as write.
- Decide how to determine and invalidate buffers (index and/or data) that are possibly down level.
- With an entry-sequenced or relative record data set, do not allow secondary allocation. If you do allow secondary allocation you should provide a communication mechanism to the read-only tasks that the extents are increased, force a CLOSE, and then issue another OPEN. Providing a buffer refresh mechanism for index I/O will accommodate secondary allocations for a key-sequenced data set.
- With an entry-sequenced or relative record data set, you must also use the VERIFY macro prior to the GET macro to update possible down-level control blocks.
- Generally, the loss of read integrity results in down-level data records and erroneous no-record-found conditions.

Invalidating Data and Index Buffers

To invalidate index buffers, you could use the following technique.

- In the ACB, specify:

  STRNO>1.
  MACRF=NSR to indicate non-shared resources.
  Let the value for BUFNI default to the minimum.

- Ensure that your index is a multilevel index.
- Ensure that all requests are for positioning by specifying the following:

  GET RPL OPTCD=DIR
  POINT
  PUT RPL OPTCD=NUP

To invalidate data buffers, ensure that all requests are for positioning by specifying the following:

- GET/PUT RPL OPTCD=(DIR,NSP) followed by ENDREQ
- POINT GET/PUT RPL OPTCD=SEQ followed by ENDREQ

Write Integrity during Cross-Region Sharing

You are responsible for ensuring write-integrity if a data set is opened with cross-region SHAREOPTIONS 3 or 4.

When an application program issues a "direct" or "skip-sequential" PUT-for-update or no-update, (RPL OPTCD=DIR|SKP), the updated control interval is written to direct access storage when you obtain control following a synchronous request (RPL OPTCD=SYN) or following the CHECK macro from an asynchronous request (RPL OPTCD=ASY). To force direct access I/O for a sequential PUT (RPL OPTCD=SEQ), the application program must issue an ENDREQ or MRKBFR TYPE=OUT.

Note: Whenever an ENDREQ is issued, the return code in Register 15 should be checked to determine if there is an error. If there is an error, normal check processing should be performed to complete the request.

The considerations that apply to read integrity also apply to write integrity. The serialization for read could be implemented as a shared ENQ and for write as an exclusive ENQ. You must ensure that all I/O is performed to DASD before dropping the serialization mechanism (usually the DEQ).

Cross-System Sharing

Use the following share options when you define a data set that must be accessed and/or updated by more than one operating system simultaneously:

- **Cross-system SHAREOPTION 3:** The data set may be fully shared. With this option, the access method uses the control block update facility (CBUF) to provide assistance.
- **Cross-system SHAREOPTION 4:** The data set may be fully shared, and buffers used for direct processing are refreshed for each request. Output processing is limited to update and/or add processing that does not change either the high-used RBA or the RBA of the high key data control interval if DISP=SHR is specified.

You must assume full responsibility for read and write integrity. Incorrect write integrity processing can cause access method program checks, lost or inaccessible records, uncorrectable data set failures, and other unpredictable results. These options place very heavy responsibility upon you and should not be treated lightly. The RESERVE and DEQ macros are required with these options to maintain data set integrity.

When sharing data sets in a cross region or system environment, you should run VERIFY before opening a data set. VERIFY updates the catalog description of the data set and discards any erroneous information that may result from improper closing of the data set. This information and its effects may not be evident to all systems sharing the data set. VERIFY eliminates the problem if it is run as the first step of a jobstream.

When the data set is shared under cross-system SHAREOPTIONS 4, regardless of cross-region requests, VSAM does not allow changes to high-used and high-key RBAs. In addition, VSAM provides assistance to the application to aid in preserving the integrity of the data:

- 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.
- The data and sequence-set control interval buffers are marked invalid following I/O operation to a direct access storage device.

Job steps of two or more systems may gain access to the same data set regardless of the disposition specified in each step's JCL. To get exclusive control of a volume, a task in one system must issue a RESERVE macro. For other methods of obtaining exclusive control using global resource serialization (GRS), see *Global Resource Serialization.*

Control Block Update Facility (CBUF)

Whenever a data set is opened with cross-region SHAREOPTION 3 or 4, cross-system SHAREOPTION 3, and DISP=SHR, VSAM record management maintains a copy of the critical control block data in common storage. The control block data in the common storage area is available to each program (each memory) sharing the data set. The common storage area is available only to regions within your operating system. Communicating this information to another operating system is your responsibility.

CBUF eliminates the restriction that prohibits control area splits under cross-region SHAREOPTION 4. Therefore, you do not need to restrict code to prevent control area splits, or allow for the control area split error condition. The restriction to prohibit control area splits for cross-systems SHAREOPTION 4 still exists.

If the data set has cross-system SHAREOPTION 4, but does not reside on shared DASD when it is opened, the data set is still processed as a cross-system SHAREOPTION 4 data set on shared DASD; that is, CBUF processing is not provided. When a key-sequenced data set has cross-system SHAREOPTION 4, control area splits are prevented; also, split of the control interval containing the high key of a key range (or data set) is prevented. With control interval access, adding a new control interval is prevented.

Cross-system sharing can be accomplished by sending the VSAM shared information (VSI) blocks to the other host at the conclusion of each output request. Generally, the VSIs will not have changed and only a check occurs.

It should be noted that the SHAREOPTION 3 user must continue to provide read/write integrity. Although VSAM ensures that SHAREOPTION 3 and 4 users will have correct control block information if serialization is done correctly, the SHAREOPTION 3 user will not get the buffer invalidation that will occur with SHAREOPTION 4.

When improved control interval processing is specified with SHAREOPTION 3 or 4, the data set can be opened; however, if another control block structure extends the data set, the control block structure using improved control interval processing will not be updated unless it is closed and reopened.

Figure 29 illustrates how the SHAREOPTIONS specified in the catalog and the disposition specified on the DD statement interact to affect the type of processing.

(CR CS) when DISP=SHR[1]	Functions Provided
(3 3)	CBUF
(3 4)	Data and sequence set buffers invalidated. CA split not allowed.
(4 3)	Data and index component buffers invalidated. CBUF
(4 4)	Data and sequence set buffers invalidated. CA split not allowed.

Legend:

CA = Control Area
CR = Cross-Region
CS = Cross-System
CBUF = Control Block Update Facility
Buffer invalidated = Invalidation of buffers is automatic

[1] When DISP=OLD is specified or the data set is in create or reset mode (regardless of the disposition specified), the share options specified in the catalog are ignored. The data set is processed under the rules for SHAREOPTIONS(1 3). OPEN ensures that the user has exclusive control of the data set within a single system. If the data set can be shared between systems, VSAM does nothing to ensure that another system is not accessing the data set concurrently. With cross-system sharing, the user must ensure that another system is not accessing the data set before specifying DISP=OLD.

Figure 29. Relationship between SHAREOPTIONS and VSAM Functions

User Considerations with CBUF Processing

If your program shares a data set defined with SHAREOPTIONS(3 3) or SHAREOPTIONS (4 3), you should note that:

- In a shared environment, VSAM does not allow you to process the data set in an initial load or reset mode (create). VSAM forces your data set to be processed as though it were defined with SHAREOPTIONS(1 3).
- A user program cannot share a system data set (for example, the master catalog, page space data sets, SYS1. data sets, duplex data sets, and swap data sets).
- The user's program must serialize all VSAM requests against the data set, using ENQ/DEQ (or a similar function).
- The user's program must ensure that all VSAM resources are acquired and released within ENQ/DEQ protocol to:

  - Force VSAM to write sequential update and insert requests.
  - Release VSAM's positioning within the data set.

- VSAM invalidates buffers used with SHAREOPTIONS 4 data sets, but does not invalidate buffers used with SHAREOPTIONS 3 data sets. When a buffer is marked invalid (it is invalidated), it is identified as a buffer that VSAM must refresh (read in a fresh copy of the control interval from DASD) before your program can use the buffer's contents.

- Programs that use GSR and LSR can invalidate and force writing of buffers using the MRKBFR and WRTBFR macros.

- Because programs in many regions can share the same data set, an error that occurs in one region may 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.

- When a VSAM data set requires additional space, VSAM end-of-volume processing acquires new extents for the VSAM data set, 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 so that this new space is accessible by all regions using this VSAM data set. If an abend or unexpected error occurs, which prevents this space allocation from being completed, all regions will be prevented from further extending the data set. To obtain additional space, you must close the VSAM data set in all regions, then reopen it.

- To correct the data set's control blocks following an abnormal termination (ABEND), the ACBSWARN flag should be off in the ACB (the default) that is reopening the data set in order to allow implicit VERIFY processing to take place. If implicit verify is suppressed (ACBSWARN flag is on), the VERIFY macro should be issued to update the data set's control blocks. A subsequent CLOSE will update the catalog record. Updating of the data set's catalog record is bypassed following an abnormal termination.

- Implicit VERIFY is invoked by the open-for-output indicator in the catalog. When a data set is opened and the open-for-output indicator is already on, CLOSE processing resets the indicator only if the data set was just opened for output; otherwise it leaves the bit on.

- Data sets shared in a cross-region or cross-system environment should either use access method services VERIFY command or issue the VERIFY macro from within the application program.

Issuing a Checkpoint with Shared Data Sets

If you issue a checkpoint or if a restart occurs, none of the VSAM data sets open in your region at that time may be using CBUF processing. If you issue checkpoints, you should open the VSAM data sets that are eligible for CBUF processing with a disposition of OLD, or CLOSE them prior to the checkpoint. Note that, if an alternate index was using CBUF processing, the associated base cluster and any other paths open over that base cluster must also be closed prior to the checkpoint, even if they are not using CBUF processing.

Techniques of Sharing—Examples

Cross-Region Sharing

To maintain write integrity for the data set, your program must ensure that there is no conflicting activity against the data set until your program completes updating the control interval. Conflicting activity may be divided into two categories:

1. A data set that is totally preformatted and the only write activity is update-in-place.

   In this case, the sharing problem is simplified by the fact that data cannot change its position in the data set. The lock that must be held for any write operation (GET/PUT RPL OPTCD=UPD) is the unit of transfer that is the control interval. It is your responsibility to associate a lock with this unit of transfer; the record key is not sufficient unless only a single logical record resides in a control interval.

   The following is an example of the required procedures:

   a. Issue a GET for the RPL that has the parameters OPTCD=(SYN,KEY,NUP,DIR),ARG=MYKEY.

   b. Determine the RBA of the control interval (RELCI) where the record resides. This is based on the RBA field supplied in the RPL(RPLDDDD).

      ```
      RELCI=CISIZE * integer-part-of (RPLDDDD / CISIZE)
      ```

   c. Enqueue MYDATA.DSNAME.RELCI (the calculated value).

   d. Issue a GET for the RPL that has the parameters OPTCD=(SYN,KEY,UPD,DIR),ARG=MYKEY.

   e. Issue a PUT for the RPL that has the parameters OPTCD=(SYN,KEY,DIR,NSP).

   f. Issue an ENDREQ. This forces I/O to DASD, and will drop the position maintained by NSP and cause data buffer invalidation.

   g. Dequeue MYDATA.DSNAME.RELCI.

2. A data set in which record additions and updates with length changes are permitted.

   In this case, the minimum locking unit is a control area to accommodate control interval splits. A higher level lock must be held during operations involving a control area split. The split activity must be serialized at a data set level. To implement a multi-level locking procedure, you must be prepared to provide additional programming using the information provided during VSAM JRNAD processing. This exit is responsible for determining the level of data movement and obtaining the appropriate lock(s).

   Higher concurrency can be achieved by a hierarchy of locks. Based on the particular condition, one or more of the locking hierarchies must be obtained.

Lock	**Condition**
Control Interval	Updating a record in place or adding a record to a control interval without causing a split.
Control Area	Adding a record or updating a record with a length change, causing a control interval split, but not a control area split.
Data Set	Adding a record or updating a record with a length change, causing a control area split.

The following is a basic procedure to provide the necessary protection while incurring the penalty of locking all updates at the data set level:

```
SHAREOPTION = (4 3)       CBUF processing
Enqueue MYDATA.DSNAME     Shared for read only;
                          exclusive for write

Issue VSAM request macros
      .
      .
      .
Dequeue MYDATA.DSNAME
```

In any sharing situation, it is a general rule that all resources be obtained and released between the locking protocol. All positioning must be released by using all direct requests or by issuing the ENQREQ macro prior to ending the procedure with the DEQ.

Cross-System Sharing

With cross-system SHAREOPTIONS 3, you have the added responsibility of passing the VSAM shared information (VSI) and invalidating data and/or index buffers. This may be done by the use of an informational control record as the low-key or first record in the data set. The following information is required to accomplish the necessary index record invalidation:

1. Number of data control interval splits and index updates for sequence set invalidation

2. Number of data control area splits for index set invalidation

All data buffers should always be invalidated. In order to perform selective buffer invalidation, an internal knowledge of the VSAM control blocks is required.

Your program must serialize the following types of requests (precede the request with an ENQ and, when the request completes, issue a DEQ):

- All PUT requests.
- POINT, GET-direct-NSP, GET-skip, and GET-for-update requests that are followed by a PUT-insert or PUT-update request.
- VERIFY requests. When VERIFY is executed by VSAM, your program must have exclusive control of the data set.
- Sequential GET and PUT requests.

User Access to VSAM Shared Information

You can code the following instructions to get the length and address of the data to be sent to another processor:

- 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 may be located through the AMB. 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 the use of ENQ/DEQ to prevent simultaneous updates to the transmitted data.

If the data set can be shared between MVS/XA operating systems, a user's program in another system may concurrently access the data set. Before you open the data set specifying DISP=OLD, it is your responsibility to protect across systems with ENQ/DEQ in the UCB option or equivalent functions.

End of General-Use Programming Interface

Chapter 10. Sharing Resources among Data Sets

General-Use Programming Interface

This chapter is intended to help you share resources among your data sets. It contains general-use programming interfaces, which allow you to write programs that use the services of MVS/XA Data Facility Product.

VSAM has a set of macros that enables you to share I/O buffers and I/O-related control blocks among many VSAM data sets. In VSAM, an I/O buffer is a virtual storage area from which the contents of a control interval are read and written. Sharing these resources optimizes their use, reducing the requirement for virtual storage and therefore reducing paging of virtual storage.

Sharing these resources is not the same as sharing a data set itself (that is, sharing among different tasks that independently open it). Data set sharing can be done with or without sharing I/O buffers and I/O-related control blocks. For a discussion of data set sharing. see Chapter 9, "Sharing a VSAM Data Set" on page 113.

There are also macros that allow you to manage I/O buffers for shared resources.

Sharing resources does not improve sequential processing. VSAM does not automatically position itself at the beginning of a data set opened for sequential access, because placeholders belong to the resource pool, not to individual data sets. When you share resources for sequential access, positioning at the beginning of a data set has to be specified explicitly with the POINT macro or the direct GET macro with RPL OPTCD=NSP. You may not use a resource pool to load records into an empty data set.

Providing a Resource Pool

To share resources, follow this procedure:

1. Use the BLDVRP macro to build a resource pool.
2. Code a MACRF operand in the ACB and use OPEN to connect your data sets to the resource pool.
3. After you have closed all the data sets, use the DLVRP macro to delete the resource pool.

Building a Resource Pool: BLDVRP

Issuing BLDVRP causes VSAM to share the I/O buffers and I/O-related control blocks of data sets whose ACBs indicate the corresponding option for shared resources. Control blocks are shared automatically; you may control the sharing of buffers.

You may share resources locally or globally:

- **LSR (local shared resources)**

  You can build up to 16 data resource pools and 16 index resource pools in one address space. Each resource pool must be built individually. ‚The data pool must exist before the index pool with the same share pool identification can be built. The parameter lists for these multiple LSR pools can reside above or below 16 megabytes. The BLDVRP macro RMODE31 parameter indicates where VSAM is to obtain virtual storage when the LSR pool control blocks and data buffers (specified by the SHRPOOL keyword) are built.

  These resource pools are built with the BLDVRP macro TYPE=LSR and DATA|INDEX specifications. Specifying MACRF=LSR on the ACB or GENCB-ACB macros causes the data set to use the LSR pools built by the BLDVRP macro. The DLVRP macro processes both the data and index resource pools.

  The separate index resource pools are not supported for GSR.

- **GSR (global shared resources)**

  All address spaces for a given protection key in the system share one resource pool. Only one resource pool can be built for each of the protection keys 0 through 7. With GSR, an access method control block and all related request parameter lists, exit lists, data areas, and extent control blocks must be in the common area of virtual storage with protection key the same as that of the resource pool. To get storage in the common area with that protection key, issue the GETMAIN macro while in that key, for storage in subpool 241.

  Generate ACBs, RPLs, and EXLSTs with the GENCB macro—code the WAREA and LENGTH operands. The program that issues macros related to that global resource pool must be in supervisor state with the same key. (The macros are: BLDVRP, CHECK, CLOSE, DLVRP, ENDREQ, ERASE, GENCB, GET, GETIX, MODCB, MRKBFR, OPEN, POINT, PUT, PUTIX, SCHBFR, SHOWCB, TESTCB, and WRTBFR. The SHOWCAT macro is not related to a resource pool, because it is issued independently of an opened data set.)

You may have both a global resource pool and local resource pools. Tasks in an address space that have a local resource pool may use either the global resource pool, under the restrictions described above, or the local resource pool. There may be multiple buffer pools based on buffer size for each resource pool.

To share resources locally, a task in the address space issues BLDVRP TYPE=LSR, DATA|INDEX. To share resources globally, a system task issues BLDVRP TYPE=GSR. The program that issues BLDVRP TYPE=GSR must be in supervisor state with key 0 to 7.

When you issue BLDVRP, you specify the size and number of buffers in each buffer pool and the number of buffer pools in the resource pool. For the data pool or the separate index pool at OPEN time, a data set is assigned the one buffer pool with buffers of the appropriate size—either the exact control interval size requested, or the next larger size available.

Deciding How Big a Resource Pool to Provide

You have to provide a resource pool before any clusters or alternate indexes are opened to use it. To specify the BUFFERS, KEYLEN, and STRNO operands of the BLDVRP macro, you must know the size of the control intervals, data records (if spanned), and key fields in the components that will use the resource pool. You must also know how the components are processed. You can use the SHOWCAT and SHOWCB macros to get this information.

For each VSAM cluster that will share the resource pool you are building, follow this procedure:

1. Determine the number of concurrent requests you expect to process. This number represents 'STRNO' for the cluster.
2. Specify 'BUFFERS=(SIZE(STRNO+1))' for the data component of the cluster.
    - If the cluster is a key-sequenced cluster and the index CISZ (control interval size) is the same as the data CISZ, change the specification to 'BUFFERS=(SIZE(2 X STRNO)+1)'.
    - If the index CISZ is not the same as the data component CISZ, specify 'BUFFERS=(dataCISZ(STRNO+1),indexCISZ(STRNO))'.

Following this procedure will provide the minimum number of buffers needed to support concurrently active STRNO strings. An additional string is not dynamically added to a shared resource pool. The calculation can be repeated for each cluster which will share the resource pool, including associated alternate index clusters and clusters in the associated alternate index upgrade sets.

For each cluster component having a different CISZ, add another ',SIZE(NUMBER)' range to the 'BUFFERS=' specification. Note that the data component and index component buffers may be created as one set of buffers, or, by use of the 'TYPE=' statement, may be created in separate index and data buffer sets.

Additional buffers may be added to enhance performance of applications requiring read access to data sets by reducing I/O requirements. You should also consider the need for cross region or cross system sharing of the data sets where modified data buffers must be written frequently to enhance read and update integrity. A large number of buffers is not usually an advantage in such environments. In some applications where a resource pool is shared by multiple data sets and not all data set strings are active concurrently, less than the recommended number of buffers may produce satisfactory results.

If the specified number of buffers is not adequate, VSAM will return a logical error indicating the out of buffer condition.

Displaying Information about an Unopened Data Set

The SHOWCAT macro enables you to get information about a component before its cluster or alternate index is opened. The program that is to issue BLDVRP can issue SHOWCAT on all the components to find out the sizes of control intervals, records, and keys. This information enables the program to calculate values for the BUFFERS and KEYLEN operands of BLDVRP.

A program need not be in supervisor state with protection key 0 to 7 to issue SHOWCAT, even though it must be in supervisor state and in protection key 0 to 7 to issue BLDVRP TYPE=GSR.

The SHOWCAT macro is described in *Catalog Administration Guide.*

Displaying Statistics about a Buffer Pool

The statistics cannot be used to redefine the resource pool while it is in use. You have to make adjustments the next time you build it.

The use of SHOWCB to display an ACB is described in "Manipulating Control Block Contents" on page 44. If the ACB has MACRF=GSR, the program that issues SHOWCB must be in supervisor state with protection key 0 to 7. A program check can occur if SHOWCB is issued by a program that is not in supervisor state with the same protection key as that of the resource pool.

For buffer pool statistics, the keywords described below are specified in the FIELDS operand. These fields may be displayed only after the data set described by the ACB is opened. Each field requires one fullword in the display work area:

Field	**Description**
BFRFND	The number of requests for retrieval that could be satisfied without an I/O operation (the data was found in a buffer)
BUFRDS	The number of reads to bring data into a buffer
NUIW	The number of nonuser-initiated writes (writes that VSAM was forced to do because no buffers were available for reading the contents of a control interval)
STRMAD	The maximum number of placeholders currently active for the resource pool (for all the buffer pools in it)
UIW	The number of user-initiated writes (PUTs not deferred or WRTBFRs, see "Deferring Write Requests" on page 132).

Connecting a Data Set to a Resource Pool: OPEN

You cause a data set to use a resource pool that was built by BLDVRP by specifying LSR or GSR in the MACRF operand of the data set's ACB before you open the data set.

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

NSR, the default, indicates the data set does not use shared resources. LSR indicates it uses the local resource pool. GSR indicates it uses the global resource pool.

If the VSAM control blocks and data buffers reside above 16 megabytes, the RMODE31=ALL operand must be specified in the ACB before OPEN is issued. If the OPEN parameter list or the VSAM ACB resides above 16 megabytes, the MODE=31 parameter of the OPEN macro must also be coded.

When an ACB indicates LSR or GSR, VSAM ignores its BSTRNO, BUFNI, BUFND, BUFSP, and STRNO operands because VSAM will use the existing resource pool for the resources associated with these parameters.

To connect LSR pools with a SHRPOOL identification number other than SHRPOOL=0, you must use the SHRPOOL parameter of the ACB macro to indicate which LSR pool you are connecting.

If more than one ACB is opened for LSR processing of the same data set, the LSR pool identified by the SHRPOOL parameter for the first ACB will be used for all subsequent ACBs.

For a data set described by an ACB with MACRF=GSR, the ACB and all related RPLs, EXLSTs, ECBs, and data areas must be in the common area of virtual storage with the same protection key as that of the resource pool.

Deleting a Resource Pool: DLVRP

After all data sets using a resource pool are closed, delete the resource pool by issuing the DLVRP (delete VSAM resource pool) macro. Failure to delete a local resource pool causes virtual storage to be lost until the end of the job step or TSO session. This loss is protected with a global resource pool. If the address space that issued BLDVRP terminates without having issued DLVRP, the system deletes the global resource pool when its use count is 0.

To delete an LSR pool with a SHRPOOL identification number other than SHRPOOL=0, you must use the SHRPOOL parameter to indicate which resource pool you are deleting. If both a data resource pool and an index resource pool have the same SHRPOOL number, both will be deleted.

If the DLVRP parameter list is to reside above 16 megabytes, the MODE=31 parameter must be coded.

Managing I/O Buffers for Shared Resources

Managing I/O buffers includes:

- Deferring writes for direct PUT requests, which reduces the number of I/O operations.
- Writing buffers that have been modified by related requests.
- Locating buffers that contain the contents of specified control intervals.
- Marking a buffer to be written without issuing a PUT.
- When your program accesses an invalid buffer, VSAM refreshes the buffer (that is, reads in a fresh copy of the control interval) before making its contents available to your program.

Managing I/O buffers should enable you to speed up direct processing of VSAM data sets that are accessed randomly. You probably will not be able to speed up sequential processing or processing of a data set whose activity is consistently heavy.

Deferring Write Requests

VSAM automatically defers writes for sequential PUT requests. It normally writes out the contents of a buffer immediately for direct PUT requests. With shared resources, you can cause writes for direct PUT requests to be deferred. Buffers are finally written out:

- When you issue the WRTBFR macro.
- When VSAM needs a buffer to satisfy a GET request.
- When a data set using a buffer pool is closed. (Temporary CLOSE is ineffective against a data set that is sharing buffers, and ENDREQ does not cause buffers in a resource pool to be written.)

Deferring writes saves I/O operations when subsequent requests can be satisfied by the data in the buffer pool. If you are going to update control intervals more than once, data processing performance will be improved by deferring writes.

