GC26-3842-1
File No. S370-30

Systems

Planning for Enhanced VSAM under OS/VS

VS1 Release 5
VS2 Release 3.7

IBM

Second Edition (March 1976)

This edition replaces the previous edition (numbered GC26-3842-0) and makes that edition obsolete.

This edition applies to Release 5 of OS/VS1 and to Release 3.7 of OS/VS2 and to subsequent releases of both systems unless otherwise indicated in new editions or technical newsletters.

Significant technical changes are summarized under "Summary of Amendments" following the list of illustrations. In addition, miscellaneous editorial and technical changes have been made throughout the publication. Each technical change is marked by a vertical line to the left of the change.

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *IBM System/370 Bibliography,* GC20-0001, and associated technical newsletters to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, Programming Publishing—Department J57, 1501 California Avenue, Palo Alto, California 94304. All comments and suggestions become the property of IBM.

USING THIS PUBLICATION

VSAM (virtual storage access method) is an access method of OS/VS (operating system/virtual storage). This planning guide enables prospective users to prepare for using VSAM and describes for current users the enhanced functions and capabilities that improve VSAM's performance and make it a more versatile access method for a wider range of applications. The intended audience is data-processing managers whose decisions will influence the use of VSAM, system and application programmers who will use VSAM in both new and existing programs, and others seeking an introduction to VSAM.

This planning guide has six chapters:

- "Introducing VSAM," which outlines how VSAM meets the requirements of an access method in today's data-processing environment.
- "Getting to Know What VSAM Is and Does," which explains the concepts and functions of VSAM and is required reading for the following chapters.
- "Communicating with VSAM," which discusses, primarily for programmers, the multifunction service program Access Method Services, the macros of VSAM, and the use of JCL (job control language) with VSAM.
- "Preparing for VSAM," which indicates, for the planners, the programming languages and optional features of OS/VS that VSAM can be used with.
- "Optimizing the Performance of VSAM," which outlines, for application and system programmers, ways to achieve the best throughput of which VSAM is capable.
- "Protecting Data with VSAM," which describes, for managers and system programmers, VSAM's standard and optional features for data integrity and security.

This publication also has a glossary and an index.

The reader is expected to be familiar with basic concepts such as access method, direct-access storage, and the distinction between data-set organization and data-set processing. The sections dealing with those concepts in *OS/VS Data Management Services Guide,* GC26-3783, are suitable for preparatory reading.

In the chapter "Preparing for VSAM," the section "How Can Existing Programs That Use ISAM Be Used with VSAM?" is intended for those who use ISAM (indexed sequential access method). Other readers may ignore this section and any other references to ISAM. The section of the *Data Management Services Guide* that discusses ISAM is suitable for reference.

The discussion on JCL in the chapter "Communicating with VSAM" presupposes the reader's familiarity with the sections of *OS/VS1 JCL Reference,* GC24-5099, or *OS/VS2 JCL,* GC28-0692, that discuss the JCL parameters described in this planning guide.

Other publications referred to in this publication are:

- *Introduction to Virtual Storage in System/370,* GR20-4260
- *OS/VS Checkpoint/Restart,* GC26-3784

- *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications,* GC26-3819
- *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide,* GC26-3838
- *OS/VS1 Access Method Services,* GC26-3840
- *OS/VS1 Master Index* GC24-5104
- *OS/VS1 Service Aids,* GC28-0665
- *OS/VS1 System Management Facilities (SMF),* GC24-5115
- *OS/VS1 VSAM Cross Reference,* SYB6-3844
- *OS/VS2 Access Method Services,* GC26-3841
- *OS/VS2 Master Index,* GC28-0693
- *OS/VS2 System Programming Library: Service Aids,* GC28-0674
- *OS/VS2 System Programming Library: System Management Facilities (SMF),* GC28-0706
- *OS/VS2 Catalog Management Logic,* SY26-3826
- *OS/VS2 VSAM Cross Reference,* SYB6-3842
- *OS/VS2 Catalog Management Cross Reference,* SYB6-3843
- *OS/VS2 TSO Command Language Reference,* GC28-0646
- *OS/VS2 TSO Guide,* GC28-0644
- *OS/VS2 TSO Terminal User's Guide,* GC28-0645

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

The following changes have been made to this publication:

- Add a new section to describe how VSAM data sets and catalogs are protected. This section describes all the recovery tools available through Access Method Services and VSAM and tells how they are most effectively used to back up data sets and catalogs and to recover from unusable catalogs and inaccessible data sets and volumes.
- Add information about control interval sizes and control interval and control area split strategy relative to freespace specifications.
- Clarify the use of the AMORG subparameter in the VSAM DD AMP parameter.
- Describe the VSAM SNAP Dump Facility (MVS Release 3), which provides a dump of VSAM-owned control blocks in CSA (common service area).
- Add a new section that describes the structure of the VSAM catalog, discusses the data and index components of the catalog, lists catalog control area sizes relative to device types, and describes how catalog space is allocated and utilized.
- Expand the discussion of the cross-region share options.

INTRODUCING VSAM

This chapter is intended for all readers new to VSAM (virtual storage access method). It introduces VSAM, outlines the access-method capabilities that are required in today's data-processing environment, shows how VSAM has those capabilities by describing its area of applicability and summarizing its basic features, and indicates the CPUs (central processing units) and auxiliary-storage devices that VSAM can be used with.

What Is VSAM?

VSAM is a high-performance access method of OS/VS (operating system/virtual storage), option 1 or 2, for use with direct-access storage.

VSAM resides in virtual storage along with the processing program using it. Figure 1 illustrates VSAM's relationship to OS/VS, the processing program, and the data stored on a direct-access storage device and in mass storage.