You indicate 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},...),...
```

The DFR option is incompatible with SHAREOPTIONS 4. (SHAREOPTIONS is a parameter of the DEFINE command of access method services. It is described in *Access Method Services Reference*.) A request to open a data set with SHAREOPTIONS 4 for deferred writes is rejected.

VSAM notifies the processing program when an unmodified buffer has been found for the current request and there will be no more unmodified buffers into which to read the contents of a control interval for the next request. (VSAM will be forced to write a buffer to make a buffer available for the next I/O request.) VSAM sets register 15 to 0 and puts 12 (X'0C') in the feedback field of the RPL that defines the PUT request detecting the condition.

VSAM also notifies the processing program when there are no buffers available to be assigned to a placeholder for a request. This is a logical error (register 15 contains 8 unless an exit is taken to a LERAD routine); the feedback field in the RPL contains 152 (X'98'). You may retry the request; it gets a buffer if one is freed.

Relating Deferred Requests by Transaction ID

You can relate action requests (GET, PUT, and so forth) according to transaction by specifying the same ID in the RPLs that define the requests.

The purpose of relating the requests that belong to a transaction is to enable WRTBFR to cause all the modified buffers used for a transaction to be written. When the WRTBFR request is complete, the transaction is physically complete.

```
RPL      TRANSID=number,...
```

TRANSID specifies a number from 0 to 31. The number 0, which is the default, indicates that requests defined by the RPL are not associated with other requests. A number from 1 to 31 relates the request(s) defined by this RPL to the request(s) defined by other RPLs with the same transaction ID.

You can find out what transaction ID an RPL has by issuing SHOWCB or TESTCB.

```
SHOWCB     FIELDS=([TRANSID],...),...
```

TRANSID requires one fullword in the display work area.

```
TESTCB     TRANSID=number,...
```

If the ACB to which the RPL is related has MACRF=GSR, the program that issues SHOWCB or TESTCB must be in supervisor state with the same protection key as that of the resource pool. With MACRF=GSR specified in the ACB to which the RPL is related, a program check can occur if SHOWCB or TESTCB is issued by a program that is not in supervisor state with protection key 0 to 7. For more information on the use of SHOWCB and TESTCB, see "Manipulating Control Block Contents" on page 44.

Writing Buffers Whose Writing Is Deferred: WRTBFR

If any PUTs to a data set using a shared resource pool are deferred, you can use the WRTBFR (write buffer) macro to write:

- All modified unwritten index and data buffers for a given data set
- All modified unwritten index and data buffers in the resource pool
- The least recently used modified buffers in each buffer pool of the resource pool
- All buffers that were modified by requests with the same transaction ID
- A buffer, identified by an RBA value, that has been modified and has a use count of zero

You can specify the DFR option in an ACB without using WRTBFR to write buffers—a buffer is written when VSAM needs one to satisfy a GET request. or all modified buffers are written when the last of the data sets that uses them is closed.

Besides using WRTBFR to write buffers whose writing is deferred, you can use it to write buffers that are marked for output with the MRKBFR macro, which is described in "Marking a Buffer for Output: MRKBFR" on page 135.

Using WRTBFR can improve performance, if you schedule WRTBFR to overlap other processing.

When sharing the data set with a user in another region, your program might want to write the contents of a specified buffer without writing all other modified buffers. Your program issues the WRTBFR macro to search your buffer pool for a buffer containing the specified RBA. If found, the buffer is examined to verify that it is modified and has a use count of zero. If so, VSAM writes the contents of the buffer into the data set.

Note: Before using WRTBFR TYPE=CHK|TRN|DRBA, be sure to release all buffers. (For details about releasing buffers, see "Multistring Processing" on page 52.) If one of the buffers is not released, VSAM defers processing until the buffer is released.

Handling Exits to Physical Error Analysis Routines

With deferred writes of buffers, a processing program continues after its PUT request has been completed, even though the buffer has not been written. The processing program is not synchronized with a physical error that occurs when the buffer is finally written. A processing program that uses MRKBFR MARK=OUT is also not synchronized with a physical error. An EXCEPTION or a SYNAD exit routine must be supplied to analyze the error.

The ddname field of the physical error message identifies the data set that was using the buffer, but, because the buffer might have been released, its contents might be unavailable. You can provide a JRNAD exit routine to record the contents of buffers for I/O errors. It can be coordinated with a physical error analysis routine to handle I/O errors for buffers whose writing has been deferred. If a JRNAD exit routine is used to cancel I/O errors during a transaction, the physical error analysis routine will get only the last error return code. See *Data Facility Product: Customization* for more information on the SYNAD and JRNAD exit routines.

Using the JRNAD Exit with Shared Resources

VSAM takes the JRNAD exit for the following reasons when the exit is associated with a data set whose ACB has MACRF=LSR or GSR:

- A data or index control interval buffer has been modified and is about to be written.
- A physical error occurred. VSAM takes the JRNAD exit first—your routine can direct VSAM to bypass the error and continue processing or to terminate the request that occasioned the error and proceed with error processing.
- A control interval or a control area is about to be split for a key-sequenced data set. Your routine can cancel the request for the split and leave VSAM. An example of using JRNAD exit for this purpose is given in *Data Facility Product: Customization.*

See *Data Facility Product: Customization* for information describing the contents of the registers when VSAM exists to the JRNAD routine, and the fields in the parameter list pointed to by register 1.

Accessing a Control Interval with Shared Resources

Control interval access is not allowed with shared resources.

Locating an RBA in a Buffer Pool: SCHBFR

When a resource pool is built, the buffers in each buffer pool are numbered from 1 through the number of buffers in each buffer pool. At a given time, several buffers in a buffer pool may hold the contents of control intervals for a particular data set. These buffers may or may not contain RBAs of interest to your processing program. The SCHBFR macro enables you to find out. You specify in the ARG operand of the RPL that defines SCHBFR the address of an 8-byte field that contains the first and last control interval RBAs of the range you are interested in.

The buffer pool to be searched is the one used by the data component defined by the ACB to which your RPL is related. If the ACB names a path, VSAM searches the buffer pool used by the data component of the alternate index. (If the path is defined over a base cluster alone, VSAM searches the buffer pool used by the data component of the base cluster.) VSAM begins its search at the buffer you specify and continues until it finds a buffer that contains an RBA in the range or until the highest numbered buffer is searched.

For the first buffer that satisfies the search, VSAM returns its address (OPTCD=LOC) or its contents (OPTCD=MVE) in the work area whose address is specified in the AREA operand of the RPL and returns its number in register 0. If the search fails, Register 0 is returned with the user specified buffer number and a one-byte SCHBFR code of X'0D'. To find the next buffer that contains an RBA in the range, issue SCHBFR again and specify the number of the next buffer after the first one that satisfied the search. You continue until VSAM indicates it found no buffer that contains an RBA in the range or until you reach the end of the pool.

Finding a buffer that contains a desired RBA does not get you exclusive control of the buffer. You may get exclusive control only by issuing GET for update. SCHBFR does not return the location or the contents of a buffer that is already under the exclusive control of another request.

Marking a Buffer for Output: MRKBFR

You locate a buffer that contains the RBA you are interested in by issuing a SCHBFR macro, a read-only GET, or a GET for update. When you issue GET for update, you get exclusive control of the buffer. Whether you have exclusive control or not, you can mark the buffer for output by issuing the MRKBFR macro with MARK=OUT and then change the buffer's contents. Without exclusive control, you should not change the control information in the CIDF or RDFs (do not change the record lengths).

MRKBFR MARK=OUT, indicates that the buffer's contents are modified. You must modify the contents of the buffer itself—not a copy. Consequently, when you issue SCHBFR or GET to locate the buffer, you must specify RPL OPTCD=LOC. (If you use OPTCD=MVE, you get a copy of the buffer but do not learn its location.) The buffer is written when a WRTBFR is issued or when VSAM is forced to write a buffer to satisfy a GET request.

If you are sharing a buffer or have exclusive control of it, you can release it from shared status or exclusive control with MRKBFR MARK=RLS. If the buffer was marked for output, MRKBFR with MARK=RLS does not nullify it; the buffer is eventually written. Sequential positioning is lost. MRKBFR with MARK=RLS is similar to the ENDREQ macro.

Summary of Restrictions for Shared Resources

Restrictions for using the LSR and GSR options are:

- Empty data sets cannot be processed (that is, loaded).
- Multiple LSR pools in an address space are obtained by using the SHRPOOL parameter of the BLDVRP macro to identify each LSR pool.
- Control interval access cannot be used.
- Control blocks in common (CBIC) cannot be used.
- User buffering is not allowed (ACB MACRF=UBF).

- Writes for data sets with SHAREOPTIONS 4 cannot be deferred (ACB MACRF=DFR).
- Request parameter lists for MRKBFR, SCHBFR, and WRTBFR cannot be chained (the NXTRPL operand of the RPL macro is ignored).
- For sequential access, positioning at the beginning of a data set must be explicit: with a POINT macro or a direct GET macro with RPL OPTCD=NSP.
- Temporary CLOSE and ENDREQ do not cause buffers to be written if MACRF=DFR was specified in the associated ACB.
- With GSR, an ACB and all related RPLs, EXLSTs, data areas, and ECBs must be stored in the common area of virtual storage with protection key 0 to 7; all VSAM requests related to the global resource pool may be issued only by a program in supervisor state with protection key 0 to 7 (the same as that of the resource pool).
- Checkpoints cannot be taken for data sets whose resources are shared in a global resource pool. When a program in an address space that opened a data set whose ACB has MACRF=GSR issues the CHKPT macro, 8 is returned in register 15. If a program in another address space issues the CHKPT macro, the checkpoint is taken, but only for data sets that are not using the global resource pool.

  Checkpoint/restart can be used with data sets whose resources are shared in a local resource pool, but the restart program does not reposition for processing at the point where the checkpoint occurred—processing is restarted at a data set's highest used RBA. For information about restarting the processing of VSAM data sets, see *Checkpoint/Restart User's Guide*.
- If a physical I/O error is encountered while writing a control interval to the direct access device, the buffer remains in the resource pool. The write-required flag (BUFCMW) and associated mod bits (BUFCMDBT) are turned off, and the BUFC is flagged in error (BUFCER2=ON). The buffer is not replaced in the pool, and buffer writing is not attempted. To release this buffer for reuse, a WRTBFR macro with TYPE=DS can be issued or the data set can be closed (CLOSE issues the WRTBFR macro).

End of General-Use Programming Interface

Chapter 11. User-Written Exit Routines

User-written routines may be supplied to:

- Analyze logical errors (LERAD routine)
- Analyze physical errors (SYNAD routine)
- Perform end-of-data processing (EODAD routine)
- Record transactions made against a data set (JRNAD routine)
- Perform special user processing (UPAD routine)
- Perform user security verification (USVR routine)
- Perform datestamp processing (IDATMSTP program)

For more information about user-written exit routines, refer to *Data Facility Product: Customization*.

Chapter 12. Checking a VSAM Key-Sequenced Data Set Cluster for Errors

This chapter describes how the service aid, EXAMINE, is used to analyze a VSAM key-sequenced data set (KSDS) cluster for structural errors. The topics discussed are:

- Types of data sets that can be tested by EXAMINE
- Types of EXAMINE end users
- Selection criteria for running the tests INDEXTEST and DATATEST
- Controlling message printout
- Skipping the DATATEST on major INDEXTEST errors
- Breakdown and meaning of EXAMINE error message types
- Samples of output from EXAMINE runs

Introduction to EXAMINE

EXAMINE is an access method services command that allows the user to analyze and collect information on the structural consistency of key-sequenced data set clusters. This service aid consists of two tests: INDEXTEST and DATATEST.

INDEXTEST examines the index component of the key-sequenced data set cluster by cross-checking vertical and horizontal pointers contained within the index control intervals, and by performing analysis of the index information. It is the default test of EXAMINE.

DATATEST performs an evaluation of the 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.

For a complete description of the EXAMINE command format, see the "Functional Command Format" chapter in *Integrated Catalog Administration: Access Method Services Reference*.

Types of Data Sets

There are three types of data sets that can be tested by EXAMINE:

- Key-Sequenced Data Set
- VSAM Catalog
- Integrated Catalog Facility Catalog (BCS component)

Users of EXAMINE

EXAMINE end users fall into two categories:

1. ***Application Programmer/Data Set Owner.*** These users will want to know of any structural inconsistencies in their data sets, and they will be directed to corresponding IBM-supported recovery methods by appropriate summary messages. Their primary focus is to know the condition of data sets, and it

is suggested that they use the ERRORLIMIT(0) parameter of EXAMINE to suppress printing of detailed error messages.

2. ***System Programmer/Support Personnel.*** System programmers or support personnel will obtain information from detailed error messages necessary to determine what is wrong with a certain data set because they may want to document or fix a problem.

Users must have master level access to a catalog or control level access to a data set in order to examine it. Master level access to the master catalog is also sufficient to examine an integrated catalog facility user catalog.

How to Run EXAMINE

Sharing Considerations

There should not be any users open to the data set during the EXAMINE run.

EXAMINE will issue message "IDC01723I ERRORS MAY BE DUE TO CONCURRENT ACCESS" if errors are detected and determine that the data set may have been open for output during testing. Presence of this message does not necessarily indicate that the reported errors were due to concurrent access.

When EXAMINE is run against a catalog, concurrent access may have occurred without the message being issued. Because it may be difficult to stop system access to the catalog, jobs should not be run that would cause an update to the catalog.

For further considerations of data set sharing, see Chapter 9, "Sharing a VSAM Data Set" on page 113.

Deciding to Run an INDEXTEST, a DATATEST or Both Tests

INDEXTEST reads the entire index component of the KSDS cluster.

DATATEST reads the sequence set from the index component and the entire data component of the KSDS cluster. For this reason, it should take considerably more elapsed time and more system resources than INDEXTEST.

If you are using EXAMINE to document an error in the data component, run both tests. If you are using EXAMINE to document an error in the index component, it is usually not necessary to run the DATATEST.

If you are using EXAMINE to confirm a data set's integrity, the decision to run one or both tests depends on the time and resources available to you.

Controlling Message Printout

For details about using the ERRORLIMIT parameter of EXAMINE to suppress printing of supportive and individual data set structural error messages, see the section "Output from EXAMINE."

Skipping DATATEST on Major INDEXTEST Errors

If you decide to run both tests (INDEXTEST and DATATEST), INDEXTEST runs first. If INDEXTEST finds major structural errors, DATATEST doesn't run, even though you requested it. This gives you a chance to review the output from INDEXTEST and to decide whether or not you need to run DATATEST.

If you want to run DATATEST unconditionally, you must specify the NOINDEXTEST parameter on the EXAMINE statement to bypass the INDEXTEST.

Special Considerations When Examining a VSAM Catalog

When analyzing a VSAM catalog, a STEPCAT DD statement should be used to point to that catalog, and a VERIFY DATASET command should be applied before the EXAMINE command. The STEPCAT DD statement is not required for the master catalog.

Special Considerations When Examining an Integrated Catalog Facility User Catalog

The user must have master level access to either the user catalog being examined or to the master catalog. If the user has master level access to the master catalog, the self-describing records in the user catalog will not be read during open. If the user has only master level access to the user catalog being examined, a STEPCAT DD statement for the user catalog is required, and the catalog self-describing records will be read during open.

If the master catalog is protected by the Resource Access Control Facility or an equivalent product and the user does not have alter authority to it, a message may be issued indicating an authorization failure when the check indicated above is made. This is normal, and, if the user has master level access to the catalog being examined and the required STEPCAT DD statement, the examination will continue.

Output from EXAMINE

Message Hierarchy

Messages describing errors or inconsistencies are generated during EXAMINE processing as that condition is detected. The detection of an error condition may result in the generation of many messages. There are five distinct types of EXAMINE error messages:

1. **Status and Statistical Message**. This type of message tells you the status of the EXAMINE operation, such as the beginning and completion of each test. It provides general statistical data, such as the number of data records, the percentage of free space in data control intervals (CIs), and the number of deleted CIs. The four status messages are: IDC01700I, IDC01701I, IDC01709I, and IDC01724I. The five statistical messages are IDC01708I, IDC01710I, IDC01711I, IDC01712I, and IDC01722I.
2. **Supportive (Informational) Message**. Supportive messages (all remaining IDC0-type messages) issued by EXAMINE are used to clarify individual data set structural errors and to provide additional information pertinent to an error.

3. **Individual Data Set Structural Error Message**. The identification of an error is always reported by an individual data set structural error (IDC1-type) message which may be immediately followed by one or more supportive messages.

4. **Summary Error Message**. One or more summary error (IDC2-type) messages are generated at the completion of either INDEXTEST or DATATEST to categorize all individual data set structural error (IDC1-type) messages displayed during the examination. The summary error message represents the final analysis of the error(s) encountered, and the user should follow the course of recovery action as prescribed by the documentation.

5. **Function-Not-Performed Message**. Function-not-performed messages are all IDC3-type messages which indicate that the function you requested cannot be successfully performed by EXAMINE. In each case, the test operation terminates before the function completes.

   Function-not-performed messages are issued for a variety of reasons, among which are the following:

   - An invalid request such as an attempt to examine an entry-sequenced (ESDS) data set
   - A physical I/O error in a data set
   - A system condition such as insufficient storage
   - A system error (such as an OBTAIN DSCB failed)
   - An error encountered during INDEXTEST (see "Skipping DATATEST on Major INDEXTEST Errors" on page 141)

Controlling Message Printout

The ERRORLIMIT parameter in the EXAMINE command can be used to suppress supportive and individual data set structural error messages during an EXAMINE run. This parameter indicates the number of errors for which supportive and individual data set structural error messages are to be printed. When this value is reached, EXAMINE stops issuing error messages, but continues to scan the data set. ERRORLIMIT (0) means that none of these messages will be printed. When the ERRORLIMIT parameter is not specified by the user (the default condition), all supportive and individual data set structural error messages are printed. Note that the status and statistical messages, summary messages, and function-not-performed messages are not under the control of ERRORLIMIT, and will print regardless of the ERRORLIMIT settings. The ERRORLIMIT parameter is used separately by INDEXTEST and DATATEST. For more details about using this parameter, see *Integrated Catalog Administration: Access Method Services Reference*.

Samples of Output from EXAMINE Runs

INDEXTEST/DATATEST against an Error-Free Data Set

In this run, INDEXTEST and DATATEST are both executed successfully against an error-free data set. The first four messages tell us the status of the two EXAMINE tests performed with no errors detected. The next five messages then summarize component statistics as revealed by the DATATEST.

```
IDCAMS SYSTEM SERVICES

    EXAMINE NAME(EXAMINE.KD05) -
      INDEXTEST -
      DATATEST

IDC01700I INDEXTEST BEGINS
IDC01724I INDEXTEST COMPLETES NO ERRORS DETECTED
IDC01701I DATATEST BEGINS
IDC01709I DATATEST COMPLETES NO ERRORS DETECTED

IDC01708I 45 CONTROL INTERVALS ENCOUNTERED
IDC01710I DATA COMPONENT CONTAINS 1000 RECORDS
IDC01711I DATA COMPONENT CONTAINS 0 DELETED CONTROL INTERVALS
IDC01712I MAXIMUM LENGTH DATA RECORD CONTAINS 255 BYTES
IDC01722I 65 PERCENT FREE SPACE

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

INDEXTEST/DATATEST against a Data Set with a Structural Error

The user intended to run both tests, INDEXTEST and DATATEST, but INDEXTEST found an error in the sequence set. From the messages, we learn the following:

- A structural problem was found in the index component of the KSDS cluster.
- The current index level is 1 (which is the sequence set).
- The index control interval (beginning at the relative byte address of decimal 23552) where it found the error is displayed.
- The error is located at offset hexadecimal 10 into the control interval.

Because of this severe INDEXTEST error, DATATEST will not run in this particular case.

INDEXTEST/DATATEST against a Data Set with a Duplicate Key Error

The user intended to run both tests, INDEXTEST and DATATEST. INDEXTEST began and completed successfully. DATATEST looked at the data component of the KSDS cluster and found a "duplicate key" error.

EXAMINE then displayed the prior key (11), the data control interval at relative byte address decimal 512, and the offset address hexadecimal 9F into the control interval where the duplicate key was found.

```
IDCAMS SYSTEM SERVICES

    EXAMINE NAME(EXAMINE.KD99) INDEXTEST DATATEST
IDC01700I INDEXTEST BEGINS
IDC11701I STRUCTURAL PROBLEM FOUND IN INDEX
IDC01707I CURRENT INDEX LEVEL IS 1
IDC01720I INDEX CONTROL INTERVAL DISPLAY AT RBA 23552 FOLLOWS
000000  01F90301 00000000 00005E00 00000000   02000021 010701BC 2D2C2B2A 29282726   x.9..............................x
000020  25000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
000040  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
000060  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
000080  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
0000A0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
0000C0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
0000E0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
000100  00000000 0000F226 01240007 F7F12502   23F82601 22F72601 21F42601 20F32601   x......2.....71...8...7...4...3..x
000120  1FF6F025 021E001B F525011D F626011C   F526011B F226011A F5F12502 19F82601   x.60.....5...6...5...2...51...8..x
000140  18001CF4 F7250217 F4260116 F3260115   F4F02502 14260013 F6260112 001BF3F5   x...47...4...3...40......6.....35x
000160  250211F2 260110F3 F125020F F826010E   F726010D F426010C 001CF2F3 25020BF2   x...2...31...8...7...4.....23...2x
000180  F025020A 260009F6 260108F5 260107F2   26010600 40F0F0F0 F0F0F0F0 F0F0F0F0   x0......6...5...2.... 00000000000x
0001A0  F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0   F0F0F0F0 F0F0F0F0 F0F0F1F1 002705F8   x0000000000000000000000000011...8x
0001C0  260104F7 260103F4 260102F3 260101F0   F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0   x...7...4...3...00000000000000000x
0001E0  F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0   F0F0F0F0 F0F00027 000001F9 01F90000   x0000000000000000000000.....9.9..x
IDC01714I ERROR LOCATED AT OFFSET 00000010
IDC21701I MAJOR ERRORS FOUND BY INDEXTEST
IDC31705I DATATEST NOT PERFORMED DUE TO SEVERE INDEXTEST ERRORS
IDC3003I FUNCTION TERMINATED. CONDITION CODE IS 12

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 12
```

```
IDCAMS SYSTEM SERVICES

    EXAMINE NAME(EXAMINE.KD99) INDEXTEST DATATEST
IDC01700I INDEXTEST BEGINS
IDC01724I INDEXTEST COMPLETE NO ERRORS DETECTED
IDC01701I DATATEST BEGINS
IDC11741I DUPLICATE CONSECUTIVE KEYS FOUND
IDC01717I DATA KEY FOLLOWS
000000  F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0   F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0   x00000000000000000000000000000000x
000020  F0F0F0F0 F0F0F1F1                                                          x00000011                        x

DIC01713I DATA CONTROL INTERVAL DISPLAY AT RBA 512 FOLLOWS
000000  00000000 0000D0C1 AB33F0F0 F0F0F0F0   F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0   x.......A..0000000000000000000000x
000020  F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0   F1F1C3C3 C3C3C3C3 C3C3C3C3 C3C3C3C3   x000000000000000011CCCCCCCCCCCCCCx
000040  C3C3C3C3 C3C3C3C3 C3C3C3C3 C3C3C3C3   C3C3C3C3 C3C3C3C3 C3C3C3C3 C3C3C3C3   xCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCx
000060  C3C3C3C3 C3C3C3C3 C3C3C3C3 C3C3C3C3   C3C3C3C3 C3C3C3C3 C3C3C3C3 C3C3C3C3   xCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCx
000080  C3C3C3C3 C3C3C3C3 C3C3C3C3 C3C3C3C3   C3C3C3C3 C3000000 0000006E E065DEF0   xCCCCCCCCCCCCCCCCCCCCC......>...0x
0000A0  F0F0F0F0 F0F0F0F0 F0F0F0F0 0F0F0F0F   F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0   x00000000000000000000000000000000x
0000C0  F0F0F0F0 F0F1F1C4 C4C4C4C4 C4C4C4C4   C4C4C4C4 C4C4C4C4 C4C4C4C4 C4C4C4C4   x0000011DDDDDDDDDDDDDDDDDDDDDDDDDx
0000E0  C4C4C4C4 C4C4C4C4 C4C4C4C4 C4C4C4C4   C4C4C4C4 00000000 000055E2 0706F0F0   xDDDDDDDDDDDDDDDDDDDD.......S..00x
000100  F0F0F0F0 F0F0F0F0 F0F0F0F0 0F0F0F0F   F0F0F0F0 F0F0F0F0 F0F0F0F0 F0F0F0F0   x00000000000000000000000000000000x
000120  F0F0F0F0 F3F1C1C1 C1C1C100 00000000   00000000 00000000 00000000 00000000   x000031AAAAA.....................x
000140  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
000160  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
000180  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
0001A0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
0001C0  00000000 00000000 00000000 00000000   00000000 00000000 00000000 00000000   x................................x
0001E0  00000000 00000000 00000000 00000000   00000000 00370000 5F000095 012B00C8   x........................-......Hx
IDC01714I ERROR LOCATED AT OFFSET 0000009F
IDC21703I MAJOR ERRORS FOUND BY DATATEST
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 8
```

Appendix A. Using 31-Bit Support

VSAM allows you to obtain control blocks, buffers and multiple local shared resource (LSR) pools above or below 16 megabytes. However, if your program uses a 24-bit address, it may program check if you attempt to reference control blocks, buffers or LSR pools located above 16 megabytes. With a 24-bit address, you do not have addressability to the data buffers.

If you attempt to use LOCATE mode to access records while in 24-bit mode, your program will program check (ABEND0C4).

When using 31-bit support, you must observe the following:

- All VSAM control blocks that currently have fields defined as 31-bit addresses *must* contain 31-bit addresses. You may not use the high-order byte of a 31-bit address field as a user-defined flag field. This is true whether you are using 24-bit or 31-bit support.
- I/O buffers and control blocks may be obtained either above or below 16 megabytes in storage.
  - I/O buffers and control blocks may be requested below 16 megabytes by taking the ACB and BLDVRP macro defaults.
  - I/O buffers may be requested above 16 megabytes and control blocks below 16 megabytes by specifying the RMODE31=BUFF parameter on the ACB or BLDVRP macro.
  - Control blocks may be requested above 16 megabytes and buffers below 16 megabytes by specifying the RMODE31=CB parameter on the ACB or BLDVRP macro.
  - Control blocks and buffers may be requested above 16 megabytes by specifying the RMODE31=ALL parameter on the ACB or BLDVRP macros.

  Note: Prior to Data Facility Product Version 2, Release 3.0, the ACB's AMODE31 subparameter specified that buffers were to be obtained above 16 megabytes. Control blocks were always obtained below 16 megabytes. With Data Facility Product Version 2, Release 3.0, the RMODE31 parameter specifies where buffers and control blocks are to be obtained. The AMODE31 subparameter still indicates buffers are requested above 16 megabytes, but it is mutually exclusive with the RMODE31 parameter. If both are specified, the AMODE31 subparameter is ignored. When you add the RMODE31 parameter to your programs, you should remove the AMODE31 subparameter.
- The parameter list passed to your UPAD and JRNAD exit routine resides in the same area specified with the VSAM control blocks. If RMODE31=CB or RMODE31=ALL is specified, the parameter list resides above 16 megabytes.
- You must recompile the portion of your program that contains the ACB, BLDVRP, and DLVRP macro specifications.
- You cannot use JCL parameters to specify 31-bit parameters.

- The following table summarizes the 31-bit support keyword parameters and their use in the applicable VSAM macros.

MACRO	RMODE31=	MODE=	LOC=
ACB	Virtual storage location of VSAM control blocks and I/O buffers	INVALID	INVALID
BLDVRP	Virtual storage location of VSAM LSR pool, VSAM control blocks and I/O buffers	Format of the BLDVRP parameter list (24-bit or 31-bit format)	INVALID
CLOSE	INVALID	Format of the CLOSE parameter list (24-bit or 31-bit format)	INVALID
DLVRP	INVALID	Format of the DLVRP parameter list (24-bit or 31-bit format)	INVALID
GENCB	RMODE31 values to be placed in the ACB that is being created. When the generated ACB is OPENed, the RMODE31 values will then determine the virtual storage location of VSAM control blocks and I/O buffers.	INVALID	Location for the virtual storage obtained by VSAM for the ACB, RPL, or EXIT LIST.
MODCB	RMODE31 values to be placed in a specified ACB	INVALID	INVALID
OPEN	INVALID	Format of the OPEN parameter list (24-bit or 31-bit format)	INVALID

"Obtaining Buffers above 16 Megabytes" on page 76 explains how to create and access buffers that reside above 16 megabytes. Chapter 10, "Sharing Resources among Data Sets" on page 127, explains how to build multiple LSR pools in an address space.

Appendix B. Job Control Language

This appendix describes job control language, an optional method for connecting a data set and the program that is to use it. There is also a description of how to code the VSAM JCL parameter (AMP).

The data set name is a necessary link between a processing program and the data set to be processed. When JCL is used, the access method control block gives the name of the DD statement so that the OPEN macro can make the connection between the program and the data set named in the DD statement, connecting the program and data. JCL is also used to catalog, uncatalog, and delete non-VSAM data sets in a catalog.

If JCL is not used, an attempt is made to dynamically allocate the data set or volume as required. When you define a VSAM data set or catalog, no DD statement is required if access method services can dynamically allocate the volume. (For an explanation of dynamic allocation, see the *JCL* manual.)

The catalog contains most of the information required by VSAM to process a data set, so VSAM requires minimal information from JCL. Data set name and disposition describe the data set. A key-sequenced data set is described with a single DD statement.

How to Code JCL

VSAM data sets are created using access method services and are cataloged in a catalog. To identify a VSAM data set through JCL, specify a DD statement of the form:

```
//ddname DD DSNAME=dsname,DISP={OLD|SHR}
```

Optionally, the **AMP** parameter may be specified to modify the program's operation.

If a data set has been defined in a user catalog, it is also necessary to identify the user catalog by means of either a JOBCAT or a STEPCAT DD statement. For more information on coding DD statements for a catalog, see *Catalog Administration Guide.*

The DSNAME parameter specifies the name of the data set you are processing. Each VSAM data set is defined as a cluster of one or two components. A key-sequenced data set is made up of a data component and an index component; and an entry-sequenced and a relative record data set are made up of only a data component. If you need to process a component separately, you may specify the component's name in the DSNAME parameter.

To allow only one job step to access the data set, specify DISP=OLD. If the data set's share options allow the type of sharing your program anticipates, you can specify DISP=SHR in the DD statements of separate jobs to enable two or more job steps to share a data set. With separate DD statements, several subtasks can share a data set under the same rules as for cross-region sharing. When separate DD statements are used and one or more subtasks will perform output processing, the DD statements must specify DISP=SHR. For more details on sharing data sets, see Chapter 9, "Sharing a VSAM Data Set" on page 113.

Mounting a Subset of Volumes

With a data set defined in a VSAM catalog, you may mount a subset of the volumes on which a data set is stored (called *subset mount*). To do this, specify the DD parameters VOLUME and UNIT. When mounting a subset, consider the following:

- This may cause excessive processing time because of mount and demount activities directed to those volumes.
- Specifying those parameters to open a DCB (to be processed through the ISAM interface program) prevents a reference to the VSAM catalog and requires that you use the AMP subparameter AMP='AMORG' to identify the data set as a VSAM data set.
- If you specify VOLUME and UNIT to open a VSAM ACB, AMORG is not required.

JCL Parameters Used with VSAM

Because the operating system allows DD parameters and subparameters that do not apply to a VSAM data set, you should only use the DD parameters and subparameters that have a clear meaning when used with VSAM. Figure 30 on page 151 shows the DD parameters and subparameters that can be used with VSAM and indicates their meaning for a VSAM data set. DD parameters and subparameters not shown in Figure 30 on page 151 should be avoided.

Figure 30 (Page 1 of 2). JCL DD Parameters

Parameter	Subparameter	Comment
AMP		See "Coding the AMP Parameter" on page 152.
DDNAME	ddname	Specifies name of DD statement.
DISP	**SHR**	Indicates that you are willing to share the data set with other jobs. This subparameter alone, however, does not guarantee that sharing will take place. See Chapter 9, "Sharing a VSAM Data Set" on page 113, for a description of data-set sharing.
	OLD	Defaults to SHR, if specified for an integrated catalog facility or VSAM catalog.
	PASS	For VSAM, KEEP is assumed for PASS.
DSNAME	dsname	Specifies VSAM data set or non-VSAM object.
DUMMY		An attempt to read results in an end-of-data condition, and an attempt to write results in a return code that indicates the write was successful. If specified, AMP='AMORG' must also be specified (see "Coding the AMP Parameter" on page 152). No I/O activity is performed for a dummy data set.
UNIT		**Note:** Unit information should not be specified for data sets cataloged in an integrated catalog facility catalog.
	address	Must be the address of a valid device for VSAM. If not, OPEN will fail.
	type	Must be a type supported by VSAM. If not, OPEN will fail.
	group	Must be a group supported by VSAM. If not, OPEN will fail.
	p	There must be enough units to mount all of the volumes specified. If sufficient units are available, parallel mounting (p) can improve performance by avoiding the mounting and demounting of volumes.
	unitcount	If the number of devices requested is greater than the number of volumes on which the data set resides, the extra devices are allocated anyway. If data and index components reside on unlike devices, the extra devices are allocated evenly between the unlike device types. If the number of devices requested is less than the number of volumes on which the data set resides but greater than the minimum number required to gain access to the data set, the devices over the minimum are allocated evenly between unlike device types. If devices beyond the count specified are in use by another task but are sharable and have mounted on them volumes containing parts of the data set to be processed, they will also be allocated to this data set.
	DEFER	The volume to be used does not have to be mounted until the data set is opened. This causes all the units associated with demountable volumes to be flagged as nonshared.
VOLUME	**PRIVATE**	Specifies volume is demounted unless RETAIN is coded.
	RETAIN	Specifies that the volume is to be retained at the system if it was demounted during the job.

Figure 30 (Page 2 of 2). JCL DD Parameters

Parameter	Subparameter	Comment
	SER	**Note:** Volume serial number should not be specified for data sets cataloged in an integrated catalog facility catalog. The volume serial number(s) used in the access method services DEFINE command for the data set must match the volume serial numbers in the VOLUME=SER specification in the job in which the data set is defined. After a VSAM data set is defined, the volume serial number(s) need not be specified on a DD statement to retrieve or process the data set. For data sets defined in VSAM catalogs if VOLUME=SER and UNIT=type are specified, only those volumes named are initially mounted. Other volumes may be mounted when they are needed, if at least one of the units allocated to the data set is not sharable and the number of OPENs issued against the volume is less than or equal to 1, or the unit count is greater than the total number of volumes initially mounted. One unit is flagged as nonsharable when unit count is less than the number of volume serial numbers specified. If VOLUME=SER is specified and the data set is cataloged in a user catalog, include a JOBCAT or STEPCAT DD statement to identify the catalog to the current job step unless the high-level qualifier of the data set name is also the name of the user catalog.

Coding the AMP Parameter

VSAM has one JCL parameter of its own: AMP. The AMP parameter takes effect when the data set defined by the DD statement is opened. It has subparameters for:

- Overriding operands specified by way of the ACB, EXLST, and GENCB macros
- Supplying operands missing from the ACB or GENCB macro
- Indicating checkpoint/restart options
- Indicating options when using ISAM macros to process a key-sequenced data set
- Indicating that the data set is a VSAM data set when you specify unit and volume information or DUMMY in the DD statement
- Indicating that you want VSAM to supply storage dumps of the access method control block(s) that identify this DD statement

The AMP parameter takes effect when the data set defined by the DD statement is opened.

The format of the AMP parameter is:

//...	DD	...AMP=(['AMORG'] [,'BUFND=*number*'] [,'BUFNI=*number*'] [,'BUFSP=*number*'] [,'CROPS={RCK|NCK|NRE|NRC}'] [,'OPTCD={I|L|IL}'] [,'RECFM={F|FB|V|VB}'] [,'STRNO=*number*'] [,'SYNAD=*modulename*'] [,'TRACE'])

where:

AMORG
specifies that the DD statement defines a VSAM data set. When you specify unit and volume information for a DCB (through the ISAM interface program) or DUMMY in the DD statement, you must specify AMORG. Under these conditions, the system doesn't have to search a catalog to find out what volume(s) are required, and therefore doesn't know that the DD statement defines a VSAM data set. You never have to specify unit and volume information, unless you want to mount a subset of the volumes on which the data set is stored, or want to defer mounting.

BUFND=*number*
specifies the number of data buffers.

BUFNI=*number*
specifies the number of index buffers.

BUFSP=*number*
specifies that one or more of these values is to override whatever was specified in the ACB or GENCB macro, or that one or more of these values is to be provided if not previously specified. For further information on BUFND, BUFNI, and BUFSP, see *VSAM Administration: Macro Instruction Reference.*

CROPS=[RCK|NCK|NRE|NRC]
specifies one of four checkpoint/restart options, described in detail in *Checkpoint/Restart User's Guide.* If you specify an option that is not applicable for a data set, such as the data-erase test for an input data set, the option is ignored.

RCK
specifies that a data-erase test and data set-post-checkpoint modification tests are to be performed.

NCK
specifies that data set-post-checkpoint modification tests are not to be performed.

NRE
specifies that a data-erase test is not to be performed.

NRC
specifies that neither a data-erase test nor data set-post-checkpoint modification tests are to be performed.

OPTCD={I|L|IL}
specifies the type of processing of records flagged for deletion (binary 1's in the first byte) with an ISAM processing program using the ISAM interface. I and L are described in Appendix F, "Using ISAM Programs with VSAM Data Sets" on page 205.

RECFM={F|FB|V|VB}
specifies record format in the same way as the DCB (data control block) parameter that is used for processing an indexed-sequential data set. You use it when processing a VSAM data set with an ISAM processing program to indicate what record format the processing program assumes. The options are described in Appendix F, "Using ISAM Programs with VSAM Data Sets" on page 205.

STRNO=*number*
specifies a value that is to override the STRNO value specified in the ACB or GENCB macro, or to provide a value if one was not specified.

SYNAD=*modulename*
specifies a value that is to override the address of a SYNAD exit routine specified in the EXLST or GENCB macro that generates the exit list. The exit list intended is the one whose address is specified in the access method control block that links this DD statement to the processing program. If no SYNAD exit was specified, the SYNAD parameter of AMP is ineffective. You can also use this parameter, when you are processing a VSAM data set with an ISAM processing program, to provide an ISAM SYNAD routine or to replace one with another.

TRACE
specifies that generalized trace facility (GTF) is to be active, along with your processing job, to gather information associated with opening, closing, and end-of-volume processing for the data set defined on this DD statement. You can print the trace output with the AMDPRDMP service program.

Note: See Appendix F, "Using ISAM Programs with VSAM Data Sets" on page 205 for additional information on the use of the AMP parameter with an ISAM processing program.

If you have more than one subparameter, you must enclose them in apostrophes. Apostrophes can enclose each individual subparameter or group of subparameters. If you have more than one pair of apostrophes, you must enclose all the subparameters in a pair of parentheses. For example, AMP='AMORG,TRACE' or AMP=('AMORG','TRACE'). If the subparameters continue from one line to another, a pair of apostrophes cannot extend from one line to the next, and you must use a pair of parentheses to enclose all the subparameters.

The AMP parameter cannot be defined as a symbolic parameter (a symbol preceded by an ampersand (&) that stands for a parameter or the value assigned to a parameter or subparameter in a cataloged or in-stream procedure).

JCL Parameters Not Used with VSAM

VSAM ignores parameters for defining tape data sets; data-set sequence numbers, NSL, NL, BLP, and AL. You cannot use the parameters for a sequential data set (DATA, SYSOUT, and *) for specifying a VSAM data set. DD names that are invalid for VSAM data sets are: JOBLIB, STEPLIB, SYSABEND, SYSUDUMP, and SYSCHK.

DD parameters that are invalid are: UCS, QNAME, DYNAM, TERM, and the forms of DSNAME for ISAM, PAM (partitioned access method), and generation data groups. VSAM does not allow for temporary data sets or concatenated data sets.

Appendix C. Examples of Defining and Manipulating Data Sets

This set of examples illustrates a wide range of functions available through access method services that allow you to do the following:

- Define data sets
- Alter a data set's attributes
- Copy and print data sets
- Define alternate indexes and paths
- Export and import VSAM data sets for backup and recovery
- Delete a linear data set

The integrated catalog facility system catalog that exists is assumed to be security protected at the update-password, control-password, and master-password levels.

Example 1: Define VSAM Data Sets

This example defines five VSAM data sets: two key-sequenced data sets into a previously defined user catalog, USERCAT1; a relative record data set into the system catalog; a reusable entry-sequenced data set into a previously defined user catalog, USERCAT2; and a linear data set into the previously defined user catalog, USERCAT2.

```
//DEFVSAM  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//VVSER03  DD    DISP=OLD,UNIT=3380,VOL=SER=VSER03
//SYSPRINT DD    SYSOUT=A
//SYSABEND DD    SYSOUT=A
//AMSDUMP  DD    SYSOUT=A
//SYSIN    DD    *

     DEFINE CLUSTER -
          (NAME(ALIAS01.MYDATA) -
          VOLUMES(VSER02) -
          RECORDS(1000 500)) -
        DATA -
          (NAME(ALIAS01.KSDATA) -
          KEYS(15 0) -
          RECORDSIZE(250 250) -
          FREESPACE(20 10) -
          BUFFERSPACE(25000)) -
        INDEX -
          (NAME(ALIAS01.KSINDEX) -
          IMBED) -
        CATALOG(USERCAT1/UCATUPPW)

     DEFINE CLUSTER -
          (NAME(ALIAS01.EXAMPLE.KSDS1) -
          READPW(KSD1PSWD) -
          MODEL(ALIAS01.MYDATA)) -
        DATA -
          (NAME(ALIAS01.EXAMPLE.KSDS1.DATA) -
          UNIQUE -
          CYLINDERS(2 1)) -
        INDEX -
          (NAME(ALIAS01.EXAMPLE.KSDS1.INDEX) -
          UNIQUE -
          CYLINDERS(1 1)) -
        CATALOG(USERCAT1/UCATUPPW)

     IF LASTCC = 0 -
        THEN -
        LISTCAT ENTRIES -
                (ALIAS01.EXAMPLE.KSDS1/KSD1PSWD) -
                ALL

     DEFINE CLUSTER -
          (NAME(MAST01.RRDS1) -
          VOLUMES(VSER01) -
          TRACKS(10 5) -
          RECORDSIZE(100 100) -
          NUMBERED) -
        CATALOG(ICFMAST1/MASTMPW1)

     IF LASTCC = 0 -
        THEN -
        LISTCAT ENTRIES -
                (MAST01.RRDS1) -
                CLUSTER -
                ALL

     DEFINE CLUSTER -
          (NAME(ALIAS02.ESDS1.CLUST01) -
          VOLUMES(VSER03) -
          FILE(DD3) -
          NONINDEXED -
          TRACKS (3 1) -
          CISIZE(4096)) -
        CATALOG(USERCAT2/USERMPW)

     IF LASTCC = 0 -
        THEN -
        LISTCAT ENTRIES -
                (ALIAS02.ESDS1.CLUST01) -
                ALLOCATION

     DEFINE CLUSTER -
          (NAME(ALIAS02.LINEAR.CLUST01) -
          VOLUMES(VSER03) -
          LINEAR -
          TRACKS(3 1)) -
        CATALOG(USERCAT2/USERMPW)

     IF LASTCC = 0 -
        THEN -
        LISTCAT ENTRIES -
                (ALIAS02.LINEAR.CLUST01) -
                ALL
/*
```

Explanation of Commands

The first DEFINE command defines a key-sequenced data set on volume VSER02. The high-level name of the data set is the alias name of the catalog into which it is defined.

1. The CLUSTER parameter is required, and NAME specifies the cluster being defined.
2. The VOLUMES parameter is required and specifies the volume containing the data set.
3. The RECORDS parameter specifies the space to be allocated to the cluster. This is a required parameter.
4. The DATA parameter is required when attributes are to be explicitly specified for the data component of the cluster. The NAME parameter specifies the name of the data component.
5. The KEYS parameter specifies the key length and offset.
6. The RECORDSIZE parameter specifies the average and maximum record sizes.
7. The BUFFERSPACE parameter is specified for improved performance.
8. The INDEX parameter is required when attributes are to be explicitly specified for the index component of the cluster. The NAME parameter specifies the name of the index component.
9. The IMBED parameter specifies that the index sequence set is to be placed with the data component.
10. Because the catalog is password protected, the CATALOG parameter is required.

The second DEFINE command defines a unique key-sequenced data set. The high-level name of the data set is the alias name of the catalog into which it is being defined. This example shows how data set attributes can be specified by modeling and direct specification.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The READPW parameter specifies the read password of this cluster. Because no master password is defined, this password will be propagated up to the master level. Thus, this cluster is security protected even though its model was not protected.
3. The MODEL parameter specifies the name of the data set to be used as the model.
4. The DATA parameter is required if attributes are to be specified for the data component. The NAME parameter specifies the name of the data component. If a name is not specified, a name is generated.
5. The UNIQUE parameter specifies that this portion of the data set is the only one that occupies the data space allocated to it.
6. The CYLINDERS parameter specifies the amount of space to be allocated to the cluster's data component.

7. The INDEX parameter is required if attributes are to be specified for the index component. The NAME parameter specifies the name of the index component. If a name is not specified, a name is generated.
8. The UNIQUE parameter specifies that this portion of the data set is the only one that occupies the data space allocated to it.
9. The CYLINDERS parameter specifies the amount of space to be allocated to the cluster's index component.
10. Because the user catalog is password protected, the CATALOG parameter is required. It specifies the name of the user catalog and its update password which is required to define into a protected catalog.

If the define of the unique key-sequenced data set was successful, then the following LISTCAT command is executed. The high-level name of the data set will direct the LISTCAT to the appropriate user catalog. The ENTRIES and ALL parameters cause the entire catalog description of the data set just defined to be listed.

The third DEFINE command defines a suballocated relative record data set on volume VSER01.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The VOLUMES parameter is required and specifies the volume containing this data set. Because the master catalog is recoverable, access method services dynamically allocate this volume to access the catalog recovery area for the data set. This requires that the volume be mounted permanently RESIDENT or RESERVED.
3. The TRACKS parameter specifies the amount of space allocated to this data set. A space parameter is required.
4. The RECORDSIZE parameter specifies the average and maximum record sizes which, in the case of a relative record data set, must be equal.
5. The NUMBERED parameter is required to specify that this is a relative record data set.
6. The CATALOG parameter is required because the master catalog is password protected. It specifies the name of the master catalog and its master password which is required (or its update password) to define into a protected catalog.

If the define of the suballocated relative record data set was successful, then the following LISTCAT command is executed. The ENTRIES, CLUSTER, and ALL parameters cause the entry just defined to be listed—limited, however, to the cluster entry (that is, the data component's entry is not listed).

The fourth DEFINE command defines a suballocated entry-sequenced data set on volume VSER03. The high-level name of the data set is the alias name of the catalog into which it is defined.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The VOLUMES parameter is required and specifies the volume containing this data set.

3. The FILE parameter names the DD statement that identifies the direct access device and volume on which space is to be allocated to the cluster.
4. The NONINDEXED parameter is required to override the default (INDEXED).
5. The TRACKS parameter specifies the amount of space to be allocated to this data set. A space parameter is required.
6. The CISIZE parameter specifies the control interval size.
7. The CATALOG parameter specifies the name of the catalog into which the cluster is to be defined. The catalog's update- or higher-level password is required.

If the define of the suballocated entry-sequenced data set was successful, then the following LISTCAT command is executed. The high-level name of the data set will direct the LISTCAT to the appropriate catalog. The ENTRIES and ALLOCATION parameters cause the data set entry just defined to be listed—limited, however, to only volume and allocation information.

The fifth DEFINE command builds a cluster entry and a data entry to define the linear data set cluster ALIAS02.LINEAR.CLUST01. The high-level name of the data set is the alias name of the catalog into which it is being defined. The command's parameters are:

1. NAME specifies the cluster's name.
2. VOLUMES specifies that the cluster is to reside on volume VSER03.
3. TRACKS specifies that 3 tracks are allocated for the cluster's space. When the cluster is extended, it is to be extended in increments of 1 track.
4. LINEAR specifies that the cluster's data organization is to be linear.
5. CATALOG specifies the catalog into which the cluster is to be defined. The example also supplies the user catalog's master password.

If the define of the linear data set was accomplished, the LISTCAT command lists the catalog entry for a linear data set.

1. The LISTCAT command invokes the AMS list catalog function.
2. The ENTRY parameter defines which object's catalog entry to list.
3. The ALL parameter defines the scope of the listing.