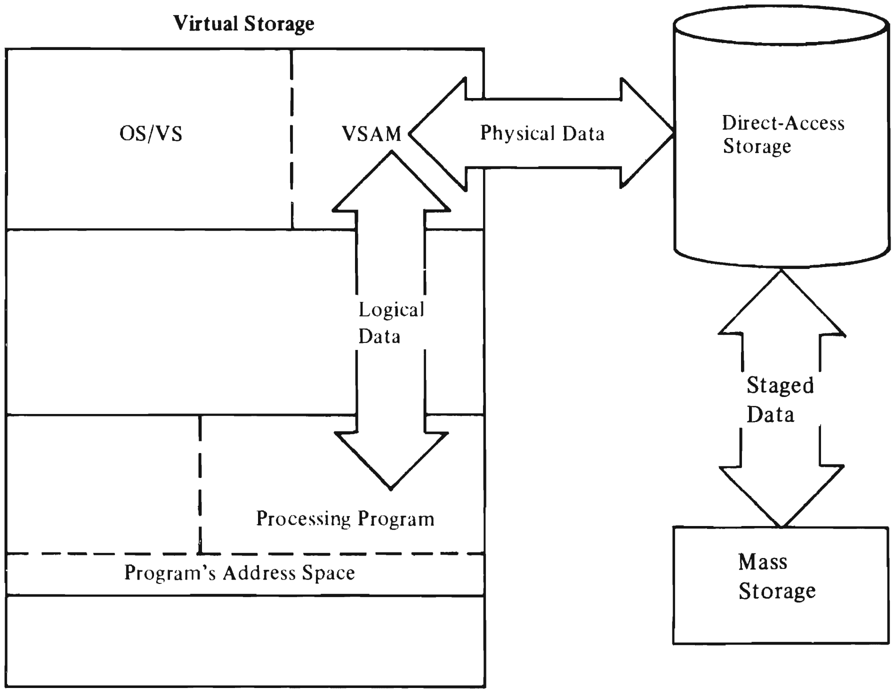


Figure 1. Relationship of VSAM, OS/VS, User's Processing Program, and Staged Data

What Are the Requirements for an Access Method?

In data processing today, it is common for a computer installation to do a number of different types of processing. An installation must provide for one combination or another of data-base processing, online processing, batch processing, inquiry and transaction processing, communications, and multiple CPUs under the control of different operating systems. This variety requires an access method that provides:

- High performance of retrieval and storage—independent of previous insertions of records into data sets and uninterrupted by requirements to reorganize data sets or copy them for backup
- Applicability to different types of processing that require different kinds of access and different levels of performance (such as online and batch processing)
- Simplicity of use by means of a common set of instructions for different types of access, simplified JCL (job control language), and optimization of the use of space in auxiliary storage
- Protection of data: security against unauthorized access and integrity through prevention of intentional or accidental loss of data
- Recovery of data: the ability to recover catalogs and data sets in the event of failure or damage
- Central control over the creation, access, and deletion of data sets and over the management of space by keeping data-set and storage information in one place and making it independent of JCL and processing programs
- Ability to move data from one operating system to another in a format that is common to both systems
- Independence from type of storage device: freedom from physical record size, control information, and record deblocking
- Ease of conversion of data and programs from other access methods to the new access method

How Does VSAM Meet These Requirements?

VSAM provides an approach to meeting these requirements through:

- A format for storing data independently of type of storage device
- Routines for sequential or direct access and for access by key or by relative address
- Options for optimizing performance
- A comprehensive catalog for defining data sets and auxiliary-storage space
- A multifunction service program (Access Method Services) for setting up catalog records and maintaining data sets

High Performance

VSAM's high performance is due to an efficiently organized index, performance options for reducing disk-arm movement and rotational delay, and distributed free space for fast insertion of records and minimal reorganization. The size of the index is reduced by compressing keys to eliminate redundant information. The type of index used for a data set is also used for VSAM catalogs.

VSAM's method of inserting records into a data set provides access whose speed following a large number of insertions is equivalent to the speed of access without previous insertions. Free space is used for efficient automatic reorganization of data sets: inserted records are stored and addressed in the same way as original records, and space given up by deletions is reclaimed as free space within the control interval.

Applicability to Different Types of Processing

VSAM is designed to meet most of the common data-organization needs of both batch and online processing. Batch processing requires the efficiency of sequential and indexed data; online processing requires efficient direct access for random requests. VSAM permits both direct and sequential access access can be by key or by relative address. Different types of processing can be intermixed in the processing of a common data base. You can select the type of access or the combination of types that best suits your application.

| TSO (Time Sharing Option), a subsystem of MVS, can execute Access Method Services commands as TSO commands, dynamically allocate a VSAM data set, and execute a program that uses VSAM macros to process the data set.

VSAM also provides options and macros for sharing a pool of I/O-related control blocks, channel programs, and buffers among several VSAM data sets open at the same time.

Simplicity of Use

There is a common way of requesting the different types of access (sequential or direct, by key or by relative address), so that the same instructions are learned and used for achieving different results.

All VSAM data sets are cataloged, so JCL is simplified. Minimal JCL parameters are required for describing data sets.

VSAM uses default values to establish the size of control intervals and control areas in which data is stored and to manage virtual storage space for I/O (input/output) buffers. Programmers can think in terms of the application, not in terms of the internal workings of VSAM.

Individual data records are passed to a processing program without any system control information: application data alone is processed by the program. Application programmers do not need to know the format of control blocks. They need not be concerned either with storage devices and device addresses or with different formats for fixed-length or variable-length records.

Protection of Data

VSAM protects data by means of its design and its integrity and security options. *Integrity* means the safety of data from inadvertent destruction or alteration; *security* means the protection of data from unauthorized use or purposeful destruction or alteration. VSAM writes records in a way that does not expose data to loss, even if an I/O error occurs. You can specify optional passwords for levels of protection (read-only, update, control, control interval, and full access) and include your own routine to check a requester's authority to gain access to data. You can select options for formatting data sets before data is stored in them and for verifying write operations for data integrity. VSAM also provides various levels of exclusive control for data to be shared between subtasks, regions, and operating systems.