Example 2: Define Non-VSAM and VSAM Data Sets

This example defines a non-VSAM data set into an integrated catalog facility user catalog, a VSAM key-sequenced data set into the integrated catalog facility system catalog, and a VSAM entry-sequenced data set into an integrated catalog facility user catalog.

```
//DEFVSM2  JOB  ...
//JOBCAT   DD   DSN=USERCAT2,DISP=SHR
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//SYSIN    DD   *

     DEFINE NONVSAM -
          (NAME(EXAMPL1.NONVSAM1) -
          VOLUMES(VSER02) -
          DEVICETYPES(3380)) -
        CATALOG(USERCAT1/UCATUPPW)

     IF LASTCC = 0 -
        THEN -
        LISTCAT NONVSAM -
                ALL -
                CATALOG(USERCAT1)

     DEFINE CLUSTER -
          (NAME(MAST01.KSDS2)) -
        DATA -
          (NAME(MAST01.KSDS2.DATA) -
          MASTERPW(DAT2MRPW) -
          UPDATEPW(DAT2UPPW) -
          READPW(DAT2RDPW) -
          RECORDS(500 100) -
          EXCEPTIONEXIT(DATEXIT) -
          ERASE -
          FREESPACE(20 10) -
          KEYS(6 4) -
          RECORDSIZE(80 100) -
          VOLUMES(VSER01)) -
        INDEX -
          (NAME(MAST01.KSDS2.INDEX) -
          MASTERPW(IND2MRPW) -
          UPDATEPW(IND2UPPW) -
          READPW(IND2RDPW) -
          RECORDS(300 300) -
          IMBED -
          VOLUMES(VSER01)) -
        CATALOG(ICFMAST1/MASTMPW1)

     IF LASTCC = 0 -
        THEN -
        LISTCAT DATA -
                ALL -
                ENTRIES -
                (MAST01.KSDS2/DAT2MRPW) -
                CATALOG(ICFMAST1)
     DEFINE CLUSTER -
          (NAME(EXAMPL2.ESDS2) -
          VOLUMES(VSER03) -
          SPANNED -
          CYLINDERS(2 1) -
          NONINDEXED -
          REUSE -
          MASTERPW(ESD2MRPW) -
          CONTROLPW(ESD2CTPW) -
          UPDATEPW(ESD2UPPW) -
          READPW(ESD2RDPW)) -
        CATALOG(USERCAT2/UCATMRPW)

     IF LASTCC = 0 -
        THEN -
        DO -
        LISTCAT ENTRIES -
                (EXAMPL2.ESDS2/ESD2MRPW) -
                ALL
        LISTCAT NAME -
                CATALOG(ICFMAST1/MASTMPW1)
        END
/*
```

Explanation of Job Control Language Statements

The JOBCAT DD statement describes the user catalog USERCAT2 as a job catalog. All references will be to this job catalog unless otherwise directed.

Explanation of Commands

The first DEFINE command defines an existing non-VSAM data set into user catalog USERCAT1.

1. The NONVSAM parameter is required and NAME specifies the non-VSAM object being defined.
2. The VOLUMES parameter is required and specifies the volume containing the data set.
3. The DEVICETYPES parameter is required and specifies the device type of the volume.
4. The CATALOG parameter specifying the name of the integrated catalog facility user catalog is required, because:
   - A job catalog also appears in the job control language so this parameter explicitly directs the define to the user catalog.
   - The catalog is protected and its update password is required for the define.

If the definition of the non-VSAM entry was successful, the following LISTCAT command is executed.

1. The NONVSAM and ALL parameters cause all the non-VSAM entries cataloged in USERCAT1 to be listed.
2. The CATALOG parameter directs the LISTCAT to a specific user catalog.

The second DEFINE command defines a key-sequenced data set. Note that attributes are specified at the data and index level rather than the cluster level.

1. The CLUSTER parameter is required, and NAME specifies the cluster being defined.
2. The DATA component is explicitly named via the NAME parameter.
3. The MASTERPW, UPDATEPW, and READPW parameters specify the master, update, and read passwords, respectively, of this data component.
4. The RECORDS parameter specifies the amount of space to be allocated to the data component. A space allocation parameter is required.
5. The EXCEPTIONEXIT parameter specifies the name of the routine to be given control if an exception occurs while processing the data component.
6. The ERASE parameter specifies that the data component is to be overwritten with binary zeros when it is deleted.
7. The FREESPACE parameter specifies the percentage of space within control intervals and control areas, respectively, that is to remain free.
8. The KEYS parameter specifies the key length and offset.
9. The RECORDSIZE parameter specifies the average and maximum record sizes.
10. The VOLUMES parameter is required and specifies the volume containing this data component.
11. The INDEX component is explicitly named via the NAME parameter.
12. The MASTERPW, UPDATEPW, and READPW parameters specify the master, update, and read passwords, respectively, of this index component.
13. The RECORDS parameter specifies the amount of space to be allocated to the index component. A space allocation parameter is required.
14. The IMBED parameter specifies that the index sequence set is to be placed with the data component.
15. The VOLUMES parameter is required and specifies the volume containing this index component.
16. Because the master catalog is password protected, the CATALOG parameter is required. It specifies the name of the master catalog and its update password, which is required to define into a protected catalog. This parameter is also required, because a DD statement for the job catalog (JOBCAT) appears in the job control language and this define must, therefore, be explicitly directed to the master catalog.

If the definition of the key-sequenced data set was successful, the following LISTCAT command is executed.

1. The DATA, ALL, and ENTRIES parameters cause all the information contained in the data component entry to be listed.
2. The CATALOG parameter directs the LISTCAT to the master catalog.

The third DEFINE command defines an entry-sequenced data set.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The VOLUMES parameter specifies the volume (VSER03) that is to contain the data set being defined.
3. The SPANNED parameter specifies that records may span control interval boundaries.
4. The CYLINDERS parameter specifies the amount of space to be allocated to this data set. A space parameter is required.
5. The NONINDEXED parameter is required to override the default (INDEXED).
6. The REUSE parameter specifies that the data set can be reused, that is, reloaded without being deleted and redefined.
7. The MASTERPW, CONTROLPW, UPDATEPW, and READPW parameters specify passwords different from the passwords specified for the data set being modeled.
8. The CATALOG parameter is required, because the user catalog is password protected.

If the define of the entry-sequenced data set was successful, the following LISTCAT commands are executed.

1. The ENTRIES and ALL parameters of the first LISTCAT command cause all the cataloged information in the entry just defined to be listed.
2. The NAME parameter of the second LISTCAT command causes only the names of the objects cataloged in the master catalog to be listed.
3. The CATALOG parameter specifies the master password of the master catalog that allows access to all objects in the catalog and directs the LISTCAT to the master catalog.

Example 3: Alter the Cataloged Attributes of VSAM Data Sets

This example shows how the cataloged attributes of three VSAM data sets are modified. Each ALTER command is followed by a LISTCAT command, which will execute only if its previous ALTER command completed successfully. The LISTCAT command prints the updated catalog entry.

This example depends on the successful completion of examples 1 and 2, which define the VSAM data sets whose attributes are being altered.

```
//ALTER    JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//SYSIN    DD   *

     ALTER -
          ALIAS01.EXAMPLE.KSDS1.DATA -
          FREESPACE(10 10)

     IF LASTCC = 0 -
          THEN -
          LISTCAT -
              ENTRIES(ALIAS01.EXAMPLE.KSDS1.DATA/KSD1PSWD) -
              ALL

     ALTER -
          EXAMPL2.ESDS2/ESD2MRPW -
          MASTERPW(ESD2PWMR) -
          CONTROLPW(ESD2PWCT) -
          UPDATEPW(ESD2PWUP) -
          READPW(ESD2PWRD)

     IF LASTCC = 0 -
          THEN -
          LISTCAT -
              ENTRIES(EXAMPL2.ESDS2/ESD2PWMR) -
              CLUSTER -
              ALL

     ALTER -
          ALIAS02.ESDS1.CLUST01 -
          TYPE(LINEAR) -
          CATALOG(USERCAT2/USERMPW)

     IF LASTCC = 0 -
          THEN -
          LISTCAT -
              ENTRIES(ALIAS02.ESDS1.CLUST01/USERMPW) -
              ALL
/*
```

Explanation of Commands

The first ALTER command shows how the space management attributes of a data set are "tuned" for optimum performance.

The data component ALIAS01.EXAMPLE.KSDS1.DATA of the key-sequenced VSAM data set ALIAS01.EXAMPLE.KSDS1 was defined with 40% free space in both control intervals and control areas. Now that data records have been loaded into the data set, its free space attributes no longer appear to require 40% free space. It is now desirable to have 10% free space in both control intervals and control areas. The catalog containing the component to be altered is located through its alias 'ALIAS01', which is the high-level qualifier of the component name.

1. ALIAS01.EXAMPLE.KSDS1.DATA names the entry whose attributes are to be altered with this command. No password is required, because the data object is not password protected.
2. The FREESPACE parameter respecifies percentages of free space that apply to the data component.

The ALTER command is followed by a modal command that examines the condition code set when the ALTER command completes. If the ALTER command completes successfully, the LISTCAT command that immediately follows it prints the changed catalog entry. The ENTRIES and ALL parameters explicitly name the entry to be listed and specify that the entire entry is to be listed. Because all information about the data component is to be listed, some information in the associated cluster must be accessed; this requires the cluster's read password.

The second ALTER command shows how a data set's passwords can be modified. You could use the same technique to provide passwords for an existing VSAM object that does not have passwords.

1. EXAMPL2.ESDS2 names the entry whose attributes are to be altered with this command. The entry's master password, ESD2MRPW, is supplied to allow the command to alter the entry's passwords.
2. The MASTERPW, CONTROLPW, UPDATEPW, and READPW parameters respecify passwords for the data set.

If the ALTER command was successful, the LISTCAT command lists the entire cluster entry.

The third ALTER command shows how to alter an existing entry-sequenced data set into a linear data set.

1. ALIAS02.ESDS1.CLUST01 names the entry-sequenced data set. (*Note*: This data set was defined with a control interval size of 4096 bytes.)
2. TYPE specifies the altered form of the original data set.
3. CATALOG specifies the location of the original data set.

If the ALTER command was successful, the LISTCAT command lists the entire cluster entry.

Example 4: Copying and Printing

This example shows various techniques that can be used to load and print data sets using the REPRO and PRINT commands. This example depends on the successful completion of examples 1 and 2 for the existence of VSAM data sets into which records are loaded. This example also requires that various non-VSAM data sets exist, so that data records can be loaded into the VSAM data sets.

```
//COPYPRNT  JOB  ...
//STEP1     EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=A
//SYSABEND  DD   SYSOUT=A
//AMSDUMP   DD   SYSOUT=A
//* NONVSAM INDEXED-SEQUENTIAL DATA SET
//INDSET1   DD   DSNAME=D40.IF100,DISP=OLD,DCB=DSORG=IS,
//             VOL=SER=VSER03,UNIT=3380
//* NONVSAM SEQUENTIAL DATA SET (VARIABLE-LENGTH
//*     RECORDS)
//INDSET2   DD   DSNAME=D40.SVB200,DISP=OLD,
//             VOL=SER=VSER03,UNIT=3380
//* NONVSAM SEQUENTIAL DATA SET (FIXED-LENGTH
//*     RECORDS)
//INDSET3   DD   DSNAME=D40.SF100,DISP=OLD,
//             VOL=SER=VSER03,UNIT=3380
//* NONVSAM  SEQUENTIAL DATA SET
//INDSET4   DD   DSNAME=EXAMPL1.NONVSAM1,DISP=OLD,
//             VOL=SER=VSER02,UNIT=3380
//* LINEAR DATA SET
//INDSE     DD   DSNAME=ALIAS02.LINEAR.CLUST01,DISP=OLD,
//             VOL=SER=VSER03,UNIT=3380,AMP=AMORG
//SYSIN     DD *

   /* LOAD A VSAM KEY-SEQUENCED DATA SET */
   /* FROM AN ISAM DATA SET */

   REPRO INFILE(INDSET1) -
         OUTDATASET(ALIAS01.EXAMPLE.KSDS1/KSD1PSWD)

   IF MAXCC = 0 -
      THEN -
      PRINT  INDATASET(ALIAS01.EXAMPLE.KSDS1/KSD1PSWD) -
             FROMKEY(X'40F0F0F0F0F6')

   /* LOAD A VSAM ENTRY-SEQUENCED DATA SET FROM AN */
   /* EXISTING VARIABLE UNBLOCKED SAM DATA SET */

   REPRO     INFILE(INDSET2) -
             OUTDATASET(ALIAS02.ESDS1.CLUST01/ESD1UPPW) -
             COUNT(030)

   IF LASTCC = 0 -
      THEN -
      PRINT  INDATASET(ALIAS01.ESDS1.CLUST01) -
             FROMADDRESS(170) -
             HEX

   /* LOAD A VSAM RELATIVE RECORD DATA SET FROM AN */
   /* EXISTING SAM DATA SET*/
```

```
          REPRO     INFILE(INDSET3) -
                    OUTDATASET(MAST01.RRDS1) -
                    SKIP(10)

          IF LASTCC = 0 -
             THEN -
             PRINT  INDATASET(MAST01.RRDS1) -
                    TONUMBER(25)

          /* PRINT THE CONTENTS OF THE SAM DATA SET */

             PRINT INFILE(INDSET3) -
                    COUNT(20) -
                    CHARACTER

          /* LOAD A VSAM KEY-SEQUENCED DATA SET */
          /* FROM A NONVSAM DATA SET */

          REPRO     INFILE(INDSET4) -
                    OUTDATASET(MAST01.KSDS2)

          IF LASTCC = 0 -
             THEN -
             PRINT  INDATASET(MAST01.KSDS2) -
                    FROMKEY(AAAAJA) -
                    TOKEY(AAAAJ9)

          /* LOAD A LINEAR DATA SET FROM ANOTHER */
          /* LINEAR DATA SET, THEN PARTIALLY PRINT THE */
          /* CONTENTS OF THE NEWLY LOADED DATA SET */

          REPRO     INFILE(INDSET5) -
                    OUTDATASET(ALIAS02.LINEAR.CLUST02) -


          IF LASTCC = 0 -
             THEN -
             PRINT  INDATASET(ALIAS02.LINEAR.CLUST02) -
                    FROMADDRESS(4096) -
                    TOADDRESS(8191)
       /*
```

Explanation of Job Control Language Statements

1. The INDSET1 DD statement describes the ISAM data set, which is to be copied into the VSAM data set ALIAS01.EXAMPLE.KSDS1.
2. The INDSET2 DD statement describes the variable-length SAM data set, which is to be copied into the VSAM data set ALIAS02.EXAMPLE.ESDS1.
3. The INDSET3 DD statement describes the SAM data set, which is to be copied into the VSAM data set MAST01.RRDS1.
4. The INDSET4 DD statement describes the SAM data set. which is to be copied into the VSAM data set MAST01.KSDS2.
5. The INDSET5 DD statement describes the linear data set, which is to be copied into the linear data set ALIAS02.LINEAR.CLUST02.

Explanation of Commands

The first REPRO command causes a VSAM key-sequenced data set to be loaded from an ISAM data set.

1. The INFILE parameter identifies the data set containing the source data. (*Note:* Either INFILE or INDATASET is required.) The ddname of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter identifies the name of the data set to be loaded. (*Note:* Either OUTFILE or OUTDATASET is required.) The data set is dynamically allocated by access method services. The catalog containing the data set is located by its alias 'ALIAS01'. The update- or higher-level password of the VSAM data set is required.

If the REPRO operation was successfully executed, the contents of the VSAM key-sequenced data set just loaded are printed. The format of the listing is DUMP because the default is taken.

1. The INDATASET parameter identifies the name of the data set to be printed. (*Note:* Either INFILE or INDATASET is required.) The data set is dynamically allocated by access method services. The catalog containing the data set is located by its alias 'ALIAS01'. The update- or higher-level password of the VSAM data set is required.
2. The FROMKEY parameter specifies that printing is to begin with the record whose key (high-order three bytes) is greater than or equal to '00006' (that is, the character equivalent of X'40F0F0F0F0F6').

The second REPRO command causes a VSAM entry-sequenced data set to be loaded from an existing variable unblocked SAM data set.

1. The INFILE parameter identifies the data set containing the source data. The ddname of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter identifies the name of the data set to be loaded. The data set is dynamically allocated by access method services. The catalog containing the data set is located by its alias 'ALIAS02'. The update- or higher-level password of the VSAM data set is required.
3. The COUNT parameter specifies that the first 30 records of the SAM data set are to be loaded.

If the REPRO operation was successfully executed, the contents of the VSAM entry-sequenced data set just loaded are printed in hexadecimal format.

1. The INDATASET parameter identifies the name of the data set to be printed. The data set is dynamically allocated by access method services. The catalog containing the data set is located by its alias 'D40'. Because the data set is not read protected, no password is required.
2. The FROMADDRESS parameter specifies that the first record printed is that record whose relative byte address is exactly equal to 170.
3. The HEX parameter specifies that the listing is to be in hexadecimal format.

The third REPRO command causes a VSAM relative record data set to be loaded from an existing fixed SAM data set. The relative record data set can receive only fixed length records that equal its defined record length.

1. The INFILE parameter identifies the data set containing the source data; Therefore, the ddname of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter identifies the name of the data set to be loaded. The data set is dynamically allocated by access method services. The data set is cataloged in the master catalog; therefore, no JOBCAT or STEPCAT DD statement is required.
3. The SKIP parameter specifies that the first 10 records of the SAM data set are to be bypassed.

If the REPRO operation was successfully executed, the contents of the VSAM relative record data set just loaded are printed.

1. The INDATASET parameter identifies the name of the data set to be printed. The data set is dynamically allocated by access method services. The data set is cataloged in the master catalog; therefore, no JOBCAT or STEPCAT DD statement is required. Because the data set is not read protected, no password is required.
2. The TONUMBER parameter limits the output to those relative records with relative record number less than or equal to 25.

The PRINT command causes the contents of the SAM data set to be printed.

1. The INFILE parameter identifies the data set to be printed. The ddname of the DD statement for this data set must be identical to this name.
2. The COUNT parameter specifies that only 20 records are to be printed.
3. The CHARACTER parameter specifies that the listing is to be in character format.

The fourth REPRO command causes a VSAM key-sequenced data set to be loaded from a non-VSAM data set.

1. The INFILE parameter identifies the data set containing the source data. The ddname of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter identifies the name of the data set to be loaded. The data set is dynamically allocated by access method services. The data set is cataloged in the master catalog. Note that, because the cluster component is not password protected, although its data and index components are, a password is not required.

If the REPRO operation was successfully executed, the contents of the VSAM key-sequenced data set just loaded are printed. The FROMKEY and TOKEY parameters are used to limit the output to a specific range of keys.

1. The INDATASET parameter identifies the name of the data set to be printed. The data set is dynamically allocated by access method services. The data set is cataloged in the master catalog. No password is required, because the cluster component is not password protected.
2. The FROMKEY and TOKEY parameters specify the keys at which printing is to begin and end, respectively.

The last REPRO command causes a linear data set to be loaded into another linear data set. For a linear data set, REPRO must include the entire data set.

1. The INFILE parameter identifies the data set containing the source data. The ddname of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter identifies the name of the data set to be loaded. The data set is dynamically allocated by access method services.

If the REPRO operation was successfully executed, the contents of the linear data set just loaded is partially printed from relative byte address 4096 up to an RBA of 8191. This is the second 4K page of the linear data set.

Example 5: Record Replacement

This example shows techniques for modifying the contents of VSAM data sets using the REPRO command.

This example depends on the successful completion of examples 1 and 2 for the existence of nonempty VSAM data sets.

```
//REPLACE  JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//INDSET4  DD   DSNAME=EXAMPL2.NONVSAM2,DISP=OLD,
//             VOL=SER=VSER03,UNIT=DISK
//SYSIN    DD   *

   REPRO   INFILE(INDSET4) -
           OUTDATASET(MAST01.KSDS2) -
           REPLACE

   IF LASTCC = 0 -
      THEN -
      PRINT INDATASET(MAST01.KSDS2) -
              FROMKEY(AAAAJA) -
              TOKEY(AAAAJ9)

   REPRO   INDATASET(MAST01.RRDS1) -
           OUTDATASET(ALIAS02.ESDS1.CLUST01/ESD2UPPW) -
           REUSE

   IF LASTCC = 0 -
      THEN -
      PRINT INDATASET(ALIAS02.ESDS1.CLUST01)
/*
```

Explanation of Job Control Language Statements

The INDSET4 DD statement describes the non-VSAM data set to be copied into the VSAM data set MAST01.KSDS2.

Explanation of Commands

The first REPRO command causes records in the VSAM key-sequenced data set to be replaced with input from a non-VSAM data set.

1. The INFILE parameter identifies the data set containing the source data. (*Note:* Either INFILE or INDATASET is required.) The ddname of the DD statement for this data set must be identical to this name.

2. The OUTDATASET parameter identifies the target data set. (*Note:* Either OUTFILE or OUTDATASET is required.) Access method services dynamically allocate the data set. The data set is cataloged in the master catalog; therefore, no JOBCAT or STEPCAT DD statement is required. Because the cluster is not password protected, no password is required.

3. The REPLACE parameter causes replacement of a record in the output data set that has the same key as a record in the input data set. Records in the input data set, whose key is not already contained in the output data set, are inserted into the output data set.

If the REPRO operation was successfully executed, the contents of the VSAM key-sequenced data set just changed are printed.

1. The INDATASET parameter identifies the data set to be printed. (*Note:* Either INFILE or INDATASET is required.) Access method services dynamically allocate the data set. The data set is cataloged in the master catalog; therefore, no JOBCAT or STEPCAT DD statement is required. Because the cluster is not password protected, no password is required.
2. The FROMKEY and TOKEY parameters specify the keys at which printing is to begin and end, respectively.

The second REPRO command causes the VSAM entry-sequenced data set to be loaded from the VSAM relative record data set.

1. The INDATASET parameter identifies the source data set. Access method services dynamically allocate the data set. The data set is cataloged in the master catalog. Because the cluster is unprotected, no password is required.
2. The OUTDATASET parameter identifies the target data set. Access method services dynamically allocate the data set. The catalog containing the data set is located by its alias "ALIAS02." The update- or higher-level password is required to load the data set.
3. The REUSE parameter specifies that any records already in the entry-sequenced data set output are to be overwritten because the entry-sequenced data set was defined with the REUSE attribute.

If the REPRO operation was successfully executed, the entire contents of the reloaded VSAM entry-sequenced data set are printed.

> The INDATASET parameter identifies the data set to be printed. Access method services dynamically allocate the data set. The catalog containing the data set is located by its alias "ALIAS02." Because no read password exists for this data set, no password is required.

Example 6: Creating an Alternate Index and Its Path

This example defines an alternate index over a previously loaded VSAM key-sequenced base cluster, defines a path over the alternate index to provide a means for processing the base cluster through the alternate index, and builds the alternate index. The alternate index, path, and base cluster must all be defined in the same catalog, in this case, the master catalog. Because the master catalog is recoverable, access method services dynamically allocate the volume containing the base cluster's index component in order to access its recovery area.

This example depends on the successful completion of examples 2, 4, and 5 for the existence of the nonempty VSAM base cluster MAST01.KSDS2.

```
//MAKEAIX  JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//IDCUT1   DD   DSNAME=SORT.WORK.ONE,DISP=OLD,
//             AMP='AMORG',VOL=SER=VSER04,UNIT=3380
//IDCUT2   DD   DSNAME=SORT.WORK.TWO,DISP=OLD,
//             AMP='AMORG',VOL=SER=VSER04,UNIT=3380
//SYSIN    DD   *

     DEFINE ALTERNATEINDEX -
          (NAME(MAST01.AIX) -
          RELATE(MAST01.KSDS2) -
          MASTERPW(AIXMRPW) -
          UPDATEPW(AIXUPPW) -
          KEYS(3 0) -
          RECORDSIZE(40 50) -
          VOLUMES(VSER01) -
          CYLINDERS(2 1) -
          NONUNIQUEKEY -
          UPGRADE ) -
        CATALOG(ICFMAST1/MASTMPW1)


     DEFINE PATH -
          (NAME(MAST01.PATH) -
          PATHENTRY(MAST01.AIX/AIXMRPW) -
          READPW(PATHRDPW)) -
        CATALOG(ICFMAST1/MASTMPW1)

     BLDINDEX INDATASET(MAST01.KSDS2) -
              OUTDATASET(MAST01.AIX/AIXUPPW) -
              CATALOG(ICFMAST1/MASTMPW1)

     PRINT INDATASET(MAST01.PATH/PATHRDPW)
/*
```

Explanation of Job Control Language Statements

The IDCUT1 and IDCUT2 DD statements describe the dsnames and a volume containing data space made available to BLDINDEX for defining and using two sort work data sets in the event an external sort is performed. This data space is not used by BLDINDEX if enough virtual storage is available to perform an internal sort.

Explanation of Commands

The first DEFINE command defines a VSAM alternate index over the base cluster MAST01.KSDS2.

1. The NAME parameter is required and names the object being defined.
2. The RELATE parameter is required and specifies the name of the base cluster over which the alternate index is defined.
3. The MASTERPW and UPDATEPW parameters specify the master and update passwords, respectively, for the alternate index.
4. The KEYS parameter specifies the length of the alternate key and its offset in the base cluster record.
5. The RECORDSIZE parameter specifies the length of the alternate index record. Because the alternate index is being defined with the NONUNIQUEKEY attribute, it must be large enough to contain the prime keys for all occurrences of any one alternate key.
6. The VOLUMES parameter is required and specifies the volume containing the alternate index MAST01.AIX.
7. The CYLINDERS parameter specifies the amount of space to be allocated to the alternate index. A space parameter is required.
8. The NONUNIQUEKEY parameter specifies that the base cluster can contain multiple occurrences of any one alternate key.
9. The UPGRADE parameter specifies that the alternate index is to reflect all changes made to the base cluster records, for example, additions or deletions of base cluster records.
10. Because the master catalog is password-protected, the CATALOG parameter is required. It specifies the name of the master catalog and its update or master password, which is required to define into a protected catalog.