Recovery of Data

VSAM catalogs that are defined with the optional recovery attribute allow data to be recovered. Recovery is based on information recorded on the volumes controlled by the catalog as well as in the catalog itself.

Access Method Services provides commands you can use to test and reestablish a data set's integrity, recover a cataloged object, reestablish objects in the catalog, and list the contents of a catalog recovery area.

Central Control

The VSAM catalog brings together extensive information about data sets and storage space. Access Method Services controls the definition and deletion of data sets and the alteration of information about them in the catalog. Its use is authorized by passwords assigned to the data sets or to the catalog itself. Consequently, the management of your inventory of data sets is centralized and made independent of the use of JCL or the actions of processing programs. Space for data sets can be allocated or deallocated without mounting volumes, because the information describing the contents of VSAM spaces on those volumes is contained in the catalog. You can assign a data set to volumes by ranges of keys that are controlled by the catalog.

Portability of Data Between Systems

VSAM's technique for storing records uses a format that is common to OS/VS and DOS/VS (disk operating system/virtual storage). Communication with VSAM is very similar for both operating systems, except for JCL. Access Method Services includes functions that facilitate moving data sets and volumes from one operating system to another.

Device Independence

VSAM is independent of particular types of storage devices, because it addresses a record in a data set without respect to the physical attributes of auxiliary storage, but with respect to the displacement of the record from the beginning of the data set. The unit in which data is transmitted between virtual and auxiliary storage does not depend on the size of the physical records in which data is stored physically on a volume.

Ease of Conversion

VSAM provides for easy conversion of indexed sequential data sets to VSAM format and the continued use of your existing ISAM (indexed sequential access method) programs to process converted data sets and new VSAM data sets. Access Method Services converts a sequential or an indexed sequential data set to VSAM format. To process the converted data set with the ISAM program, a set of interface routines within VSAM interpret each ISAM request and issue the appropriate VSAM request.

In MVS Release 3 systems, the OS catalog has been replaced by a VSAM master catalog. Access Method Services is used to convert entries in an OS catalog to entries in an existing VSAM master catalog or a VSAM user catalog.

What Machines Can VSAM Be Used With?

You can use VSAM on the following IBM System/370 CPUs:

OS/VS1	**OS/VS2**
Model 135	—
Model 145	Model 145
Model 155 (Model 2)	Model 155 (Model 2)
Model 158	Model 158
—	Model 158 (Model 3)
Model 165 (Model 2)	Model 165 (Model 2)
Model 168	Model 168
—	Model 168 (Model 3)

Each of these CPUs must have the dynamic address translator that is required by OS/VS1 and OS/VS2 and either the advanced control program support feature or the conditional swapping feature. VSAM is designed to take full advantage of the benefits of virtual storage. See *Introduction to Virtual Storage in System/370* for a discussion of virtual storage.

VSAM can be used with all the IBM direct-access storage devices that are supported by OS/VS1 and OS/VS2.

GETTING TO KNOW WHAT VSAM IS AND DOES

Familiarity with the VSAM concepts and terminology introduced in this chapter is presupposed in the following chapters. The concepts are especially important for application programmers who will design and code programs to process data with VSAM, and to system programmers who will maintain the VSAM installation.

This chapter explains the three types of VSAM data sets, discusses how to create and gain access to them, and describes the master catalog and user catalogs.

What Are VSAM's Three Types of Data Sets?

VSAM has key-sequenced, entry-sequenced, and relative record data sets. The primary difference among the three is the order in which data records are loaded into them.

Records are loaded into a *key-sequenced data set* in key sequence: that is, in the order defined by the collating sequence of the contents of the key field in each of the records. Each record has a unique value in the key field, such as employee number or invoice number. VSAM uses an index and optional free space to insert a new record into the data set in key sequence.

Records are loaded into an *entry-sequenced data set* without respect to the contents of the records. Their sequence is determined by the order in which they are physically arranged in the data set: their entry sequence. New records are stored at the end of the data set.

Records are loaded into a *relative record data set* in relative record number sequence. The data set is a string of fixed-length slots, each of which is identified by a relative record number. When a record is inserted, you can assign the relative record number or allow VSAM to assign the record the next available number in sequence. No index is used.

When you create a data set, you define it, together with its index, if any, in a *cluster.* A cluster may be a key-sequenced data set, which consists of a data component and an index component, or an entry-sequenced or relative record data set, which consists of only a data component.

VSAM stores the records of each type of data set in the same way in a fixed-length area of auxiliary storage called a *control interval.* We can better discuss the three types of data sets if we first look at the control interval in perspective with the other logical divisions of a data set and see how and why VSAM uses it for storing records.

The Use of Control Intervals

A control interval is a continuous area of auxiliary storage that VSAM uses for storing data records and control information describing them. It is the unit of information that VSAM transfers between virtual and auxiliary storage. Its size may vary from one data set to another, but for a given data set the size of each control interval in it is fixed, either by VSAM or by you, within limits acceptable to VSAM. VSAM chooses the size based on the type of direct-access storage device used to store the data set, the size of your data records, and the smallest amount of virtual-storage space your processing program will provide for VSAM's I/O buffers.

The information recorded on a track is divided into physical records that are limited by the capacity of a track. The physical-record sizes that VSAM uses begin at 512 bytes and increase by powers of 2 up to 4096 bytes: 512, 1024, 2048, 4096. (The physical-record size of 4096 does not apply to the IBM 2314 Disk Storage.) Control-interval size is limited by the requirements that it be a whole number of physical records (1, 2, 3, ..., up to 64, or a maximum size of 32,768 bytes) and that, if it is greater than 8192 bytes, it be a multiple of 2048. A data set whose control intervals correspond with the tracks of one device might have more or less than one control interval per track if it were stored on a different device. Figure 2 illustrates the independence of control intervals from physical records.

VSAM uses track allocation when you define a data set, if you specify track allocation or specify record allocation requiring less than one cylinder.

The Control Interval in Perspective

How does a data set relate to the physical attributes of auxiliary storage? And how does a control interval relate to a data set?

A volume can contain areas for VSAM's use and areas for the use of other access methods or the operating system. A storage area defined in the volume table of contents for VSAM's exclusive use is called a *data space.* It can be extended beyond its original size to include up to 16 areas *(extents)* that need not be adjacent to one another on the volume.

A data set is stored in a data space or data spaces on one or more volumes on direct-access devices of the same type. When you define a data set, you can allocate enough space to have some left at the end of the data set for additions. For new data sets, the amount requested must be available, or DEFINE will terminate. Otherwise, when additional space is needed, VSAM automatically extends the data set by the amount of space indicated in the definition of the data set in the catalog. It can be extended beyond its original size to include up to 123 extents, or to a maximum size of 2^{32} (approximately 4,290,000,000) bytes. Figure 3 illustrates the relationships among volumes, data spaces, and data sets. The figure shows portions of data sets A and C stored in different data spaces on different volumes.

A data set is made up of control intervals. A group of control intervals makes up a *control area.* It is the unit of a data set that VSAM preformats for data

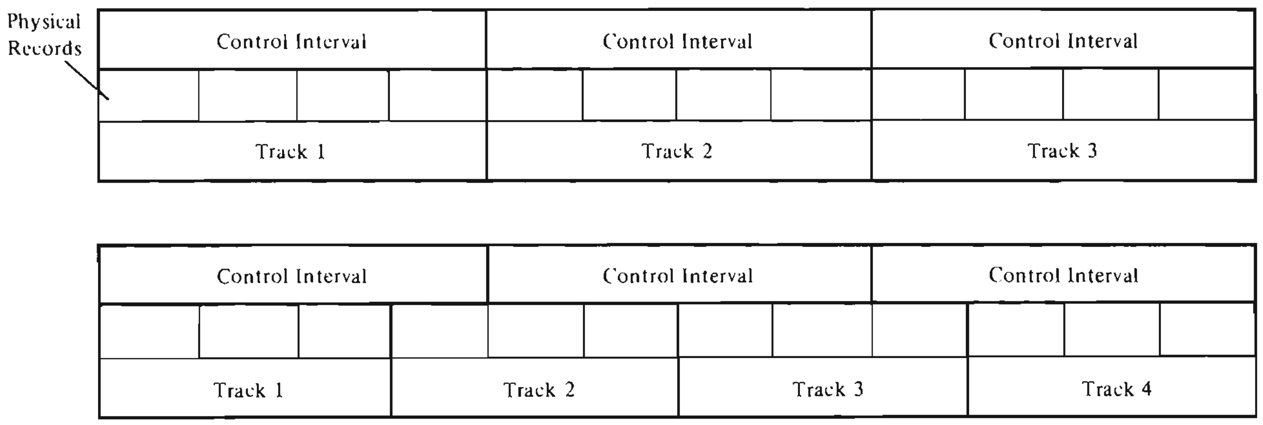


Figure 2. Control Intervals Are Independent of Physical Record Size

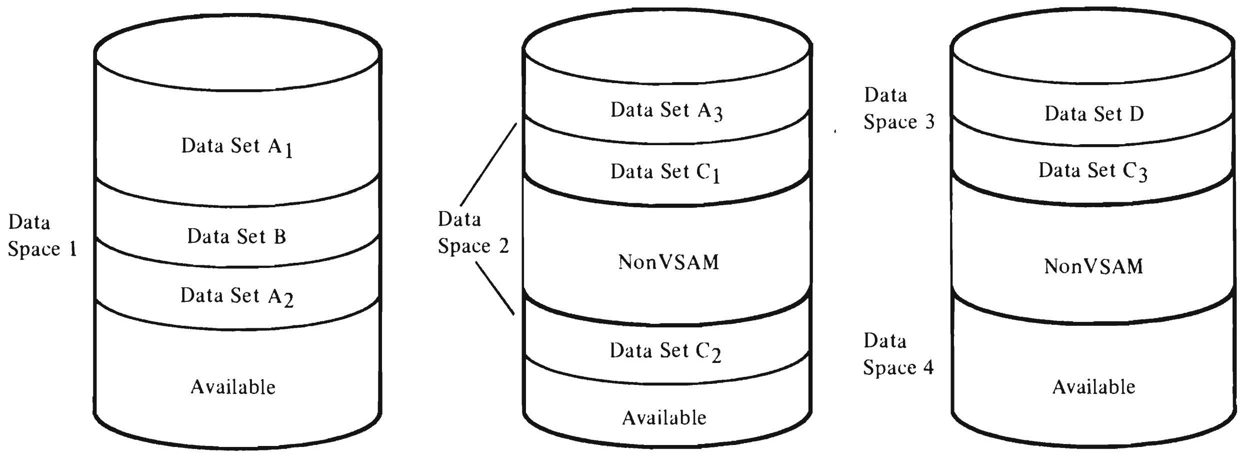


Figure 3. Relationship Among Storage Volumes, Data Spaces, and Data Sets

integrity as records are added to the data set. (See the section "Method of Indicating the End of a Data Set" in the chapter "Protecting Data with VSAM.") In a key-sequenced data set, control areas are also used for distributing free space throughout the data set as a percent of control intervals per control area and for placing portions of the index adjacent to the data set.

The number of control intervals per control area of a data set is fixed by VSAM, with a minimum of two. If 50 were the number chosen, for example, the first 50 control intervals would be the first control area; the next 50 would be the second control area, and so on. Whenever the space for a data set is extended, it is extended by a whole number of control areas. For a key-sequenced data set, the size of a control area is determined on the basis of the space-allocation request, user-specified or default index and data control-interval size, and available buffer space.