The BLDINDEX command builds an alternate index. It is assumed that enough virtual storage is available to perform an internal sort. However, DD statements with the default ddnames of IDCUT1 and IDCUT2 have been provided for two external sort work data sets in the event that the assumption is incorrect and an external sort must be performed.

1. The INDATASET parameter identifies the base cluster. Access method services dynamically allocate the base cluster. The base cluster's cluster entry is not password protected even though its data and index components are.
2. The OUTDATASET parameter identifies the alternate index. Access method services dynamically allocate the alternate index. The update- or higher-level password of the alternate index is required.
3. The CATALOG parameter specifies the name of the master catalog. If it is necessary for BLDINDEX to use external sort work data sets, they will be defined in and deleted from the master catalog. The master password permits these actions.

The second DEFINE command defines a path over the alternate index. After the alternate index has been built, opening with the path name causes processing of the base cluster via the alternate index.

1. The NAME parameter is required and names the object being defined.
2. The PATHENTRY parameter is required and specifies the name of the alternate index over which the path is defined and its master password.
3. The READPW parameter specifies a read password for the path; it will be propagated to master-password level.
4. The CATALOG parameter is required, because the master catalog is password protected. It specifies the name of the master catalog and its update or master password, which is required to define into a protected catalog.

The PRINT command causes the base cluster to be printed by means of the alternate key, using the path defined to create this relationship. The INDATASET parameter identifies the path object. Access method services dynamically allocate the path. The read password of the path is required.

Example 7: Exporting VSAM Data Sets

This example shows various methods of exporting data sets to provide backup and portability.

Note: This example depends on the successful completion of previous examples for the existence of the various objects to be exported.

```
//EXPORT   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//RECEIVE  DD   DSNAME=PORTABLE.DSET1,SPACE=(CYL,(1,1)),
//             UNIT=3380,DISP=(NEW,KEEP),VOL=SER=VSER01,
//             DCB=BLKSIZE=6000
//SYSIN    DD   *

     EXPORT -
          ALIAS01.EXAMPLE.KSDS1/KSD1PSWD -
          PURGE -
          OUTFILE(RECEIVE)

//STEP2    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//RECEIVE  DD   DSNAME=PORTABLE.DSET2,SPACE=(CYL,(1,1)),
//             UNIT=3380,DISP=(NEW,KEEP),VOL=SER=VSER01
//SYSIN    DD   *

     EXPORT -
          ALIAS02.ESDS1.CLUST01/ESD2PWMR -
          OUTFILE(RECEIVE) -
          TEMPORARY -
          INHIBITSOURCE -
          INHIBITTARGET

//STEP3    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//RECEIVE  DD   DSNAME=PORTABLE.DSET3,SPACE=(CYL,(1,1)),
//             DISP=(NEW,KEEP),VOL=SER=VSER01,UNIT=3380
//SYSIN    DD    *

     EXPORT -
          MAST01.AIX/MASTMPW1 -
          OUTFILE(RECEIVE)

//STEP4    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//RECEIVE  DD   DSNAME=PORTABLE.DSET4,SPACE=(CYL(1,1)),
//             DISP=(NEW,KEEP),VOL=SER=VSER01,UNIT=3380
//SYSIN    DD    *

     EXPORT -
          MAST01.KSDS2/MASTMPW1 -
          OUTFILE(RECEIVE)

//STEP5    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//RECEIVE  DD   DSNAME=PORTABLE.DSET5,SPACE=(CYL,(3,1)),
//             UNIT=3380,DISP=(NEW,KEEP),VOL=SER=VSER3
//SYSIN    DD   *

     EXPORT -
          ALIAS02.LINEAR.CLUST01 -
          OUTFILE(RECEIVE) -
          CIMODE
/*
```

Explanation of Job Control Language Statements

The RECEIVE DD statements describe the portable data sets. The record format (VBS) and logical record length are set by EXPORT. Except where overridden as in STEP1, the block size is set by EXPORT as 2048.

Explanation of Commands

The first EXPORT command causes a key-sequenced VSAM data set to be exported from a user catalog. When it has been exported, the key-sequenced data set is deleted from the user catalog.

Note: When an object is exported, the record format of its records on the portable file is "VBS", and the EXPORT process determines the appropriate record size. However, the RECEIVE DD statement specifies a block size (BLKSIZE=6000) to override the block size used by the EXPORT process (2048 bytes) and to improve performance.

1. ALIAS01.EXAMPLE.KSDS1 names the key-sequenced VSAM data set being exported. Its password, KSD1PSWD, is also supplied. (When the data set was defined, its read password propagated upward and all passwords for the data set are KSD1PSWD.) Access method services dynamically allocate the cluster. The catalog containing the cluster is located through its alias 'ALIAS01', which is the high-level qualifier of the cluster name.

2. The PURGE parameter is required, because the data set was defined with a retention period of 365 days. The data set cannot be exported permanently (that is, deleted from the catalog after its copy is made in the portable file) unless the PURGE parameter is specified to override its cataloged retention period.

3. The OUTFILE parameter names the DD statement that describes and allocates the output data set. (*Note:* Either OUTFILE or OUTDATASET is required.)

The second EXPORT command causes an entry-sequenced VSAM data set to be exported from a user catalog. When it has been exported, the data set's entry in the user catalog is marked "temporary export" and "inhibit update," which prevents the data set from being modified in any way. The user's program can only read the data set's records. In addition, when the data set's copy is imported into another system catalog, the data set's entry in the new, or "target," catalog is marked "inhibit update."

1. ALIAS02.EXAMPLE.ESDS1 names the entry-sequenced data set being exported. Its master password, ESD1PWMR, is also supplied. (Example 7 shows how the data set's passwords were changed.) Access method services dynamically allocate the cluster. The catalog containing the cluster is located through its alias 'ALIAS02', which is the high-level qualifier of the cluster name.
2. The OUTFILE parameter names the DD statement that describes and allocates the output data set.
3. The TEMPORARY parameter specifies that the data set is not to be deleted from the catalog when it is exported.
4. The INHIBITSOURCE parameter specifies that the data set that remains in the "source" catalog and system is not to be updated or modified.
5. The INHIBITTARGET parameter specifies that the data set's exported copy is not to be updated or modified when it has been imported into the "target" catalog and system.

The third and fourth EXPORT commands cause the alternate index and base cluster to be exported from the master catalog. Any paths defined over either object are exported with their PATHENTRY object. Because the export is permanent, both the base cluster and the alternate index are deleted from the catalog. The alternate index must be exported first, because a delete of the base cluster causes deletion of all objects defined over it.

The third EXPORT command causes an alternate index to be exported from the master catalog.

1. The name of the alternate index being exported is required. A master password is required for the deletion and to allow VSAM locates against both the alternate index and path to obtain the catalog information (including passwords) to be exported. The master password of the catalog covers all requirements. Access method services dynamically allocate the alternate index.
2. The OUTFILE parameter names the DD statement that describes and allocates the output data set.

The fourth EXPORT command causes a base cluster to be exported from the master catalog.

1. The name of the base cluster being exported is required. Because the cluster level is not protected, no password would be required for the deletion. However, a password is required for the VSAM locates against the data and index components to obtain the catalog information (including passwords) to be exported. Because only one password can be supplied, it must be that of the master catalog. Access method services dynamically allocate the cluster.
2. The OUTFILE parameter names the DD statement that describes and allocates the output data set.

The fifth EXPORT command causes the export of a linear data set from the master catalog. To be successful, both the IMPORT source and the RECEIVE data sets must be linear data sets.

1. The name of the data set being exported is required.
2. The OUTFILE parameter names the DD statement that describes and allocates the output data set.
3. CIMODE is forced for EXPORT. If RECORDMODE is specified, it will be overridden.

Example 8: Importing VSAM Data Sets

This example shows various methods of importing data sets.

Note: This example depends on the successful completion of Example 7, which created a portable data set that contains a copy of each VSAM data set to be imported.

```
//IMPORT   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//SOURCE   DD   DSNAME=PORTABLE.DSET4,
//             DISP=(OLD,DELETE),VOL=SER=VSER01,UNIT=3380
//SYSIN    DD   *

     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(MAST01.KSDS2) -
          CATALOG(ICFMAST1/MASTMPW1)

//STEP2    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//SOURCE   DD   DSNAME=PORTABLE.DSET3,
//             DISP=(OLD,DELETE),VOL=SER=VSER01,UNIT=3380
//SYSIN    DD   *

     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(MAST01.AIX/AIXMRPW) -
          CATALOG(ICFMAST1/MASTMPW1)

//STEP3    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//SOURCE   DD   DSNAME=PORTABLE.DSET2,DCB=LRECL=3004,
//             UNIT=3380,DISP=(OLD,DELETE),VOL=SER=VSER01
//RECEIVE  DD   DSN=ALIAS02.ESDS1.CLUST01,DISP=SHR
//SYSIN    DD   *

     IMPORT -
          INFILE(SOURCE) -
          OUTFILE(RECEIVE) -
          CATALOG(USERCAT2/UCATMRPW)

//STEP4    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//SOURCE   DD   DSNAME=PORTABLE.DSET1,
//             UNIT=3380,DISP=(OLD,DELETE),VOL=SER=VSER01,
//             DCB=(LRECL=479,BLKSIZE=6000)
//SYSIN    DD   *
```

```
     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(ALIAS02.EXAMNEW.KSDS1) -
          CATALOG(USERCAT2/UCATUPPW) -
          OBJECTS -
             ((ALIAS01.EXAMPLE.KSDS1 -
              VOLUMES(VSER03) -
              NEWNAME(ALIAS02.EXAMNEW.KSDS1)) -
             (ALIAS01.EXAMPLE.KSDS1.DATA -
              NEWNAME(ALIAS02.EXAMPLE.KSDS1.DATA)) -
             (ALIAS01.EXAMPLE.KSDS1.INDEX -
              NEWNAME(ALIAS02.EXAMPLE.KSDS1.INDEX)))

//STEP5    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//SOURCE   DD   DSNAME=PORTABLE.DSET5,UNIT=3380,DISP=(OLD,DELETE),
//             VOL=SER=VSER03
//SYSIN    DD   *

     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(ALIAS02.LINEAR.CLUST01) -
          CATALOG(USERCAT2/UCATMRPW)
/*
```

Explanation of Job Control Language Statements

- The SOURCE DD statements describe the portable data sets. Unless overridden by DCB parameters, the EXPORT command sets the block size to 2048 and the record length to block size minus 4.
- The RECEIVE DD statement in STEP3 describes and allocates ALIAS02.ESDS1.CLUST01, which was exported with the TEMPORARY attribute.

Explanation of Commands

The first and second IMPORT commands import the base cluster and alternate index exported in Example 11 into the master catalog. The importation causes each component to be newly defined. Because the alternate index cannot be defined until the base cluster has been defined, the base cluster must be imported first. The importation causes any paths over the exported objects to be redefined.

The first IMPORT command causes the base cluster to be imported into the master catalog.

1. The INFILE parameter names the DD statement that describes and allocates the portable data set. (**Note:** Either INFILE or INDATASET is required.)
2. The OUTDATASET parameter identifies the data set being imported. (**Note**: Either OUTFILE or OUTDATASET is required.) Access method services dynamically allocates the data set after it has been defined; this also provides access to the catalog recovery area.
3. Because the master catalog is password protected, the CATALOG parameter is required. It specifies the name of the master catalog and its update password, required to define into a protected catalog.

The second IMPORT command causes an alternate index to be imported into the master catalog.

1. The INFILE parameter names the DD statement that describes and allocates the portable data set.
2. The OUTDATASET parameter identifies the data set being imported. Access method services dynamically allocates the alternate index after it has been defined; this provides access to the catalog recovery area.
3. Because the master catalog is password protected, the CATALOG parameter is required. It specifies the name of the master catalog and its update password, required to define into a protected catalog.

The third IMPORT command causes the previously exported entry-sequenced VSAM data set to be imported. The data set is imported into the catalog from which it was exported. The portable data set (that is, the copy being imported) replaces the copy that exists in user catalog USERCAT2. With an entry name of ALIAS02.ESDS1.CLUST01, the IMPORT process searches user catalog USERCAT2 for the entry. It deletes that entry, then redefines a cluster entry using the catalog information obtained from the portable data set. Because the data set was exported with the TEMPORARY attribute (see the previous example), the IMPORT command doesn't need to supply volume information.

Note: The SOURCE DD statement specifies a record size (LRECL=3004), because the largest record in the portable data set is 3000 bytes (that is, LRECL = (largest-record size) + 4). Otherwise, the default record size, block size minus 4, would be erroneously used by the EXPORT command. In the two previous steps, the default record size was used.

1. The INFILE parameter names the DD statement that describes and allocates the portable data set to be imported.
2. The OUTFILE parameter names the DD statement that describes and allocates the data set to be imported. Note that, because the data set was exported with the TEMPORARY attribute, it exists in the catalog USERCAT2 at step allocation time.
3. Because the catalog is password protected, the CATALOG parameter is required. It names the catalog that is to contain the imported data set. The catalog's master password is supplied, and allows the IMPORT process to delete the existing entry in the catalog and redefine a new entry for the entry-sequenced VSAM data set.

The fourth IMPORT command causes the previously exported key-sequenced VSAM data set, ALIAS01.EXAMPLE.KSDS1 to be imported from its copy (that is, the first portable data set on the magnetic tape reel) to a different user catalog than it was exported from, USERCAT2. The IMPORT command renames the data set and each of its components, as specified with the NEWNAME parameters, and specifies that the data set is to reside on a volume different from which it was exported. The high-level qualifier (ALIAS02) of the new component's name is the alias name of the user catalog USERCAT2.

Note: The SOURCE DD statement describes the portable data set created in STEP1 of Example 7. The block size parameter is included to emphasize that information specified when the data set is imported must be the same as that specified when the data set is exported. The LRECL parameter is not required, because maximum record size is 475 bytes and the default (block size minus 4) is adequate. By specifying a record size, however, the default is overridden and virtual storage is used more efficiently. When record size is specified, it is the largest record size + 4.

1. The INFILE parameter names the DD statement that describes and allocates the portable data set containing the to-be-imported VSAM data set.
2. The OUTDATASET parameter identifies the renamed data set. Access method services dynamically allocates the data set after it has been defined.
3. Because the catalog is password protected, the CATALOG parameter is required. It names the catalog that is to contain the imported data set's entry. The catalog's update password is supplied and allows the data set to be imported into the catalog.
4. The OBJECTS parameter identifies the volume that is to contain the imported data set and specifies new names for each of the data set's components. The OBJECTS parameter identifies each entry of the imported data set with its original entry name, then specifies information that is to replace the information found in the portable data set's imported catalog entries.

The fifth IMPORT command processes a linear data set. To be successful, both the source and EXPORT receive data sets must be linear.

1. The INFILE parameter names the DD statement that describes and allocates the portable data set containing the to-be-imported VSAM data set.
2. The OUTDATASET parameter identifies the renamed data set. Access method services dynamically allocates the data set after it has been defined.
3. CATALOG names the catalog that is to contain the imported data set's entry.

Note: CIMODE cannot be specified because IMPORT uses the same mode as EXPORT.

Example 9: Deleting a Linear Data Set

This example shows how to delete a linear data set.

```
//DELETE   JOB ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSABEND DD   SYSOUT=A
//AMSDUMP  DD   SYSOUT=A
//DASD     DD   DISP=OLD,UNIT=3380,VOL=SER=VSER03
//SYSIN    DD   *

        DELETE (ALIAS02.LINEAR.CLUST01) -
                CLUSTER -
                FILE(DASD) -
                PURGE -
                CATALOG(USERCAT2/USERMPW)
//*
```

Explanation of Job Control Language Statements

The DASD DD statement identifies the disposition and location of the object being deleted.

Explanation of Commands

1. The complete name of the object being deleted is listed.
2. CLUSTER specifies that the object being deleted is a cluster.
3. FILE specifies the name of the DD statement that identifies the disposition and location of the object being deleted.
4. PURGE specifies the object to be deleted regardless of the specified retention period.
5. CATALOG specifies the name of the catalog that contains the object being deleted.

Appendix D. Processing the Index of a Key-Sequenced Data Set

Product-Sensitive Programming Interface

This appendix is intended to help you diagnose problems you may have with an index of a key-sequenced data set. It contains product-sensitive programming interfaces provided by MVS/XA Data Facility Product. Installation exits and other product-sensitive interfaces are provided to allow your installation to perform tasks such as product tailoring, monitoring, modification, or diagnosis. They are dependent on the detailed design or implementation of the product. Such interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

VSAM allows you to access the index of a key-sequenced data set. This may be useful to you if your index is damaged or if pointers are lost and you want to know exactly what the index contains. You should not attempt to duplicate or substitute the index processing done by VSAM during normal access to data records.

How to Gain Access to a Key-Sequenced Data Set's Index

You can gain access to the index of a key-sequenced data set in one of two ways:

- By opening the cluster and using the GETIX and PUTIX macros
- By opening the index component alone and using the macros for normal data processing (GET, PUT, and so forth)

Accessing an Index with GETIX and PUTIX

To process the index of a key-sequenced data set with GETIX and PUTIX, you must open the cluster with ACB MACRF=(CNV,...) specified. CNV provides for control interval access, which you use to gain access to the index component.

Access using GETIX and PUTIX is direct, by control interval: VSAM requires RPL OPTCD=(CNV,DIR). The search argument for GETIX is the RBA of a control interval. The increment from the RBA of one control interval to that of the next is control interval size for the index.

GETIX can be issued either for update or not for update. VSAM recognizes OPTCD=NUP or UPD but interprets OPTCD=NSP as NUP.

The contents of a control interval cannot be inserted by way of PUTIX. VSAM requires OPTCD=UPD. The contents must previously have been retrieved for update by way of GETIX.

RPL OPTCD=MVE or LOC may be specified for GETIX, but only OPTCD=MVE is valid for PUTIX. If you retrieve with OPTCD=LOC, you must change OPTCD to MVE to store. With OPTCD=MVE, AREALEN must be at least index control interval size.

Beyond these restrictions, access to an index by way of GETIX and PUTIX follows the rules under Chapter 7, "Processing Control Intervals" on page 87.

Opening the Index Component Alone

You can gain addressed or control interval access to the index component of a key-sequenced cluster by opening the index component alone and using the request macros for normal data processing. To open the index component alone, specify: **DSNAME=***indexcomponentname* in the DD statement identified in the ACB (or GENCB) macro.

You can gain access to index records with addressed access and to index control intervals with control interval access. The use of these two types of access for processing an index is identical in every respect with their use for processing a data component.

Processing the index component alone is identical to processing an entry-sequenced data set; an index itself has no index and thus cannot be processed by keyed access.

Prime Index

A key-sequenced data set always has an index that relates key values to the relative locations of the the logical records in a data set. This index is called the *prime index*. The prime index, or simply index, has two uses:

- To locate the collating position when inserting records
- To locate records for retrieval

When initially loading a data set, records must be presented to VSAM in key sequence. The index for a key-sequenced data set is built automatically by VSAM as the data set is loaded with records. The index is stored in control intervals. An index control interval contains one pointer to each index control interval in the next lower level, or one entry for each data control interval in a control area.

When a data control interval is completely loaded with logical records, free space, and control information, VSAM makes an entry in the index. The entry consists of the *highest possible key* in the data control interval and a pointer to the beginning of that control interval. The highest possible key in a data control interval is one less than the value of the first key in the next sequential data control interval. Figure 31 illustrates that a single index entry, such as **19**, contains all the information necessary to locate a logical record in a data control interval.

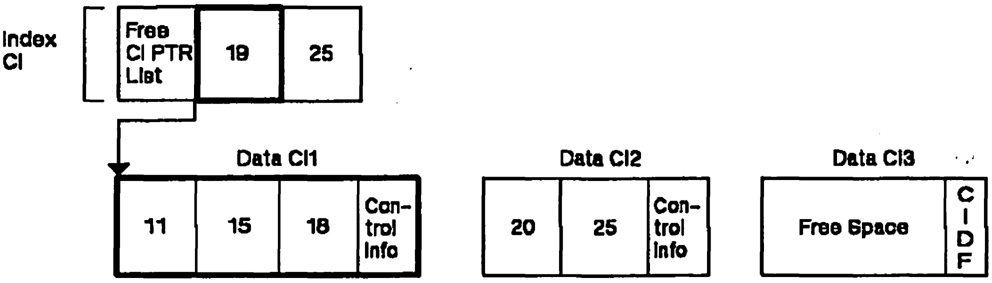


Figure 31. Relation of Index Entry to Data Control Interval

Figure 32 illustrates that a single index control interval contains all the information necessary to locate a record in a single data control area.

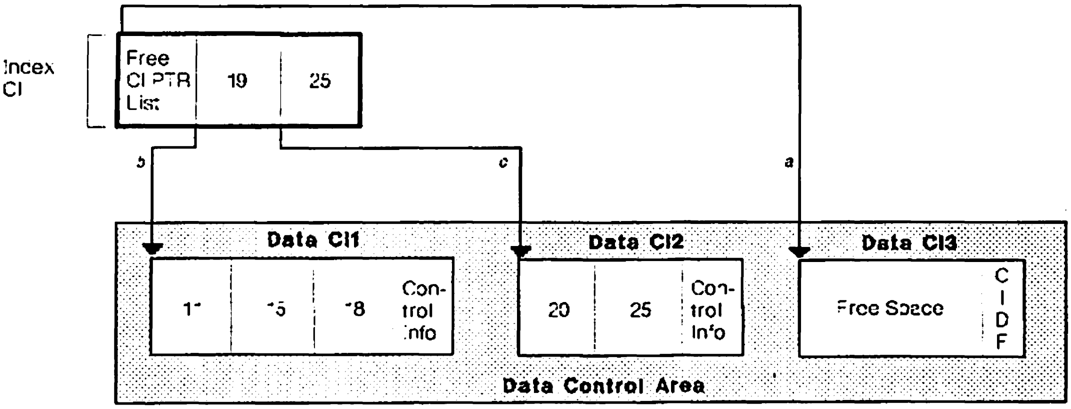


Figure 32. Relation of Index Control Interval to Data Control Area

The index contains the following entries:

a. A free control interval pointer list, which indicates available free space control intervals. Because this control area has a control interval that is reserved as free space, VSAM places a free space pointer in the index control interval to locate the free space data control interval.

b. 19, the highest possible key in data control interval 1. This entry points to the beginning of data control interval 1.

c. 25, the highest possible key in data control interval 2. This entry points to the beginning of data control interval 2.

Index Levels

A VSAM index can consist of more than one index level. Each level contains a set of records with entries giving the location of the records in the next lower level.

Sequence Set

The index records at the lowest level are the *sequence set*. There is one index sequence set level record for each control area in the data set. This sequence set record gives the location of data control intervals. An entry in a sequence set record consists of the highest possible key in a control interval of the data component, paired with a pointer to that control interval.

Index Set

If there is more than one sequence set level record, VSAM automatically builds another index level. Each entry in the second level index record points to one sequence set record. The records in all levels of the index above the sequence set are called the *index set*. An entry in an index set record consists of the highest possible key in an index record in the next lower level, and a pointer to the beginning of that index record. The highest level of the index always contains only a single record.

When you access records sequentially, VSAM refers only to the sequence set. It uses a horizontal pointer to get from one sequence set record to the next record in collating sequence. When you access records directly (not sequentially), VSAM follows vertical pointers from the highest level of the index down to the sequence set to find vertical pointers to data.

Figure 33 on page 191 illustrates the levels of a prime index and shows the relationship between sequence set index records and control areas. The sequence set shows both the horizontal pointers used for sequential processing, and the vertical pointers to the data set. Although the values of the keys are actually compressed in the index, the full key values are shown in the figure.