The Method of Storing a Record in a Control Interval

The records of a key-sequenced or entry-sequenced data set may be either fixed or variable in length; the records of a relative record data set are always fixed in length. VSAM treats them all the same. It puts control information at the end of a control interval to describe the data records stored in the control interval: the combination of a data record and its control information, though they are not physically adjacent, is called a *stored record.* When adjacent records are the same length, they share control information. Figure 4 shows how data records and control information are stored in a control interval. The data records are stored at the beginning of a control interval, and control information at the end.

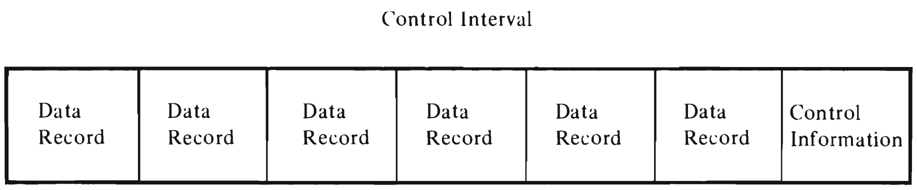


Figure 4. Placement of Data Records and Control Information in a Control Interval

When you define a data set, you can specify enough buffer space for the control intervals in the data set to be large enough for your largest stored record.

Key-sequenced and entry-sequenced data set records whose lengths exceed control interval size may cross, or span, one or more control interval boundaries. Such records are called *spanned records.* A spanned record always begins on a control interval boundary and fills one or more control intervals within a single control area. As shown in Figure 5, the control interval that contains the last segment of a spanned record can contain unused space. This free space can be used only to extend the spanned record; it cannot contain all or part of any other record. You must specify your intent to use spanned records when you define the data set.

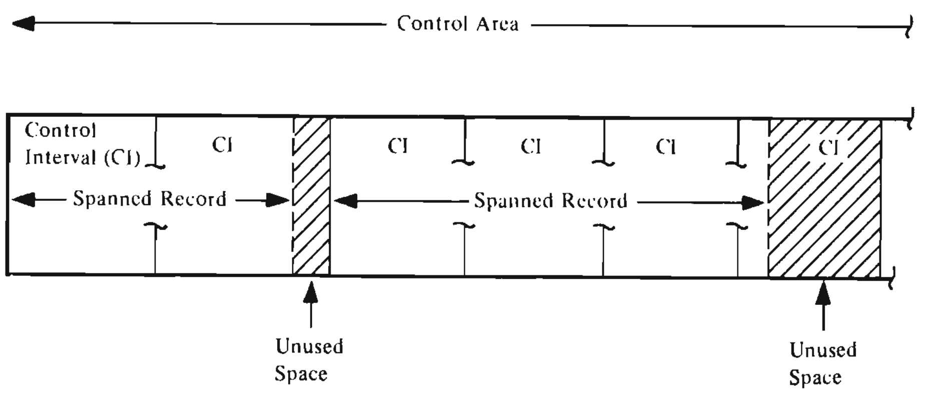


Figure 5. Control Intervals That Contain Spanned Records

A data record is addressed not by its location in terms of the physical attributes of the storage device (such as the number of tracks per cylinder), but by its displacement, in bytes, from the beginning of the data set, called its *RBA (relative byte address).* The RBA does not depend on how many extents belong to the data set or on whether they are in different data spaces or on different volumes. For relative byte addressing, VSAM considers the control intervals in the data set to be contiguous, as though the data set were stored in virtual storage beginning at address 0. For example, the first record in a data set has RBA 0. The second record has an RBA equal to the length of the first record, and so on. The bytes required for intervening control information and free space are included in the RBA value.

Key-Sequenced, Entry-Sequenced, and Relative Record Data Sets

The purpose of this section is to describe VSAM's three types of data sets in detail, to discuss how free space can be distributed in a key-sequenced data set, and to explain further how VSAM uses the control interval for data storage. Figure 6 contrasts the three types by listing the attributes of each.

Key-Sequenced Data Sets

A key-sequenced data set is always defined with an index that relates key values to the relative locations of the data records in a data set. (This index is the *prime index,* in contrast to *alternate indexes,* which are discussed later.) The prime index and distributed free space used to insert a new record in key sequence are discussed in the paragraphs that follow.

Key-Sequenced Data Set	Entry-Sequenced Data Set	Relative Record Data Set
Records are in collating sequence by key field	Records are in the order in which they are entered	Records are in relative record number order
Access is by key through an index or by RBA	Access is by RBA	Access is by relative record number, which is treated like a key
May have one or more alternate indexes	May have one or more alternate indexes	May not have alternate indexes
A record's RBA can change	A record's RBA cannot change	A record's relative record number cannot change
Distributed free space is used for inserting records and changing their length in place	Space at the end of the data set is used for adding records	Empty slots in the data set are used for adding records
Space given up by a deleted or shortened record is automatically reclaimed within a control interval	A record cannot be deleted, but you can reuse its space for a record of the same length	Space given up by a deleted record can be reused
Can have spanned records	Can have spanned records	Cannot have spanned records
Can be reused as a work file unless it has an alternate index, is associated with key ranges, is unique, or exceeds 16 extents per volume	Can be reused as a work file unless it has an alternate index, is associated with key ranges, is unique, or exceeds 16 extents per volume	Can be reused as a work file

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

An index relates key values to the relative locations of the data records. A key in the index is taken from a record's key field, whose size and position are the same for every record in the data set, and whose value cannot be altered. VSAM uses an index to locate a record for retrieval and to locate the collating position for insertion.

An index has one or more levels, each of which is a set of records that contains entries giving the location of the records in the next lower level. The index records in the lowest level are collectively called the *sequence set;* they give the location of control intervals containing the data records. The records in all the higher levels are collectively called the *index set;* they give the location of index records. The highest level always has only a single record. The index of a data set with few enough control intervals for a single sequence-set record has only one level: the sequence set itself.

Figure 7 illustrates the levels of a prime index and shows the relationship between a sequence-set index record and a control area. The figure shows that the highest-level index record (A) controls the entire next level (records B through Z); each sequence-set index record controls a control area.

An entry in an index-set record consists of the highest key that an index record in the next lower level contains, paired with a pointer to the beginning of that index record. An entry in a sequence-set record consists of the highest key in a control interval of the data component, paired with a pointer to the beginning of that control interval. Not all data records have sequence-set entries, for there is only one entry for each control interval in the data set.

For direct access by key, VSAM follows *vertical pointers* from the highest level down to the sequence set to find a vertical pointer to data; for sequential access by key, VSAM refers only to the sequence set. It uses a *horizontal pointer* in a sequence-set record to get from that sequence-set record to the one containing the next key in collating sequence so it can find a vertical pointer to data. Figure 7 shows both vertical pointers and horizontal pointers.

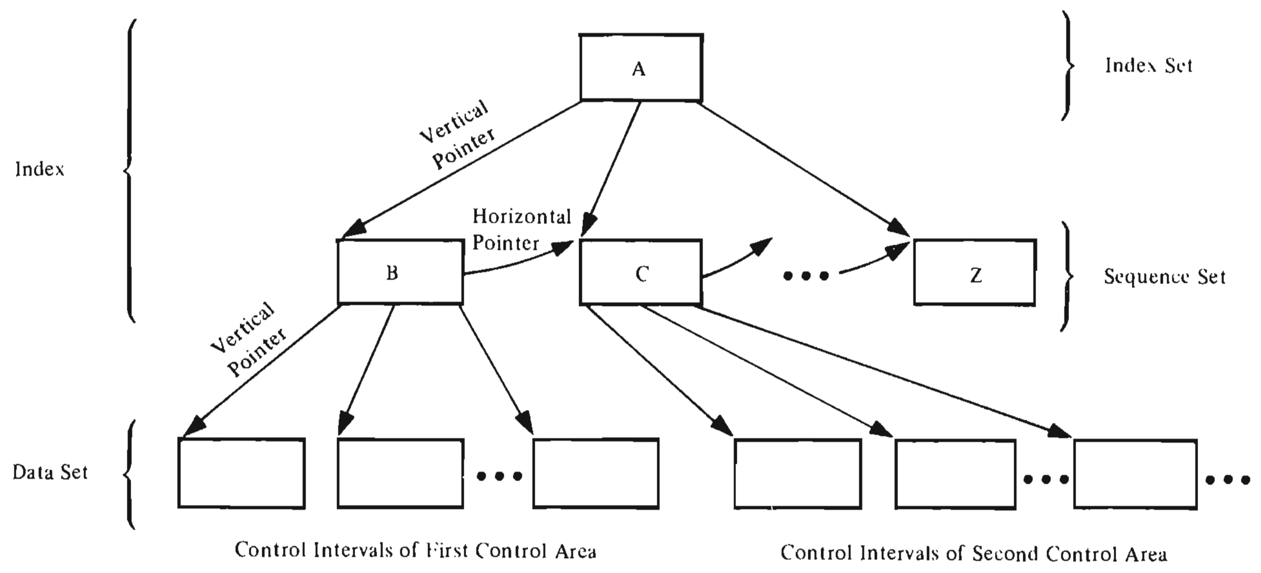


Figure 7. Relationship Among the Levels of a Prime Index and a Data Set

VSAM increases the number of entries that an index record of a given size can hold by a method of *key compression:* it eliminates from the front and the back of a key those characters that aren't necessary to distinguish it from the adjacent keys. Compression helps achieve a physically smaller index by reducing the size of keys in index entries. Key compression, in an index of a particular physical size, allows you to gain access to many more records than you could otherwise.

The number of control intervals in a control area equals the number of entries in a sequence-set index record. This equality has important uses in:

- Placing the sequence-set index record adjacent to the control area on a single cylinder (see "Sequence-Set Records Adjacent to the Data Set" in the chapter "Optimizing the Performance of VSAM")
- Distributing free space throughout a data set as a percent of free control intervals in each control area

When you define a key-sequenced data set, you can specify that free space is to be distributed throughout it in two ways: by leaving some space at the end of all the used control intervals and by leaving some control intervals completely empty. The amount of free space in a used control interval and the number of free control intervals in a control area are independent of each other. The selection of optimum free space values depends on whether your program does direct processing, sequential processing, or both. Figure 8 shows how free space might be set aside in each control area of a data set. The sequence-set record for a control area contains an entry for each free control interval, as well as for each of those that contain data.

Besides the space that you distribute when you create a key-sequenced data set, space that becomes available within a control interval when a record is shortened or deleted from the data set is automatically reclaimed by VSAM and can be used when a record is lengthened in place or directly inserted into the control interval.