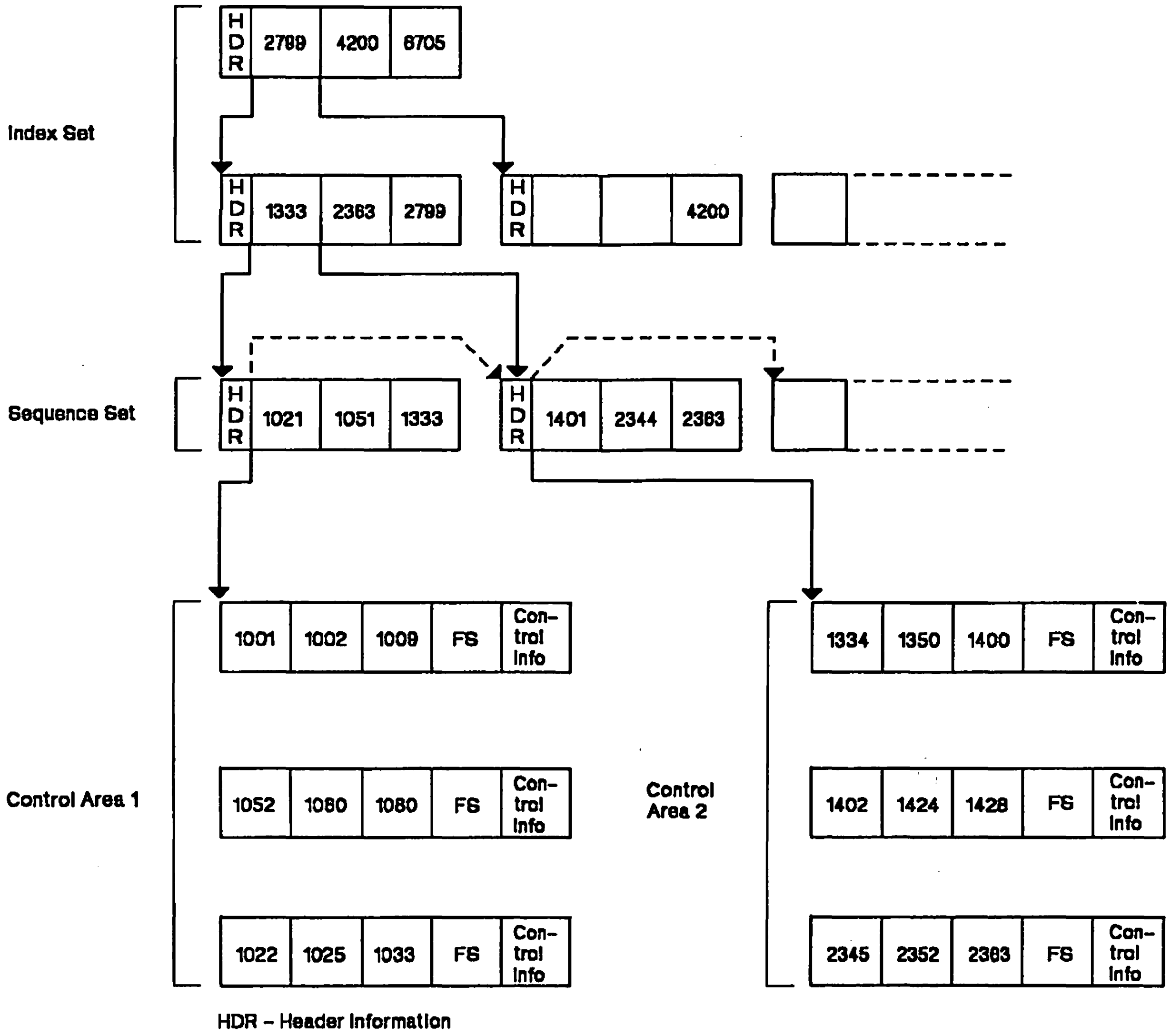


Figure 33. Levels of a Prime Index

Format of an Index Record

Index records are stored in control intervals the same as data records, except that only one index record is stored in a control interval, and there is no free space between the record and the control information. Consequently, there is only one RDF that contains the flag X'00' and the length of the record (a number equal to the length of the control interval minus 7). The CIDF also contains the length of the record (the displacement from the beginning of the control interval to the control information); its second number is 0 (no free space). The contents of the RDF and CIDF are the same for every used control interval in an index. The control interval after the last-used control interval has a CIDF filled with 0's, and is used to represent the software end-of-file (SEOF).

Index control intervals are not grouped into control areas as are data control intervals. When a new index record is required, it is stored in a new control interval at the end of the index data set. As a result, the records of one index level are not segregated from the records of another level, except when the sequence set is separate from the index set. The level of each index record is identified by a field in the index header (see "Header Portion".)

When an index record is replicated on a track, each copy of the record is identical to the other copies. Replication has no effect on the contents of records.

Figure 34 shows the parts of an index record.

Header	Free CI PTR List	Unused Space	Index – Entry Portion

Figure 34. General Format of an Index Record

An index record contains the following:

- A 24-byte header containing control information about the record.
- For a sequence set index record governing a control area that has free control intervals, there are entries pointing to those free control intervals.
- Unused space, if any.
- A set of index entries used to locate, for an index set record, control intervals in the next lower level of the index, or, for a sequence set record, used control intervals in the control area governed by the index record.

Header Portion

The first 24 bytes of an index record is the header, which gives control information about the index record. Figure 35 on page 190 shows its format. All lengths and displacements are in bytes. The discussions in the following two sections amplify the meaning and use of some of the fields in the header.

Figure 35 (Page 3 of 3). Format of the Header of an Index Record

Field	Offset	Length	Description
IXHLL	0(0)	2	Index record length. The length of the index record is equal to the length of the control interval minus 7.
IXHFLPLN	2(2)	1	Index entry control information length. This is the length of the last three of the four fields in an index entry. (The length of the first field is variable.) The length of the control information is 3, 4, or 5 bytes.

Figure 35 (Page 2 of 3). Format of the Header of an Index Record

Field	Offset	Length	Description
IXHPTLS	3(3)	1	Vertical-pointer-length indicator. The fourth field in an index entry is a vertical pointer to a control interval. In an index set record, the pointer is a binary number that designates a control interval in the index. The number is calculated by dividing the RBA of the control interval by the length of the control interval. To allow for a possibly large index, the pointer is always 3 bytes. In a sequence set record, the pointer is a binary number, beginning at 0, and calculated the same as for index set record, that designates a control interval in the data control area governed by the sequence set record. A free-control-interval entry is nothing more than a vertical pointer. There are as many index entries and free control interval entries in a sequence set record as there are control intervals in a control area. Depending on the number of control intervals in a control area, the pointer is 1, 2, or 3 bytes. An IXHPTLS value of X'01' indicates a 1-byte pointer; X'03' indicates a 2-byte pointer; X'07' indicates a 3-byte pointer.
IXHBRBA	4(4)	4	Base RBA. In an index-set record, this is the beginning RBA of the index. Its value is 0. The RBA of a control interval in the index is calculated by multiplying index control interval length times the vertical pointer and adding the result to the base RBA. In a sequence set record, this is the RBA of the control area governed by the record. The RBA of a control interval in the control area is calculated by multiplying data control interval length times the vertical pointer and adding the result to the base RBA. Thus, the first control interval in a control area has the same RBA as the control area (length times 0, plus base RBA, equals base RBA).
IXHHP	8(8)	4	Horizontal-pointer RBA. This is the RBA of the next index record in the same level as this record. The next index record contains keys next in ascending sequence after the keys in this record.
	12(C)	4	Reserved.
IXHLV	16(10)	1	Level number. The sequence set is the first level of an index, and each of its records has an IXHLV of 1. Records in the next higher level have a 2, and so on.
	17(11)	1	Reserved.
IXHFSO	18(12)	2	Displacement to the unused space in the record. In an index set record, this is the length of the header (24)—there are no free control interval entries. In a sequence set record, the displacement is equal to 24, plus the length of free control interval entries, if any.

Figure 35 (Page 1 of 3). Format of the Header of an Index Record

Field	Offset	Length	Description
IXHLEO	20(14)	2	Displacement to the control information in the last index entry. The last (leftmost) index entry contains the highest key in the index record. In a search, if the search-argument key is greater than the highest key in the preceding index record but less than or equal to the highest key in this index record, then this index record governs either the index records in the next lower level that have the range of the search-argument key or the control area in which a data record having the search-argument key is stored.
IXHSEO	22(16)	2	Displacement to the control information in the last (leftmost) index entry in the first (rightmost) section. Index entries are divided into sections to facilitate a quick search. Individual entries are not examined until the right section is located.

Free Control Interval Entry Portion

If the control area governed by a sequence set record has free control intervals, the sequence set record has entries pointing to those free control intervals. Each entry is 1, 2, or 3 bytes long (indicated by IXHPTLS in the header: the same length as the pointers in the index entries).

The entries come immediately after the header. They are used from right to left. The rightmost entry is immediately before the unused space (whose displacement is given in IXHFSO in the header). When a free control interval gets used, its free entry is converted to zero, the space becomes part of the unused space, and a new index entry is created in the position determined by ascending key sequence.

Thus, the free control interval entry portion contracts to the left, and the index entry portion expands to the left. When all the free control intervals in a control area have been used, the sequence set record governing the control area no longer has free control interval entries, and the number of index entries equals the number of control intervals in the control area. Note that if the index control interval size was specified with too small a value, it is possible for the unused space to be used up for index entries before all the free control intervals have been used, resulting in control intervals within a data control area that cannot be utilized.

Index Entry Portion

The index entry portion of an index record takes up all of the record that is left over after the header, the free control interval entries, if any, and the unused space.

Figure 36 on page 193 shows the format of the index entry portion of an index record. To improve search speed, index entries are grouped into sections, of which there are approximately as many as the square root of the number of entries. For example, if there are 100 index entries in an index record, they are grouped inio 10 sections of 10 entries each. (The number of sections does not change, even though the number of index entries increases as free control intervals get used.)

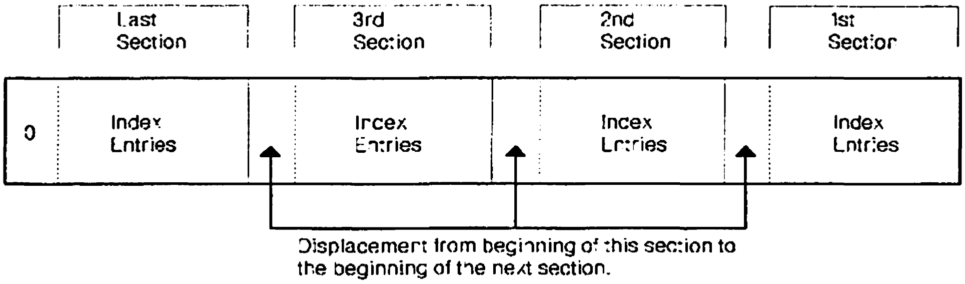
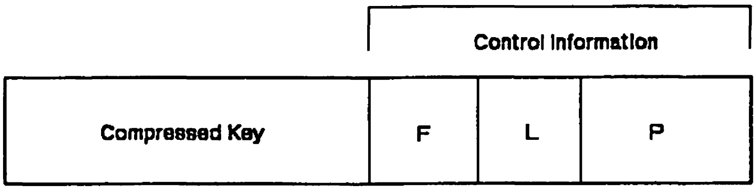


Figure 36. Format of the Index Entry Portion of an Index Record

The sections, and the entries within a section, are arranged from right to left. IXHLEO in the header gives the displacement from the beginning of the index record to the control information in the leftmost index entry. IXHSEO gives the displacement to the control information in the leftmost index entry in the rightmost section. You calculate the displacement of the control information of the rightmost index entry in the index record (the entry with the lowest key) by subtracting IXHFLPLN from IXHLL in the header (the length of the control information in an index entry from the length of the record).

Each section is preceded by a 2-byte field that gives the displacement from the control information in the leftmost index entry in the section to the control information in the leftmost index entry in the next section (to the left). The last (leftmost) section's 2-byte field contains 0's.

Figure 37 gives the format of an index entry.



F-Number of characters eliminated from the front
L-Number of characters left in key after compression
P-Vertical pointer

Figure 37. Format of an Index Entry

Key Compression

Index entries are variable in length within an index record, because VSAM compresses keys. That is, it eliminates redundant or unnecessary characters from the front and back of a key to save space. The number of characters that can be eliminated from a key depends on the relationship between that key and the preceding and following keys.

For front compression, VSAM compares a key in the index with the preceding key in the index and eliminates from the key those leading characters that are the same as the leading characters in the preceding key. For example, if key 12356 follows key 12345, the characters 123 are eliminated from 12356 because they are equal to the first three characters in the preceding key. The lowest key in an index record has no front compression; there is no preceding key in the index record.

There is an exception for the highest key in a section. For front compression, it is compared with the highest key in the preceding section, rather than with the preceding key. The highest key in the rightmost section of an index record has no front compression; there is no preceding section in the index record.

What is referred to as "rear compression" of keys is actually the process of eliminating the insignificant values from the end of a key in the index. The values eliminated may be represented by X'FF'. VSAM compares a key in the index with the following key in the data and eliminates from the key those characters to the right of the first character that are unequal to the corresponding character in the following key. For example, if the key 12345 (in the index) precedes key 12356 (in the data), the character 5 is eliminated from 12345 because the fourth character in the two keys is the first unequal pair.

The first of the control information fields gives the number of characters eliminated from the front of the key, and the second field gives the number of characters that remain. When the sum of these two numbers is subtracted from the full key length (available from the catalog when the index is opened), the result is the number of characters eliminated from the rear. The third field indicates the control interval that contains a record with the key.

The example in Figure 38 on page 198 gives a list of full keys and shows the contents of the index entries corresponding to the keys that get into the index (the highest key in each data control interval). A sequence-set record is assumed, with vertical pointers 1 byte long. The index entries shown in the figure from top to bottom are arranged from right to left in the assumed index record.

Key 12345 has no front compression because it is the first key in the index record. Key 12356 has no rear compression because, in the comparison between 12356 and 12357, there are no characters following 6, which is the first character that is unequal to the corresponding character in the following key.

You can always figure out what characters have been eliminated from the front of a key; you cannot figure out the ones eliminated from the rear. Rear compression, in effect, establishes the key in the entry as a boundary value instead of an exact high key. That is, an entry does not give the exact value of the highest key in a control interval, but gives only enough of the key to distinguish it from the lowest key in the next control interval. In Figure 38 on page 198, for example, the last three index keys, after rear compression, are 12401, 124, and 134. Data records with key fields between 12402 and 124FF are associated with index key 124; data records with key fields between 12500 and 134FF are associated with index key 134.

If the last data record in a control interval is deleted, and if the control interval does not contain the high key for the control area, then the space is reclaimed as free space. Space reclamation can be suppressed by setting the RPLNOCIR bit, which has an equated value of X'20', at offset 43 into the RPL.

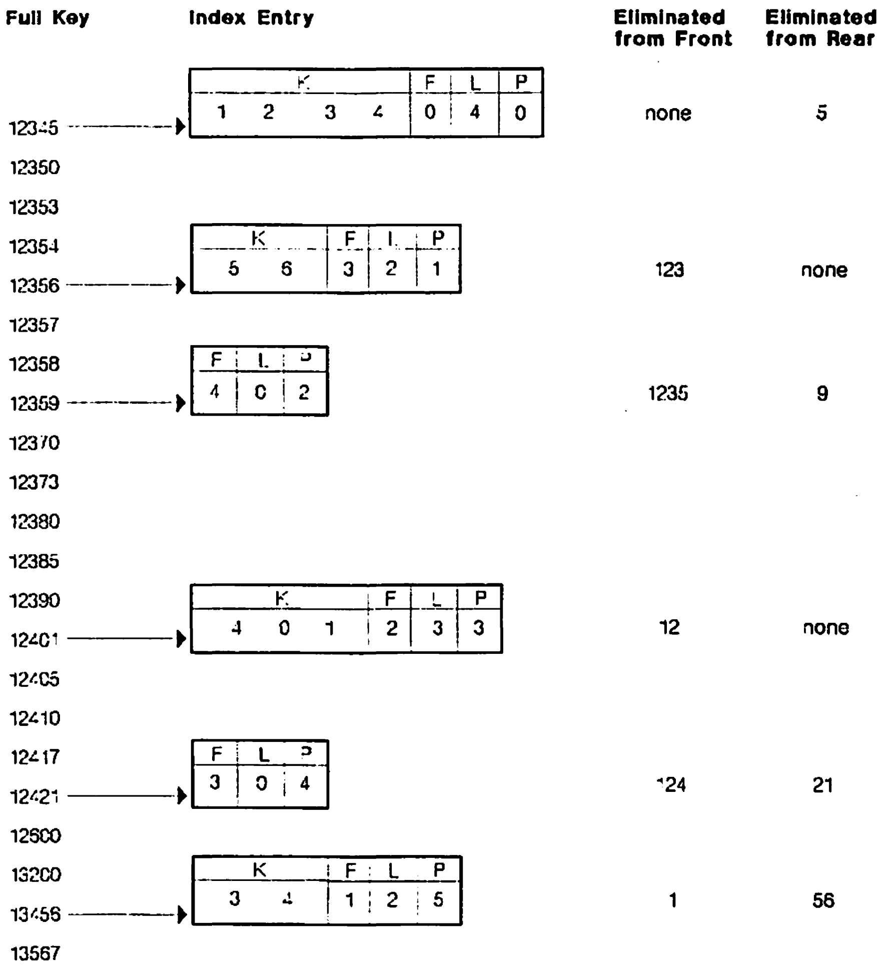
The last index entry in an index level indicates the highest possible key value. The convention for expressing this value is to give none of its characters and indicate that no characters have been eliminated from the front. The last index entry in the last record in the sequence set looks like this:

F	L	P
0	0	X

where x is a binary number from 0 to 255, assuming a 1-byte pointer.

In a search, the two 0's signify the highest possible key value in this way:

- The fact that 0 characters have been eliminated from the front implies that the first character in the key is greater than the first character in the preceding key.
- A length of 0 indicates that no character comparison is required to determine whether the search is successful; that is, when a search encounters the last index entry, a hit has been made.



Legend:
K-Characters left in key after compression
F-Number of characters eliminated from the front
L-Number of characters left in key after compression
P-Vertical pointer

Figure 38. Example of Key Compression

Index Update following a Control Interval Split

When a data set is first loaded, the key sequence of data records and their physical order are the same. However, when data records are inserted, *control interval splits* can occur, causing the data control intervals to have a physical order that differs from the key sequence.

Figure 39 illustrates how the control interval is split and the index is updated when a record with a key of 12 is inserted in the control area shown in Figure 32 on page 189.

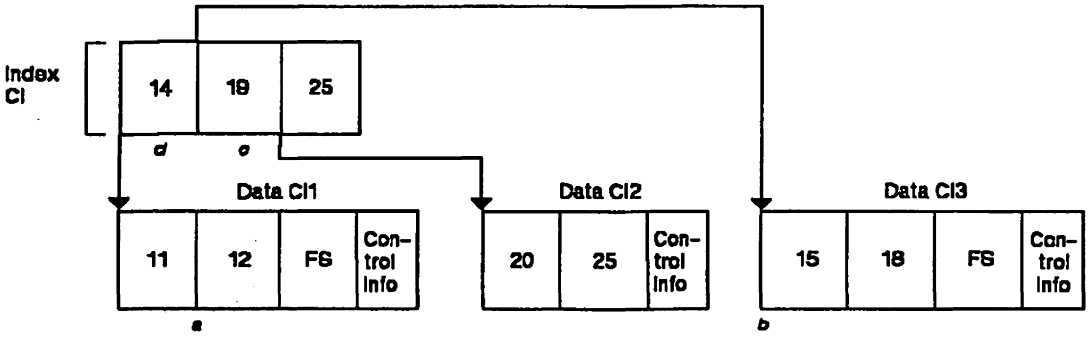


Figure 39. Control Interval Split and Index Update

a. A control interval split occurs in data control interval 1, where a record with the key of 12 must be inserted.

b. Half the records in data control interval 1 are moved by VSAM to the free space control interval (data control interval 3).

c. An index entry is inserted in key sequence to point to data control interval 3, that now contains data records moved from data control interval 1.

d. A new index entry is created for data control interval 1, because after the control interval split, the highest possible key is 14. Because data control interval 3 now contains data, the pointer to this control interval is removed from the free list and associated with the new key entry in the index. Note that key values in the index are in proper ascending sequence, but the data control intervals are no longer in physical sequence.

Index Entries for a Spanned Record

In a key-sequenced data set, there is an index entry for each control interval that contains a segment of a spanned record. All the index entries for a spanned record are grouped together in the same section. They are ordered from right to left according to the sequence of segments (first, second, third, and so on).

Only the last (leftmost) index entry for a spanned record contains the key of the record. The key is compressed according to the rules described above. All the other index entries for the record look like this:

F	L	P
Y	0	X

where y is a binary number equal to the length of the key (y indicates that the entire key has been "eliminated from the front"); L indicates that 0 characters remain; and x identifies the control interval that contains the segment.

End of Product-Sensitive Programming Interface

Appendix E. Calculating Virtual Storage Space for an Alternate Index

When an alternate index is built by BLDINDEX processing, access method services opens the base cluster to sequentially read the data records, sorts the information obtained from the data records, and builds the alternate index records:

1. The base cluster is opened for read-only processing. To prevent other users from updating the base cluster's records during BLDINDEX processing, include the DISP=OLD parameter in the base cluster's DD statement. If INDATASET is specified, access method services dynamically allocates the base cluster with DISP=OLD.

2. The base cluster's data records are read and information is extracted to form the key-pointer pair:

   - When the base cluster is entry-sequenced, the alternate key value and the data record's RBA form the key-pointer pair.

   - When the base cluster is key-sequenced, the alternate key value and the data record's prime key value form the key-pointer pair.

   If the base cluster's data records can span control intervals the alternate key must be in the record's first control interval.

3. The key-pointer pairs are sorted in ascending alternate key order. If your program provides enough virtual storage, access method services performs an internal sort. (The sorting of key-pointer pairs takes place entirely within virtual storage.)

   Use the following process to determine the amount of virtual storage required to sort the records internally:

   a. Sort record length = alternate key length + (prime key length (for a key-sequenced data set) or 4 (for an entry-sequenced data set)).

   b. Record sort area size = either the sort record length times the number of records in the base cluster rounded up to the next integer multiple of 2048 (the next 2K boundary), or a minimum of 32768, whichever is greater.

   c. Sort table size = (record sort area size/sort record length) x 4.

   d. The sum of b + c = required amount of virtual storage for an internal sort. (This amount is in addition to the normal storage requirements for processing an access method services command.)

   If you do not provide enough virtual storage for an internal sort, or if you specify the EXTERNALSORT parameter, access method services defines and uses two sort work files and sorts the key-pointer pairs externally. Access method services uses the sort work files to contain most of the key-pointer pairs while it sorts some of them in virtual storage. An external sort work file is a VSAM entry-sequenced cluster, marked reusable. The minimum amount of virtual storage you need for an external sort is:

   32768 + ((32768/sort record length) x 4)

The amount of space that access method services requests when defining each sort work file is calculated as follows:

a. Sort records per block = 2041/sort record length

b. Primary space allocation in records = (number of records in base cluster/sort records per block) + 10

c. Secondary space allocation in records = (primary space allocation x 0.10) + 10

Both primary and secondary space allocation are requested in records with a fixed-length record size of 2041 bytes; the control interval size is 2048 bytes.

There must be enough space on a single DASD volume to satisfy the primary allocation request; if there is not, the request fails. To correct the problem, specify the volume serial of a device that has sufficient space (see "DD Statements That Describe the Sort Work Files").

4. When the key-pointer pairs are sorted into ascending alternate key order, access method services builds an alternate index record for each key-pointer pair. If the NONUNIQUEKEY attribute is used and more than one key-pointer pair has the same alternate key values, the alternate index record contains the alternate key value, followed by the pointer values in ascending order. If the UNIQUEKEY attribute is used, each alternate key value must be unique.

   When the record is built, it is written into the alternate index as though it is a data record loaded into a key-sequenced cluster. Attributes and values to the load data records specified when the alternate index is defined include:

   RECORDSIZE
   CONTROLINTERVALSIZE
   BUFFERSPACE
   FREESPACE
   WRITECHECK
   SPEED
   RECOVERY
   REPLICATE
   IMBED

5. When all alternate index records are built and loaded into the alternate index, the alternate index and its base cluster are closed. Steps 1 through 4 are repeated for each alternate index that is specified with the OUTFILE and OUTDATASET parameter. When all alternate indexes are built, any defined external sort work files are deleted. Access method services finishes processing and issues messages that indicate the results of the processing.

DD Statements That Describe the Sort Work Files

VSAM data set space available for the sort routine can be identified by specifying two dnames with the WORKFILES parameter and supplying two DD statements that describe the work files to be defined. Each work file DD statement should be coded:

```
//ddname DD  DSNAME=dsname,VOL=SER=volser,
//           UNIT=devtype,DISP=OLD,AMP='AMORG'
```

ddname
As specified in the WORKFILES parameter. If you do not specify the WORKFILES parameter and you intend to provide VSAM data set space VSAM data space for external sort work files, identify the work file DD statements with the names IDCUT1 and IDCUT2.

dsname
A data set name. The scheduler generates a data set name for the work file if none is provided. A data set name must be specified if the user is defined to RACF with a connect attribute of ADSP. The data set name must be a valid group name.

VOL=SER=*volser*
Required. Identifies the volume owned by the STEPCAT, JOBCAT, or master catalog where the work file is cataloged. The work file's space is allocated from the volume's space. You can specify a maximum of five volumes for each work file. For a description of how to calculate the amount of space to be allocated for each sort work file, see "How an Alternate Index Is Built" on page 34. If your BLDINDEX job requires external sort work files, this space must be available on the volume(s) identified by volser or your job will fail.