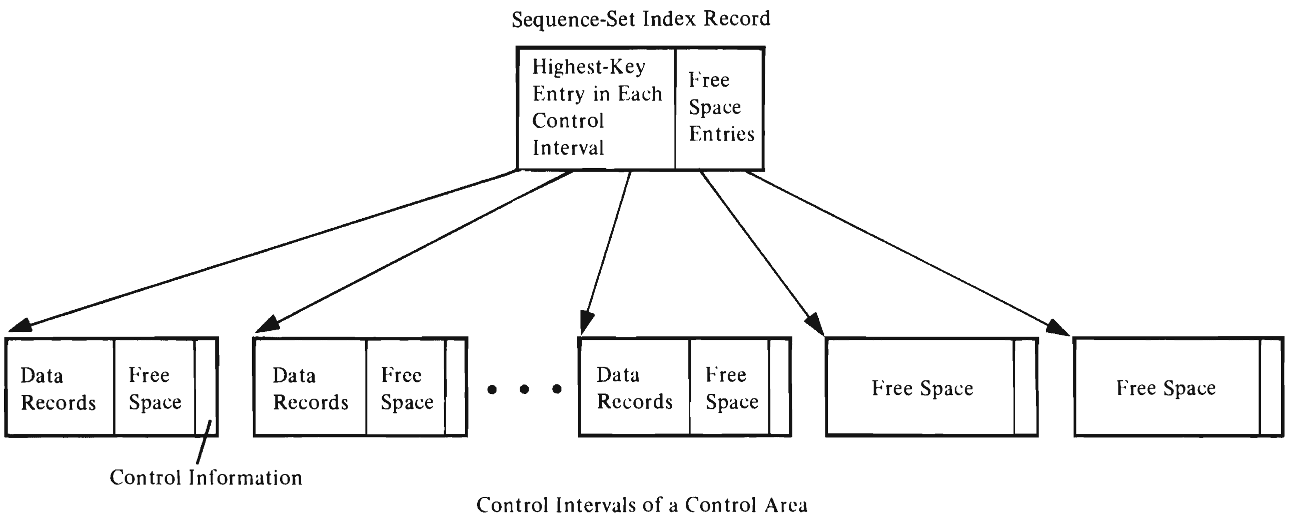


Figure 8. Distribution of Free Space in a Key-Sequenced Data Set

Reclaiming space and using distributed free space may cause RBAs of some records to change. As Figure 8 illustrates, free space within a used control interval is between the data in the front and the control information in the back. If a record is deleted or shortened, any succeeding records in the control interval are moved to the left and their RBAs are changed so that the space vacated can be combined with the free space already in the control interval. Conversely, an insertion or a lengthening causes any succeeding records in the control interval to be moved to the right into free space and their RBAs to be changed.

The discussion thus far has assumed that there is enough free space in the control interval for a new or lengthened record. The following paragraphs describe what VSAM does when there is not enough free space in the control interval to contain the record. For simplicity, only insertion is referred to explicitly.

If the record to be inserted will not fit in the control interval, there is a *control interval split:* VSAM moves stored records in the control interval to an empty control interval in the same control area, and inserts the new record in its proper key sequence. The number of records moved depends on the position of insertion of the new record and on the type of insertion.

For sequential insertion, records are inserted in the control interval leaving any specified free space; when the next record to be inserted will not fit in the control interval, the records with keys greater than the key of the record to be inserted are moved to a new control interval. The new record is inserted in the old control interval, if space permits; otherwise, the new record is moved to a third control interval. If there are no records in the control interval with a key greater than the new record, the new record is placed in a new control interval.

For direct insertion, approximately half of the records in a control interval are moved when a control-interval split is required.

Figure 9 illustrates a control-interval split caused by a sequential insert and shows the resulting free space available in the two affected control intervals. Some of the records in the control interval that is too full for insertion are moved to a free control interval, and the new record is inserted into the control interval according to key sequence. Because the number of records in the first control interval is reduced, subsequent insertions revert to the simpler case, instead of becoming more complex.

If the control intervals involved in a split are not adjacent, the physical sequence of data records is no longer the same as their key sequence. In Figure 9, the entry sequence of the records in the last three control intervals on the right is: 55, 56, 57, 60, 61, 58, 59. But the sequence-set index record reflects the key sequence, so that, for keyed sequential requests, the data records are retrieved in the order: 55, 56, 57, 58, 59, 60, 61.

For a listing of the sequence in which VSAM performs write and update operations for a control-interval split, see "Method of Inserting Records into a Key-Sequenced Data Set" in the chapter "Protecting Data with VSAM."

Should there not be a free control interval in the control area, an insertion requiring a free control interval causes a *control area split:* VSAM establishes a new control area, either by using space already allocated or by extending the data set, if the initially allocated space is full and you provided for extensions when you defined the data set. VSAM moves the contents of approximately half of the control intervals in the full control area to free control intervals in the new control area and inserts the new record into one of the two control areas, as its key dictates. Since about half of the control intervals of each of these control areas are now free for a direct or skip-sequential insert, subsequent insertions won't require control-area splitting. Likewise, subsequent sequential inserts will be added to the end of the control area, and additional splits will not be required. Splitting should be an infrequent occurrence for data sets with sufficient distributed free space; splitting a control area does make it possible, however, to insert records into a key-sequenced data set without previously distributed free space.

Generally speaking, direct or skip sequential inserts cause control intervals and control areas to be split at the mid-point. Also, when processing direct or