UNIT=*devtype*
Type of direct access device on which the volume is mounted. You can specify a generic device type (for example, 3380) or a device number (for example 121). You cannot specify SYSDA.

DISP=OLD
Required.

AMP='AMORG'
Required.

If BLDINDEX is used interactively in a TSO environment, these sort work file DD statements must be in the logon procedure.

Appendix F. Using ISAM Programs with VSAM Data Sets

General-Use Programming Interface

This appendix is intended to help you use ISAM programs with VSAM data sets. It contains general-use programming interfaces, which allow you to write programs that use the services of MVS/XA Data Facility Product. Use of ISAM is not recommended. The information in this appendix is shown for compatibility only.

Although the ISAM interface is an efficient way of processing your existing ISAM programs, all new programs that you write should be VSAM programs. ISAM data sets should be migrated to VSAM key-sequenced data sets. Existing programs can use the ISAM/VSAM interface to access those data sets and need not be deleted. You can use the REPRO command with the ENVIRONMENT keyword to handle the VSAM "dummy" records.

VSAM, through its ISAM interface program, enables a debugged program that processes an indexed-sequential data set to process a key-sequenced data set. The key-sequenced data set may have been converted from an indexed-sequential or a sequential data set (or another VSAM data set) or may have been loaded by one of your own programs. The loading program may be coded with VSAM macros or with ISAM macros or PL/I or COBOL statements. That is, you can load records into a newly defined key-sequenced data set with a program that was coded to load records into an indexed sequential data set.

Figure 40 on page 206 shows the relationship between ISAM programs processing VSAM data with the ISAM interface and VSAM programs processing the data.

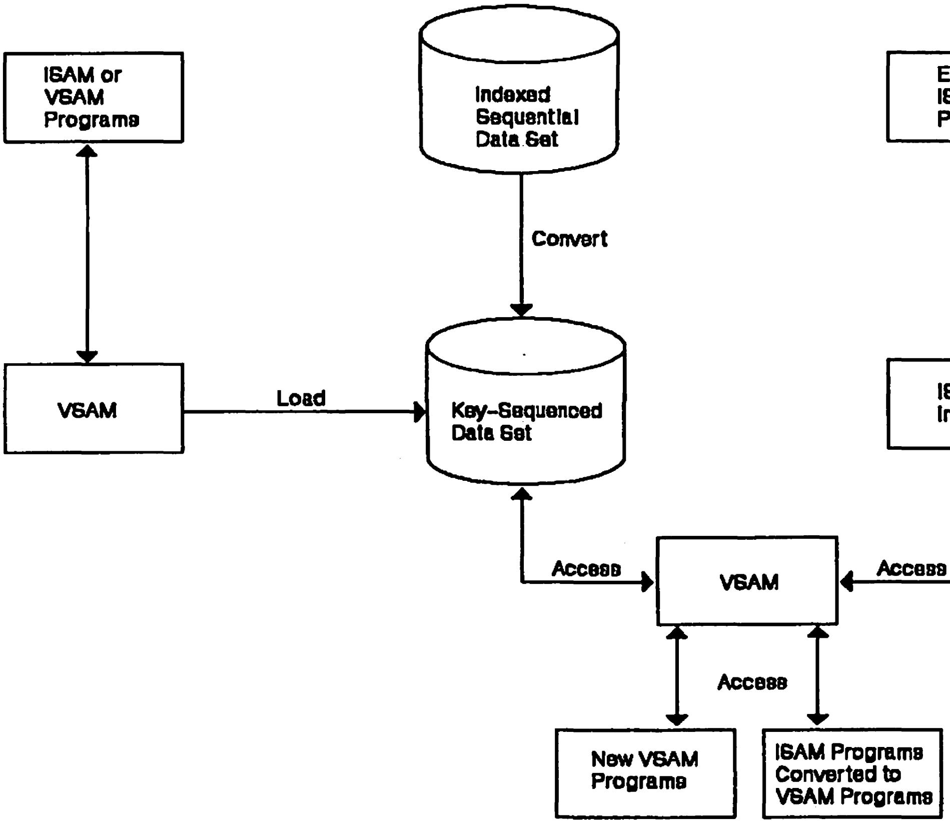


Figure 40. Use of ISAM Processing Programs

There are some minor restrictions on the types of processing an ISAM program may do if it is to be able to process a key-sequenced data set. These restrictions are described in "Restrictions on the Use of the ISAM Interface" on page 217.

Significant performance improvement can be gained by modifying an ISAM program that issues multiple OPEN and CLOSE macros to switch between a QISAM and BISAM DCB. The ISAM program can be modified to open the QISAM and BISAM DCBs at the beginning of the program and to close them when all processing is complete. The performance improvement is proportional to the frequency of OPEN and CLOSE macros in the ISAM program.

How an ISAM Program Can Process a VSAM Data Set

When a processing program that uses ISAM (assembler-language macros, PL/I, or COBOL) issues an OPEN to open a key-sequenced data set, the ISAM interface is given control to:

- Construct control blocks that are required by VSAM
- Load the appropriate ISAM interface routines into virtual storage
- Initialize the ISAM DCB (data control block) to enable the interface to intercept ISAM requests
- Take the DCB exit requested by the processing program

The ISAM interface intercepts each subsequent ISAM request, analyzes it to determine the equivalent keyed VSAM request, defines the keyed VSAM request in a request parameter list, and initiates the request.

The ISAM interface receives return codes and exception codes for logical and physical errors from VSAM, translates them to ISAM codes, and routes them to the processing program or error-analysis (SYNAD) routine by way of the ISAM DCB or DECB. Figure 41 shows QISAM error conditions and the meaning they have when the ISAM interface is being used.

Figure 41 (Page 1 of 2). QISAM Error Conditions

Byte and Offset	QISAM Meaning	Error Detected By	Request Parameter List Error Code	Interface/VSAM Meaning
DCBEXCD1				
Bit 0	Record not found	Interface		Record not found (SETL K for a deleted record)
		VSAM	16	Record not found
		VSAM	24	Record on nonmountable volume
Bit 1	Invalid device address	–	–	Always 0
Bit 2	Space not found	VSAM	28	Data set cannot be extended
		VSAM	40	Virtual storage not available
Bit 3	Invalid request	Interface		Two consecutive SETL requests
		Interface		Invalid SETL (I or ID)
		Interface		Invalid generic key (KEY=0)
		VSAM	4	Request after end-of-data
		VSAM	20	Exclusive use conflict
		VSAM	36	No key range defined for insertion
		VSAM	64	Placeholder not available for concurrent data-set positioning
		VSAM	96	Key change attempted
Bit 4	Uncorrectable input error	VSAM	4	Physical read error (register 15 contains a value of 12) in the data component
		VSAM	8	Physical read error (register 15 contains a value of 12) in the index component
		VSAM	12	Physical read error (register 15 contains a value of 12) in the sequence set of the index
Bit 5	Uncorrectable output error	VSAM	16	Physical write error (register 15 contains a value of 12) in the data component
		VSAM	20	Physical write error (register 15 contains a value of 12) in the index component
		VSAM	24	Physical write error (register 15 contains a value of 12) in the sequence set of the index

Figure 41 (Page 2 of 2). QISAM Error Conditions

Byte and Offset	QISAM Meaning	Error Detected By	Request Parameter List Error Code	Interface/VSAM Meaning
Bit 6	Unreachable block input	VSAM		Logical error not covered by other exception codes
Bit 7	Unreachable block (output)	VSAM		Logical error not covered by other exception codes
DEBEXCD2				
Bit 0	Sequence check	VSAM Interface	12	Sequence check Sequence check (occurs only during resume load)
Bit 1	Duplicate record	VSAM	8	Duplicate record
Bit 2	DCB closed when error routine entered	VSAM		Error in close error routine entered
Bit 3	Overflow record	Interface	—	Always 1
Bit 4	Length of logical record is greater than DCBLRECL (VLR only)	Interface	—	Length of logical record is greater than DCBLRECL (VLR only)
		VSAM	108	Invalid record length
Bits 5-7	Reserved		—	Always 0

Figure 42 shows BISAM error conditions and the meaning they have when the ISAM interface is being used.

If invalid requests occur in BISAM that didn't occur previously and the request parameter list indicates that VSAM isn't able to handle concurrent data-set positioning, the value specified for the STRNO AMP parameter should be increased. If the request parameter list indicates an exclusive-use conflict, reevaluate the share options associated with the data.

Figure 42 (Page 1 of 2). BISAM Error Conditions

Byte and Offset	BISAM Meaning	Error Detected By	Request Parameter List Error Code	Interface/VSAM Meaning
DCBEXC1				
Bit 0	Record not found	VSAM	16	Record not found
		VSAM	24	Record on nonmountable volume
Bit 1	Record length check	VSAM	108	Record length check
Bit 2	Space not found	VSAM	28	Data set cannot be extended
Bit 3	Invalid request	Interface	—	No request parameter list available
		VSAM	20	Exclusive-use conflict
		VSAM	36	No key range defined for insertion
		VSAM	64	Placeholder not available for concurrent data-set positioning

Figure 42 (Page 2 of 2). BISAM Error Conditions

Byte and Offset	BISAM Meaning	Error Detected By	Request Parameter List Error Code	Interface/VSAM Meaning
		VSAM	96	Key change attempted
Bit 4	Uncorrectable I/O	VSAM	–	Physical error (register 15 will contain a value of 12)
Bit 5	Unreachable block	VSAM	–	Logical error not covered by any other exception code
Bit 6	Overflow record	Interface	–	Always 1 for a successful READ request
Bit 7	Duplicate record	VSAM	8	Duplicate record
DECBEXC2				
Bits 0-5	Reserved		–	Always 0
Bit 6	Channel program initiated by an asynchronous routine		–	Always 0
Bit 7	Previous macro was READ KU	Interface	–	Previous macro was READ KU

Figure 43 gives the contents of registers 0 and 1 when a SYNAD routine specified in a DCB gets control.

Figure 43. Register Contents for DCB-Specified ISAM SYNAD Routine

Reg.	BISAM	QISAM
0	Address of the DECB	0, or, for a sequence check, the address of a field containing the higher key involved in the check
1	Address of the DECB	0

You may also specify a SYNAD routine by way of the DD AMP parameter (see "JCL for Processing with the ISAM Interface" later in this chapter). Figure 44 gives the contents of registers 0 and 1 when a SYNAD routine specified by way of AMP gets control.

Figure 44. Register Contents for AMP-Specified ISAM SYNAD Routine

Reg.	BISAM	QISAM
0	Address of the DECB	0, or, for a sequence check, the address of a field containing the higher key involved in the check
1	Address of the DECB	Address of the DCB

If your SYNAD routine issues the SYNADAF macro, registers 0 and 1 are used to communicate. When you issue SYNADAF, register 0 must have the same contents it had when the SYNAD routine got control and register 1 must contain the address of the DCB.

data set: Register 0 contains a completion code, and register 1 contains the address of the SYNADAF message.

The completion codes and the format of a SYNADAF message are given in *Data Administration: Macro Instruction Reference*.

Figure 45 shows abend codes issued by the ISAM interface when there is no other method of communicating the error to the user.

Figure 45. Abend Codes Issued by the ISAM Interface

ABEND Code	Error Detected By	DCB/DECB Set By Module/Routine	Abend Issued By	Error Condition
03B	OPEN	OPEN/OPEN ACB and VALID CHECK	OPEN	Validity check; either (1) access method services and DCB values for LRECL, KEYLE, and RKP do not correspond, (2) DISP=OLD, the DCB was opened for output, and the number of logical records is greater than zero (RELOAD is implied), or (3) OPEN ACB error code 116 was returned for a request to open a VSAM structure.
031	VSAM	SYNAD	SYNAD	SYNAD (ISAM) was not specified and a VSAM physical and logical error occurred.
	VSAM	SCAN/GET and SETL	SYNAD	SYNAD (ISAM) was not specified and an invalid request was found.
	LOAD	LOAD/RESUME	LOAD	SYNAD (ISAM) was not specified and a sequence check occurred.
	LOAD	LOAD	LOAD	SYNAD (ISAM) was not specified and the RDW (record descriptor word) was greater than LRECL.
039	VSAM	SCAN/EODAD	SCAN	End-of-data was found, but there was no EODAD exit.
001	VSAM	SYNAD		I/O error detected.
	BISAM	SYNAD	BISAM	I/O error detected during check.
	BISAM	BISAM	BISAM	Invalid request.

If a SYNAD routine specified by way of AMP issues the SYNADAF macro, the operand ACSMETH may specify either QISAM or BISAM, regardless of which of the two is used by your processing program.

A dummy DEB is built by the ISAM interface to support:

- References by the ISAM processing program
- Checkpoint/restart
- Abend

- Checkpoint/restart
- Abend

Figure 46 shows the DEB fields that are supported by the ISAM interface. Except as noted, field meanings are the same as in ISAM.

Figure 46. DEB Fields Supported by ISAM Interface

DEB Section	Bytes	Fields Supported
PREFIX	16	LNGTH
BASIC	32	TCBAD, OPATB, DEBAD, OFLGS (DISP ONLY), FLGS1 (ISAM-interface bit), AMLNG (104), NMEXT(2), PRIOR, PROTG, DEBID, DCBAD, EXSCL (0-DUMMY DEB), APPAD
ISAM Device	16	EXPTR, FPEAD
Direct Access	16	UCBAD (VSAM UCB)
Access Method	24	WKPT5 (ISAM-interface control block pointer), FREED (pointer to IDAIIFBF)

Converting an Indexed-Sequential Data Set

Access method services is used to convert an indexed-sequential data set to a key-sequenced data set. Assuming that a master and/or user catalog has been defined, define a key-sequenced data set with the attributes and performance options you want. Then use the access method services REPRO command to convert the indexed-sequential records and load them into the key-sequenced data set. VSAM builds the index for the key-sequenced data set as it loads the data set.

Each volume of a multivolume component must be on the same type of device; the data component and the index component, however, may be on volumes of devices of different types.

When you define the key-sequenced data set into which the indexed-sequential data set is to be copied, you must specify the attributes of the VSAM data set for variable- and fixed-length records.

For variable-length records:

- VSAM record length equals ISAM DCBLRECL-4.
- VSAM key length equals ISAM DCBKEYLE.
- VSAM key position equals ISAM DCBRKP-4.

For fixed-length records:

- VSAM record length (average and maximum must be the same) equals ISAM DCBLRECL (+ DCBKEYLE, if ISAM DCBRKP equals 0 and records are unblocked).

- VSAM key length equals ISAM DCBKEYLE.

- VSAM key position equals ISAM DCBRKP.

The level of sharing allowed when the key-sequenced data set is defined should be considered. If the ISAM program opens multiple DCBs pointing to different

DD statements, a share-options value of 1, which is the default, allows only the first DD statement to be opened. See "Sharing" for a description of the share-options values.

JCL for Converting from ISAM to VSAM

JCL is used to identify data sets and volumes for allocation. Data sets can also be allocated dynamically. For a description of dynamic allocation, see *JCL* and *System Modifications*.

If JCL is used to describe an indexed-sequential data set to be converted to VSAM using the access method services REPRO command, include DCB=DSORG=IS. Use a STEPCAT or JOBCAT DD statement as described in the chapter Appendix B, "Job Control Language" on page 149 to make user catalogs available; you may also use dynamic allocation.

With ISAM, deleted records are flagged as deleted, but are not actually removed from the data set. To avoid reading VSAM records that are flagged as deleted (X'FF'), code DCB=OPTCD=L. If your program depends upon a record's only being flagged and not actually removed, you may want to keep these flagged records when you convert and continue to have your programs process these records. The access method services REPRO command has a parameter (ENVIRONMENT) that causes VSAM to keep the flagged records when you convert.

JCL for Processing with the ISAM Interface

To execute your ISAM processing program to process a key-sequenced data set, replace the ISAM DD card with a VSAM DD card using the DDNAME that was used for ISAM. The VSAM DD card names the key-sequenced data set and gives any necessary VSAM parameters (by way of AMP). Specify DISP=MOD for resume loading and DISP=OLD or SHR for all other processing. You don't have to specify anything about the ISAM interface itself. The interface is automatically brought into action when your processing program opens a DCB whose associated DD statement describes a key-sequenced data set (instead of an indexed-sequential data set). If you have defined your VSAM data set in a user catalog, specify the user catalog in a JOBCAT or STEPCAT DD statement.

The DCB parameter in the DD statement that identifies a VSAM data set is invalid and must be removed. If the DCB parameter is not removed, unpredictable results can occur. Certain DCB-type information may be specified in the AMP parameter, which is described later in this chapter.

Figure 47 on page 213 shows the DCB fields supported by the ISAM interface.

Field Name	Meaning
BFALN	Same as in ISAM; defaults to a doubleword
BLKSI	Set equal to LRECL if not specified
BUFCB	Same as in ISAM
BUFL	The greater value of AMDLRECL or DCBLRECL if not specified
BUFNO	For QISAM, one; for BISAM, the value of STRNO if not specified
DDNAM	Same as in ISAM
DEBAD	During the DCB exit, contains the address of the OPEN work area; after the DCB exit, contains the address of the dummy DEB built by the ISAM interface
DEVT	Set from the VSAM UCB TYPE
DSORG	Same as in ISAM
EODAD	Same as in ISAM
ESETL	Address of the ISAM interface ESETL routine
EXCD1	See the QISAM exception codes
EXCD2	See the QISAM exception codes
EXLST	Same as in ISAM (except that VSAM does not support the JFCBE exit)
FREED	Address of the ISAM-interface dynamic buffering routine (IDAIIFBF)
GET/PUT	For QISAM LOAD, the address of the ISAM-interface PUT routine; for QISAM SCAN, 0, the address of the ISAM-interface GET routine; 4, the address of the ISAM-interface PUTX routine; and 8, the address of the ISAM-interface RELSE routine
KEYLE	Same as in ISAM
LRAN	Address of the ISAM-interface READ K/WRITE K routine
LRECL	Set to the maximum record size specified in the access method services DEFINE command if not specified (adjusted for variable-length, fixed, unblocked, and RKP=0 records)

Figure 47 (Part 1 of 2). DCB Fields Supported by ISAM Interface

Field Name	Meaning
LWKN	Address of the ISAM-interface WRITE KN routine
MACRF	Same as in ISAM
NCP	For BISAM, defaults to one
NCRHI	Set to a value of 8 before DCB exit
OFLGS	Same as in ISAM
OPTCD	Bit 0 (W), same as in ISAM; bit 3 (I), dummy records are not to be written in the VSAM data set; bit 6 (L), VSAM-deleted records (XX'FF') are not read; dummy records are to be treated as in ISAM; all other options ignored
RECFM	Same as in ISAM; default to unblocked, variable-length records
RKP	Same as in ISAM
RORG1	Set to a value of 0 after DCB exit
RORG2	Set to a value of XX'7FFFF' after DCB exit
RORG3	Set to a value of 0 after DCB exit
SETL	For BISAM, address of the ISAM-interface CHECK routine; for QISAM, address of the ISAM-interface SETL routine
ST	Bit 1 (key-sequence check), same as in ISAM; bit 2 (loading has completed), same as in ISAM
SYNAD	Same as in ISAM
TIOT	Same as in ISAM
WKPT1	For QISAM SCAN, WKPT1 +112=address of the W1CBF field pointing to the current buffer
WKPT5	Address of the ISAM-interface control block (IICB)
WKPT6	For QISAM LOAD, address of the dummy DCB work area vector pointers; the only field supported is ISLVPTRS+4=pointer to KEYSAVE

Figure 47 (Part 2 of 2). DCB Fields Supported by ISAM Interface

AMP Parameter Specification

When an ISAM processing program is run with the ISAM interface, the AMP parameter enables you to specify:

- That a VSAM data set is to be processed (AMORG)
- The need for extra index buffers for simulating the residency of the highest level(s) of an index in virtual storage (BUFNI)
- The need for additional data buffers to improve sequential performance (BUFND)
- Whether to remove records flagged (OPTCD)

- What record format (RECFM) is used by the processing program
- The number of concurrent BISAM and QISAM (basic and queued indexed-sequential access methods) requests that the processing program may issue (STRNO)
- The name of an ISAM exit routine to analyze physical and logical errors (SYNAD)

The AMP parameter has some subparameters that are peculiar to the ISAM interface. The other subparameters of AMP (BUFSP, CROPS, and TRACE), which can also be used with the interface, are described in Appendix B, "Job Control Language" on page 149. The format of the AMP parameter (with the subparameters discussed here) is:

//...	**DD**	**AMP=(['AMORG']** **[,'BUFND=***number***']** **[,'BUFNI=***number***']** **[,'OPTCD={I|L|IL}']** **[,'RECFM={F|FB|V|VB}']** **[,'STRNO=***number***']** **[,'SYNAD=***modulename***'])**

where:

AMORG
specifies that a VSAM data set is to be processed. When you specify unit and volume information for a DCB (through the ISAM interface program) or when you specify DUMMY in the DD statement, you must specify AMORG. Under these conditions, the system doesn't have to search a catalog to find out which volume(s) are required, and therefore doesn't know that the DD statement defines a VSAM data set. You never have to specify unit and volume information unless you want to mount some, but not all, of the data set's volumes, or if you want to defer volume mounting.

BUFND=*number*
specifies the number of I/O buffers VSAM is to use for data records. The minimum number you may specify is 1 plus the number specified for STRNO (if you omit STRNO, BUFND must be at least 2, because the default for STRNO is 1).

BUFNI=*number*
specifies the number of I/O buffers VSAM is to use for index records. If you don't specify BUFNI, VSAM uses as many index buffers as the number specified for STRNO (1 if you don't specify STRNO). You may specify for BUFNI a number 1 greater than STRNO (2 if you don't specify STRNO) to simulate having the highest level of an ISAM index resident. If you specify for BUFNI a number 2 or more greater than STRNO, you simulate having intermediate levels of the index resident.

OPTCD={I|L|IL}
specifies how records flagged for deletion are to be treated. The values that can be specified are:

L

specifies that a record marked for deletion by your processing program is to be kept in the data set. Although this parameter has the same meaning and restrictions for the ISAM interface as it has for ISAM, it may have to be specified in the AMP parameter when it wasn't previously needed in the ISAM job control language. It is required when OPTCD=L is not specified in the DCB in the processing program because OPTCD is not merged into the DSCB when the ISAM interface is used.

I

specifies that, when coded along with OPTCD=L in the DCB, records marked for deletion by your processing program are not written into the data set by the ISAM interface. If OPTCD=I is specified in the AMP parameter, but OPTCD=L isn't specified in the processing program's DCB, records flagged for deletion are treated as any other records: that is, AMP='OPTCD=I', without L anywhere specified, has no effect.

IL

specifies that, if your processing program writes a record marked for deletion, the ISAM interface is not to put the record into the data set. (It issues a VSAM ERASE to delete the old record if your processing program had previously read the record for update.) The result of this parameter is the same as when AMP='OPTCD=I' is coded with OPTCD=L in the DCB in the processing program.

RECFM={F|FB|V|VB}

specifies the ISAM record format that your processing program is coded for. Although this parameter has the same meaning and restrictions for the ISAM interface as it has for ISAM, it may have to be specified in the AMP parameter when it wasn't previously required in the ISAM job control language. RECFM is required when it is not specified in the DCB in the processing program because RECFM is not merged into the DSCB when the ISAM interface is used. All VSAM requests are for unblocked records. If your program issues a request for blocked records, the ISAM interface sets the overflow-record indicator for each record to indicate that each is being passed to your program unblocked. If RECFM isn't specified in the AMP parameter or in the processing program's DCB, V is the default.

STRNO=*number*

specifies the number of request parameter lists the processing program can use concurrently. Neither VSAM nor the ISAM interface can anticipate the number, so you should indicate it in the STRNO parameter. Specify a number at least equal to the number of BISAM and QISAM requests that your program can issue concurrently. (If you have subtasks. add together the number of such requests for each subtask, plus an additional one for each subtask that sequentially processes the same data set.) In a create step, STRNO cannot be greater than 1. The ISAM interface uses a request parameter list to describe a request that your program issues. The interface uses the same request parameter list over and over:

- With BISAM, a READ for update uses a request parameter list until a WRITE or FREEDBUF is issued (at which time the interface issues an ENDREQ for the request parameter list).
- With QISAM, a request parameter list is used until an ESETL is issued (at which time the interface issues ENDREQ).

If the processing program issues an ISAM request when no more request parameter lists are available, the ISAM interface returns an ISAM code that indicates an invalid request. If you're running subtasks, it's possible to reissue the invalid request and have it complete successfully when another subtask frees a request parameter list.

SYNAD=*modulename*
specifies the name of a routine to which the ISAM interface loads and exits if a physical or logical error occurs when you are gaining access to the key-sequenced data set. If your processing program already indicates a SYNAD routine, the routine specified in the AMP SYNAD parameter replaces it.

The SYNAD routine must not issue VSAM macros or check for VSAM return codes. The ISAM interface translates all VSAM codes to appropriate ISAM codes.

You need not modify or replace a SYNAD routine that issues only a CLOSE, ABEND, SYNADAF, or SYNADRLS macro or that merely examines DCB or DECB exception codes.

Restrictions on the Use of the ISAM Interface

Some restrictions were indicated earlier in this chapter that may require you to modify an ISAM processing program to process a key-sequenced data set. All operating system and VSAM restrictions apply to the use of the ISAM interface; for example:

- VSAM doesn't allow the OPEN TYPE=J macro: If your program issues it, remove it or replace it with the OPEN macro.
- If your processing program was coded on the assumption that the indexed-sequential data set it was processing was a temporary data set, you may need to modify the program: A VSAM data set cannot be temporary.
- If a GET command is issued to an empty data set, the resulting messages will indicate "no record found (NRF)" rather than "end of data (EOD)," as it would appear in the noninterface QISAM environment.

Additional restrictions are:

- A program must run successfully under ISAM using standard ISAM interfaces; the interface doesn't check for parameters that are invalid for ISAM.
- If your DCB exit list contains an entry for a JFCBE exit routine, remove it. The interface doesn't support the use of a JFCBE exit routine. If the DCB exit list contains an entry for a DCB open exit routine, that exit is taken.
- If your ISAM program creates dummy records with a maximum key to avoid overflow, remove that code for VSAM.
- If your program counts overflow records to determine reorganization needs, its results will be meaningless with VSAM data sets.
- The work area into which data records are read must not be shorter than a record. If your processing program is designed to read a portion of a record into a work area, you must change the design. The interface takes the record length indicated in the DCB to be the actual length of the data record. The record length in a BISAM DECB is ignored, except when you are replacing a variable-length record with the WRITE macro.
- You may share data among subtasks that specify the same DD statement in their DCB(s), and VSAM ensures data integrity. But, if you share data

among subtasks that specify different DD statements for the data, you are responsible for data integrity. The ISAM interface doesn't ensure DCB integrity when two or more DCBs are opened for a data set. All of the fields in a DCB cannot be depended on to contain valid information.

- When a data set is shared by several jobs (DISP=SHR), you must use the ENQ and DEQ macros to ensure exclusive control of the data set. Exclusive control is necessary to ensure data integrity when your program adds or updates records in the data set. You can share the data set with other users (that is, relinquish exclusive control) when reading records.

- If your processing program issues the SETL I or SETL ID instruction, you must modify the instruction to some other form of the SETL or remove it. The ISAM interface cannot translate a request that depends on a specific block or device address.

- Although asynchronous processing may be specified in an ISAM processing program, all ISAM requests are handled synchronously by the ISAM interface; WAIT and CHECK requests are always satisfied immediately. The ISAM CHECK macro doesn't result in a VSAM CHECK macro's being issued but merely causes exception codes in the DECB (data event control block) to be tested.

- For processing programs that use locate processing, the ISAM interface constructs buffers to simulate locate processing.

- For blocked-record processing, the ISAM interface simulates unblocked-record processing by setting the overflow-record indicator for each record. (In ISAM, an overflow record is never blocked with other records.) Programs that examine ISAM internal data areas (for example, block descriptor words (BDW) or the MBBCCHHR address of the next overflow record) must be modified to use only standard ISAM interfaces. The ISAM RELSE instruction causes no action to take place.

- If your ISAM SYNAD routine examines information that cannot be supported by the ISAM interface (for example, the IOB), specify a replacement ISAM SYNAD routine in the AMP parameter of the VSAM DD statement.

- Your ISAM program (on TSO) cannot dynamically allocate a VSAM data set (use LOGON PROC).

- CATALOG/DADSM macros in the ISAM processing program must be replaced with access method services commands.

- The ISAM interface uses the same RPL over and over, thus, for BISAM, a READ for update uses up an RPL until a WRITE or FREEDBUF is issued (at which time the interface issues an ENDREQ for the RPL). (When using ISAM you may merely issue another READ if you don't want to update a record after issuing a BISAM READ for update.)

- ISAM programs will run, with sequential processing, if the key length is defined as smaller than it actually is. This is not permitted with the ISAM interface.

- VSAM path processing is not supported by the ISAM interface.

- The ISAM interface does not support RELOAD processing. RELOAD processing is implied when an attempt is made to open a VSAM data set for output, specifying DISP=OLD, and, in addition, the number of logical records in the data set is greater than zero.

Example: Converting a Data Set

In this example, the indexed-sequential data set to be converted (ISAMDATA) is cataloged either in the system catalog or in a VSAM catalog. A key-sequenced data set, VSAMDATA, has previously been defined in user catalog USERCTLG. Because both the indexed-sequential and key-sequenced data set are cataloged, unit and volume information need not be specified.

ISAMDATA contains records flagged for deletion; these records are to be kept in the VSAM data set.

```
//CONVERT  JOB  ...
//JOBCAT   DD   DISP=SHR,DSNAME=USERCTLG
//STEP     EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//ISAM     DD   DISP=OLD,DSNAME=ISAMDATA,DCB=DSORG=IS
//VSAM     DD   DISP=OLD,DSNAME=VSAMDATA
//SYSIN    DD   *
      REPRO -
           INFILE(ISAM  ENVIRONMENT(DUMMY)) -
           OUTFILE(VSAM)
/*
```

To drop records flagged for deletion in the indexed-sequential data set, omit ENVIRONMENT(DUMMY).

Example: Issuing a SYNADAF Macro

The following example illustrates how a SYNAD routine specified by way of AMP may issue a SYNADAF macro without preliminaries. Registers 0 and 1 already contain what SYNADAF expects to find.

```
AMPSYN   CSECT

         USING      *,15              Register 15 contains the entry
                                      address to AMPSYN.

         SYNADAF    ACSMETH=QISAM     Either QISAM or BISAM may be
                                      specified.

         STM        14,12,12(13)

         BALR       7,0               Load address of next instruction
                                      into register 7 for base register.
         USING      *,7

         L          15,132(1)         The address of the DCB is stored
                                      132 bytes into the SYNADAF message.

         L          14,128(1)         The address of the DECB is stored
                                      128 bytes into the SYNADAF message.

         TM         42(15),X'40'      Does the DCB indicate QISAM scan?

         BO         QISAM             Yes.

         TM         43(15),X'40'      Does the DCB indicate QISAM load?

         BO         QISAM             Yes.
```

```
BISAM    TM         24(14),X'10'       Does the DECB indicate an invalid
                                      BISAM request?

         BO         INVBISAM           Yes.

.                              The routine might print the SYNADAF
.                              message or issue ABEND.
.
QISAM    TM         80(15),X'10'       Does the DCB indicate an invalid
                                      QISAM request?

         BO         INVQISAM           Yes.

.                              The routine might print the SYNADAF
.                              message or issue ABEND.
.
INVBISAM EQU        *

INVQISAM EQU        *

         LM         14,12,12(13)

         DROP       7

         USING      AMPSYN,15

         SYNADRLS

         BR         14

         END        AMPSYN
```

When the processing program closes the data set, the interface issues VSAM PUT macros for ISAM PUT locate requests (in load mode), deletes the interface routines from virtual storage, frees virtual-storage space that was obtained for the interface, and gives control to VSAM.

End of General-Use Programming Interface

Glossary of Terms and Abbreviations

The following terms are defined as they are used in this book. If you do not find the term you are looking for, see the index or the *Dictionary of Computing*, SC20-1699.

A

access method services. A multifunction service program that is used to define VSAM data sets and allocate space for them, convert indexed-sequential data sets to key-sequenced data sets, modify data set attributes in the catalog, reorganize data sets, facilitate data portability between operating systems, create backup copies of data sets, help make inaccessible data sets accessible, list the records of data sets and catalogs, define and build alternate indexes, and convert OS CVOLs and VSAM catalogs to integrated catalog facility catalogs.

acquire. To allocate space on a staging drive and to stage data from an MSS cartridge to the staging drive.

addressed-direct access. The retrieval or storage of a data record identified by its RBA, independent of the record's location relative to the previously retrieved or stored record. (*See also* keyed-direct access, addressed- sequential access, and keyed-sequential access.)

addressed-sequential address. The retrieval or storage of a data record in its entry sequence relative to the previously retrieved or stored record. (*See also* keyed-sequential access, addressed-direct access, and keyed-direct access.)

ADSP. (*See* automatic data set protection.)

AIX. (*See* alternate index.)

alternate index. A collection of index entries organized by the alternate keys of its associated base data records. It provides an alternate means of locating records in the data component of a cluster on which the alternate index is based.

alternate index cluster. The data and index components of an alternate index.

alternate index entry. A catalog entry that contains information about an alternate index. An alternate-index entry points to a data entry and an index entry to describe the alternate index's components, and to a cluster entry to identify the alternate index's base cluster.

alternate index record. A collection of items used to sequence and locate one or more data records in a base cluster. Each alternate-index record contains an alternate-key value and one or more pointers. When the alternate index supports a key-sequenced data set, each data record's prime key value is the pointer. When the alternate index supports an entry-sequenced data set, the data record's RBA value is the pointer.

alternate index upgrade. The process of reflecting changes made to a base cluster in its associated alternate indexes.

alternate key. One or more consecutive characters taken from a data record and used to build an alternate index or to locate one or more base data records via an alternate index. (*See also* generic key, key, and key field.)

APF. (*See* authorized program facility.)

application. As used in this publication, the use to which an access method is put or the end result that it serves; contrasted to the internal operation of the access method.

authorized program facility. A facility that permits the identification of programs that are authorized to use restricted functions.

automatic data set protection. A user attribute that causes all permanent DASD data sets created by the user to be automatically defined to RACF.

B

base cluster. A key-sequenced or entry-sequenced data set over which one or more alternate indexes are built.

base RBA. The RBA stored in the header of an index record that is used to calculate the RBAs of data or index control intervals governed by the index record.

BIND. (1) An attribute of a data set that keeps the data set on one or more MSS staging drives until the data set is released by the user regardless of the length of time or the demands for space. (2) An attribute of a mass storage volume that reserves an entire staging pack for the mass storage volume whenever the volume is mounted.

C

catalog. (*See* master catalog and user catalog.)

CBIC. Control blocks in common, a facility that allows a user to open a VSAM data set so the VSAM control blocks are placed in the common service area (CSA) of the MVS operating system. This provides the capability for multiple memory accesses to a single VSAM control structure for the same VSAM data set.

chained RPL. (*See* RPL string.)

CI. (*See* control interval.)

CIDF. (*See* control interval definition field.)

CKDS. In the Programmed Cryptographic Facility, cryptographic key data set.

cluster. A named structure consisting of a group of related components (for example, a data component with its index component). A cluster may consist of a single component. (*See also* base cluster and alternate index cluster.)

collating sequence. An ordering assigned to a set of items, such that any two sets in that assigned order can be collated.

component. A named, cataloged collection of stored records. A component, the lowest member of the hierarchy of data structures that can be cataloged, contains no named subsets.

control area. A group of control intervals used as a unit for formatting a data set before adding records to it. Also, in a key-sequenced data set, the set of control intervals pointed to by a sequence-set index record; used by VSAM for distributing free space and for placing a sequence-set index record adjacent to its data.

control area split. The movement of the contents of some of the control intervals in a control area to a newly created control area, to facilitate the insertion or lengthening of a data record when there are no remaining free control intervals in the original control area.

control interval. A fixed-length area of auxiliary storage space in which VSAM stores records. It is the unit of information transmitted to or from auxiliary storage by VSAM.

control interval access. The retrieval or storage of the contents of a control interval.

control interval definition field. In VSAM, the 4-byte control information field at the end of a control interval that gives the displacement from the beginning of the control interval to free space and the length of the free space. If the length is 0, the displacement is to the beginning of the control information.

control interval split. The movement of some of the stored records in a control interval to a free control interval, to facilitate the insertion or lengthening of a record that won't fit in the original control interval.

control volume. A volume that contains one or more indexes of the catalog.

cross memory. A synchronous method of communication between address spaces.

CVOL. (*See* control volume.)

cylinder fault. A condition that occurs when the operating system requires data that has not been staged. The cylinder fault causes a cylinder of data to be staged.

D

DASD. (*See* direct access storage device.)

data integrity. Preservation of data or programs for their intended purpose. As used in this publication, the safety of data from inadvertent destruction or alteration.

data record. A collection of items of information from the standpoint of its use in an application, as a user supplies it to VSAM for storage.

data security. Prevention of access to or use of data or programs without authorization. As used in this publication, the safety of data from unauthorized use, theft, or purposeful destruction.

data set. The major unit of data storage and retrieval in the operating system, consisting of data in a prescribed arrangement and described by control information to which the system has access. As used in this publication, a collection of fixed- or variable-length records in auxiliary storage, arranged by VSAM in key sequence or in entry sequence. (*See also* key-sequenced data set and entry-sequenced data set.)

data space. A storage area defined in the volume table of contents of a direct access volume for the exclusive use of VSAM to store data sets, indexes, and catalogs.

DD statement. Data definition statement.

DES. The United States National Bureau of Standards data encryption standard.

destage. To move data from a staging drive to a mass storage volume.

direct access. The retrieval or storage of data by a reference to its location in a data set rather than relative to the previously retrieved or stored data. (*See also* addressed-direct access and keyed-direct access.)

direct access storage device. A device in which the access time is effectively independent of the location of the data.

E

EBDIC. Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

ECB. (*See* event control block.)

entry sequence. The order in which data records are physically arranged (according to ascending RBA) in auxiliary storage, without respect to their contents. (Contrast with key sequence.)

entry-sequenced data set. A data set whose records are loaded without respect to their contents, and whose RBAs cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

EOD. End of data.

EODAD. The end of data exit routine.

EOKR. End-of-key range.

EOV. End of volume.

event control block. A control block used to represent the status of an event.

exception. An abnormal condition such as an I/O error encountered in processing a data set or a file.

EXCEPTIONEXIT. An exit routine invoked by an exception.

F

field. In a record or a control block, a specified area used for a particular category of data or control information.

free control interval pointer list. In a sequence-set index record, a vertical pointer that gives the location of a free control interval in the control area governed by the record.

free space. Space reserved within the control intervals of a key-sequenced data set for inserting new records into the data set in key sequence; also, whole control intervals reserved in a control area for the same purpose.

front compression. The elimination, from the front of a key, of characters that are the same as the characters in the front of the preceding key.

G

GENDSP. An option of LOCATE to obtain the control interval number of the catalog record of each object.

generation data group. A collection of data sets that are kept in chronological order; each data set is called a generation data set.

generic key. A high-order portion of a key, containing characters that identify those records that are significant for a certain application. For example, it might be desirable to retrieve all records whose keys begin with the generic key AB, regardless of the full key values.

global shared resources. An option for sharing I/O buffers, I/O-related control blocks, and channel programs among VSAM data sets in a resource pool that serves all address spaces in the system.

GSR. (*See* global shared resources.)

H

header, index record. In an index record, the 24-byte field at the beginning of the record that contains control information about the record.

header entry. In a parameter list of GENCB, MODCB, SHOWCB, or TESTCB, the entry that identifies the type of request and control block and gives other general information about the request.

horizontal pointer. In the header of an index record, the RBA of the index record in the same level as this one that contains keys next in ascending sequence after the keys in this one.

I

index. As used in this publication, an ordered collection of pairs, each consisting of a key and a pointer, used by VSAM to sequence and locate the records of a key-sequenced data set.

index level. A set of index records that order and give the location of all the control intervals in the next lower level or in the data set that it controls.

index record. A collection of index entries that are retrieved and stored as a group. (Contrast to data record.)

index record header. In an index record, the 24-byte field at the beginning of the record that contains control information about the record.

index replication. The use of an entire track of direct access storage to contain as many copies of a single index record as possible; reduces rotational delay.

index set. The set of index levels above the sequence set. The index set and the sequence set together comprise the index.

integrated catalog facility. The name of the catalog associated with the Data Facility Product program product.

ISAM. indexed sequential access method

ISAM interface. A set of routines that allow a processing program coded to use ISAM (indexed sequential access method) to gain access to a key-sequenced data set.

J

JCL. (*See* job control language.)

job catalog. A catalog made available for a job by means of the JOBCAT DD statement.

job control language. A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

job step catalog. A catalog made available for a job by means of the STEPCAT DD statement.

K

key. One or more characters within an item of data that are used to identify it or control its use. As used in this publication, one or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records. (*See also* key field and generic key.)

key compression. The elimination of characters from the front and the back of a key that VSAM does not need to distinguish the key from the preceding or following key in the index record; reduces storage space for an index.

key field. A field located in the same position in each record of a data set, whose contents are used for the key of a record.

key sequence. The collating sequence of data records, determined by the value of the key field in each of the data records. May be the same as, or different from, the entry sequence of the records.

key-sequenced data set. A VSAM file (data set) whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in key sequence by means of distributed free space. Relative byte addresses of records can change, because of control interval or control area splits.

keyed-direct access. The retrieval or storage of a data record by use of either an index that relates the record's key to its relative location in the data set or a relative record number, independent of the record's location relative to the previously retrieved or stored record. (*See also* addressed-direct access, keyed-sequential access, and addressed-sequential access.)

keyed-sequential access. The retrieval or storage of a data record in its key or relative record sequence relative to the previously retrieved or stored record, as defined by the sequence set of an index. (*See also* addressed-sequential access, keyed-direct access, and addressed-direct access.)

L

LDS. (*See* linear data set.)

LERAD. The logical error exit routine

level number. For the index of a key-sequenced data set, a binary number in the header of an index record that indicates the index level to which the record belongs.

linear data set. A named linear string of data, stored in such a way that it can be retrieved or updated in 4096 byte units. An LDS object is essentially a VSAM entry-sequenced data set that is processed as a control interval. However, unlike a control interval, an LDS contains data only; that is, it contains no record definition fields (RDFs) or control interval definition fields (CIDFs).

local shared resources. An option for sharing I/O buffers, I/O-related control blocks, and channel programs among VSAM data sets in a resource pool that serves one partition or address space.

LSR. (*See* local shared resources.)

M

Mass Storage System (3850). The name for the entire storage system, consisting of the Mass Storage Facility and all devices that are defined to the Mass Storage Control.

master catalog. A catalog that contains extensive data set and volume information that VSAM requires to locate data sets, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and to accumulate usage statistics for data sets.

MSS. (*See* Mass Storage System.)

multiple virtual storage. Another name for OS/VS2, release 2.

MVS. (*See* multiple virtual storage.)

O

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

P

page space. A system data set that contains pages of virtual storage. The pages are stored into and retrieved from the page space by the auxiliary storage manager.

password. A unique string of characters stored in a catalog that a program, a computer operator, or a terminal user must supply to meet security requirements before a program gains access to a data set.

path. A named, logical entity composed of one or more clusters (an alternate index and its base cluster, for example).

physical record. A physical unit or recording on a medium. For example, the physical unit between address markers on a disk.

pointer. An address or other indication of location. For example, an RBA is a pointer that gives the relative location of a data record or a control interval in the data set to which it belongs.

portability. The ability to use VSAM data sets with different operating systems. Volumes whose data sets are cataloged in a user catalog can be demounted from storage devices of one system, moved to another system, and mounted on storage devices of that system. Individual data sets can be transported between operating systems using access method services.

prestage. To move data from an MSS cartridge to a staging drive before the data is needed by the processing program.

prime index. The index component of a key-sequenced data set that has one or more alternate indexes. (*See also* index and alternate index.)

prime key. (*See* key.)

Q

QSAM. (*See* queued sequential access method.)

queued sequential access method. An extended version of the basic sequential access method (BSAM). When this method is used, a queue is formed of input data blocks that are awaiting processing or output data blocks that have been processed and are awaiting transfer to auxiliary storage or to an output device.

R

RACF. Resource Access Control Facility.

random access. (*See* direct access.)

RBA. Relative byte address. The displacement (expressed as a fullword binary integer) of a data record or a control interval from the beginning of the data set to which it belongs; independent of the manner in which the data set is stored.

RDF. (*See* record definition field.)

rear compression. The elimination, from a key, of characters to the right of the first character that is unequal to the corresponding character in the following key.

record. (*See* index record, data record.)

record definition field. A field stored as part of a stored record segment; it contains the control information required to manage stored record segments within a control interval.

recoverable catalog. A catalog defined with the recoverable attribute. Duplicate catalog entries are put into CRAs that can be used to recover data in the event of catalog failure. (*See also* CRA.)

relative byte address. (*See* RBA.)

relative record data set. A data set whose records are loaded into fixed-length slots.

relative record number. A number that identifies not only the slot, or data space, in a relative record data set but also the record occupying the slot. Used as the key for keyed access to a relative record data set.

replication. (*See* index replication.)

resource pool, VSAM. (*See* VSAM resource pool.)

reusable data set. A VSAM data set that can be reused as a work file, regardless of its old contents. Must not be a base cluster.

RPL string. A set of chained RPLs (the set may contain one or more RPLs) used to gain access to a VSAM data set by action macros (GET, PUT, etc). Two or more RPL strings may be used for concurrent direct or sequential requests made from a processing program or its subtasks.

S

SAM. (*See* sequential access method.)

security. (*See* data security.)

sequence checking. The process of verifying the order of a set of records relative to some field's collating sequence.

sequence set. The lowest level of the index of a key-sequenced data set; it gives the locations of the control intervals in the data set and orders them by the key sequence of the data records they contain. The sequence set and the index set together comprise the index.

sequential access. The retrieval or storage of a data record in either its entry sequence, its key sequence, or its relative record number sequence, relative to the previously retrieved or stored record. (*See also* addressed-sequential access and keyed-sequential access.)

sequential access method. An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or a'direct access device.

shared resources. A set of functions that permit the sharing of a pool of I/O-related control blocks, channel programs, and buffers among several VSAM data sets open at the same time.

skip-sequential access. Keyed-sequential retrieval or storage of records here and there throughout a data set, skipping automatically to the desired record or collating position for insertion: VSAM scans the sequence set to find a record or a collating position. Valid for processing in ascending sequences only.

slot. For a relative record data set, the data area addressed by a relative record number which may contain a record or be empty.

spanned record. A logical record whose length exceeds control interval length, and as a result, crosses, or spans, one or more control interval boundaries within a single control area.

SRB. Service request block. A system control block used for dispatching tasks.

stage. To move data from an MSS cartridge to a staging drive.

step catalog. A catalog made available for a step by means of the STEPCAT DD statement.

supervisor call instruction. An instruction that interrupts the program being executed and passes control to the supervisor so that it can perform a specific service indicated by the instruction.

SVC. (*See* supervisor call instruction.)

SYNAD. The physical error exit routine.

task control block. The consolidation of control information related to a task.

T

TCB. (*See* task control block.)

terminal monitor program. In TSO, a program that accepts and interprets commands from the terminal, and causes the appropriate command processors to be scheduled and executed.

time sharing option. An optional configuration of the operating system that provides conversational time sharing from remote stations.

TMP. (*See* terminal monitor program.)

transaction ID. A number associated with each of several request parameter lists that define requests belonging to the same data transaction.

TSO. (*See* time sharing option.)

U

UPAD. The user processing exit routine.

update number. For a spanned record, a binary number in the second RDF of a record segment that indicates how many times the segments of a spanned record should be equal. An inequality indicates a possible error.

upgrade set. All the alternate indexes that VSAM has been instructed to update whenever there is a change to the data component of the base cluster.

user buffering. The use of a work area in the processing program's address space for an I/O buffer; VSAM transmits the contents of a control interval between the work area and direct access storage without intermediary buffering.

user catalog. An optional catalog used in the same way as the master catalog and pointed to by the master catalog. It also lessens the contention for the master catalog and facilitates volume portability.

V

vertical pointer. A pointer in an index record of a given level that gives the location of an index record in the next lower level or the location of a control interval in the data set controlled by the index.

virtual storage access method. An access method for direct or sequential processing of fixed and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry sequence), or by relative record number.

virtual volume. The data from a mass storage volume while it is located on a staging drive.

VSAM. (*See* virtual storage access method.)

VSAM resource pool. A virtual storage area that is used to share I/O buffers, I/O-related control blocks, and channel programs among VSAM data sets. A resource pool is local or global; it serves tasks in one partition or address space or tasks in all address spaces in the system.

VSAM shared information. Blocks that are used for cross-system sharing.

VSI. (*See* VSAM shared information.)

Index

MVS/Extended Architecture
VSAM Administration Guide

GC26-4151-5

Reader's Comment Form

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you have applied any technical newsletters (TNLs) to this book, please list them here: ____________________

Comments (please include specific chapter and page references) :

If you want a reply, please complete the following information:

Name ______________________ Date ______________________

Company ______________________ Phone No. (_____) ______________

Address ____

____

Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automatic mail-sorting equipment.
Please use pressure-sensitive or other gummed tape to seal this form.

GC26-4151-5

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department J57
P.O. Box 49023
San Jose, CA 95161-9945

Fold and tape | Please do not staple | Fold and tape

IBM®

Program Number
5665-XA2

File Number
S370-34

GC26-4151-5

Printed in U S A