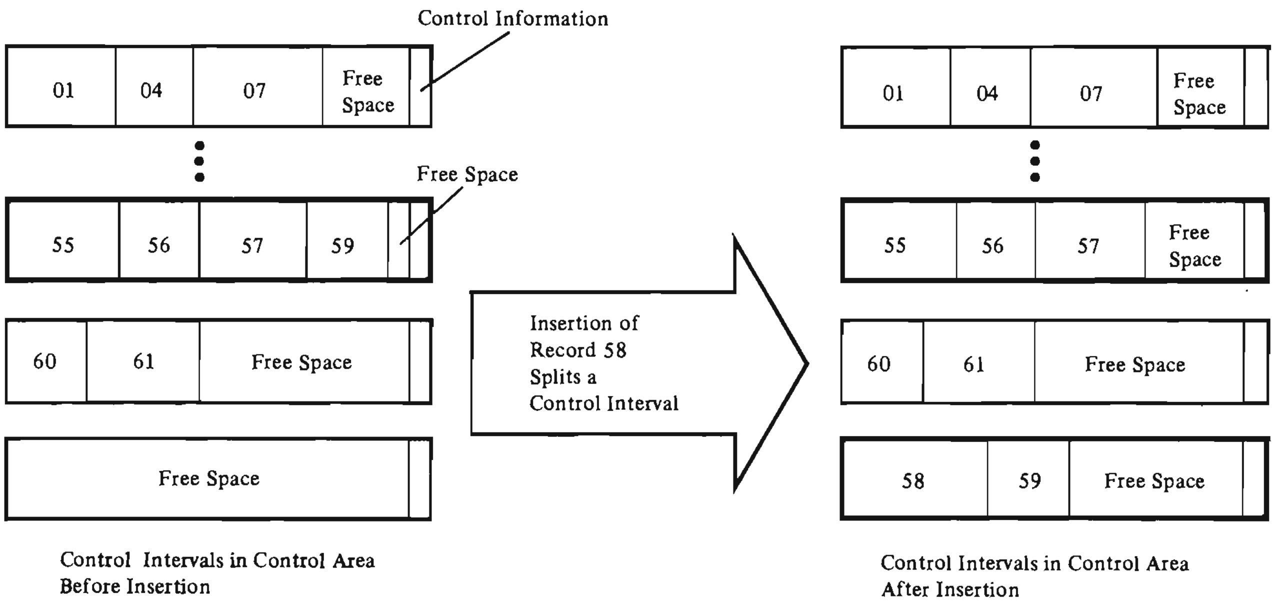

Control Information
01
04
07
Free Space
Free Space
55
56
57
59
60
61
Free Space
Free Space
Insertion of Record 58 Splits a Control Interval
01
04
07
Free Space
55
56
57
Free Space
60
61
Free Space
58
59
Free Space
Control Intervals in Control Area Before Insertion
Control Intervals in Control Area After Insertion

Figure 9. Splitting a Control Interval for Record Insertion

skip sequential inserts, VSAM attempts to use all the free space available in the control interval or control area. Sequential inserts, on the other hand, cause control intervals and control areas to split at the point of insertion. Furthermore, during sequential inserts, VSAM attempts to reserve the free space quantity defined for the data set.

Key-sequenced data sets are appropriate for most applications. You can use the full range of VSAM's processing options to gain access to your data by a key field rather than some location-dependent manner. A simplified approach to planning is to assume that you will store your records in key-sequenced data sets and handle as exceptions those applications that are more suited to entry-sequenced or relative record data sets.

For additional information about processing key-sequenced data sets, see the section "In What Ways Can VSAM Data Sets Be Processed?" in this chapter.

Entry-Sequenced Data Sets

No prime index is associated with an entry-sequenced data set. When a record is loaded or subsequently added, VSAM indicates its RBA to you. You must keep track of the RBAs of the records yourself to gain access to them by direct processing. One way to keep track is to build your own index.

Sequential access with an entry-sequenced data set is similar to that of QSAM (queued sequential access method).

You can use direct access with an entry-sequenced data set in a way similar to BDAM (basic direct access method) by preformatting the data set with records of your choice (filled with blanks, for instance) and providing a routine that randomly associates an RBA with the key field of a record in the data set and thus distributes records throughout the data set. To store a record initially, you convert its key field to an RBA, retrieve the preformatted record at that RBA, and store the new record back at that RBA. The routine must have a procedure for determining an alternate RBA when two or more keys are converted to the same RBA. To retrieve a record subsequently, you convert its key field to its RBA and determine the alternate RBA, if one is required.

An entry-sequenced data set is appropriate for applications that require no particular ordering of data by the contents of a record. Thus, it is well-suited for a log or a journal in which the order corresponds to a sequence of events.

For additional information about processing entry-sequenced data sets, see the section "In What Ways Can VSAM Data Sets Be Processed?" in this chapter.

Relative Record Data Sets

A relative record data set has no index. It has a string of fixed-length slots, each of which has a relative record number from 1 to n, the maximum number of records that can be stored in the data set. Each record occupies a slot and is stored and retrieved by the relative record number of the slot. Relative record number 9 in Figure 10, for example, occupies the ninth slot; whether slots 1 through 8 are filled makes no difference.

Records in a relative record data set are grouped together in control intervals, just as they are in a key-sequenced or an entry-sequenced data set. Each control interval contains the same number of slots, the size of which is the record length you specified when you defined the data set. The number of

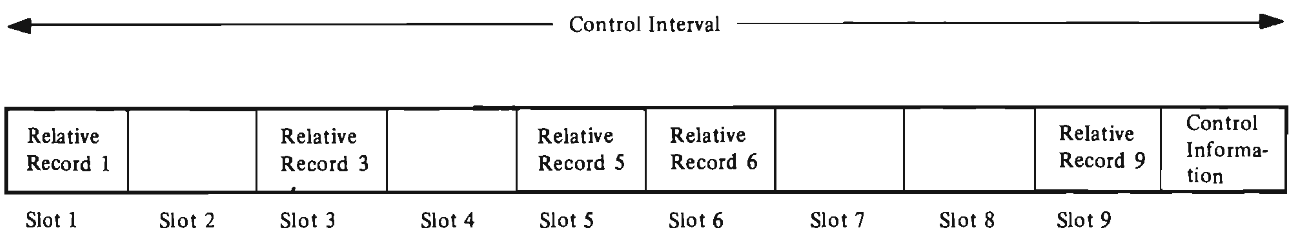


Figure 10. The First Control Interval Within a Relative Record Data Set

slots in a control interval is determined by the control interval size and the record length.

Because the slot can contain data or be empty, a data record can be inserted, deleted, or moved without affecting the position of other data records in the relative record cluster. Records can be retrieved sequentially. They can also be retrieved directly, based on relative record number. You cannot retrieve a relative record based on its RBA.

A relative record data set is appropriate for many applications that use fixed-length records. Each record could be processed to yield a unique relative record number (an employee's serial number, for example). Each record could be located as though it were in a key-sequenced cluster, but without the time it takes to search the key-sequenced cluster's index.

Additional information about processing relative record data sets appears in the section "In What Ways Can VSAM Data Sets Be Processed?" in this chapter.

How Are Alternate Indexes Used with Key-Sequenced and Entry-Sequenced Data Sets?

An alternate index provides a unique way to gain access to a related base data set, so that you need not keep multiple copies of the same information organized in different ways for different applications. For example, a payroll data set indexed by employee number can also be indexed by other fields such as employee name or department number.

You use Access Method Services to define and build one or more alternate indexes over a key-sequenced or an entry-sequenced data set. See "How is Access Method Services Used?" in the chapter "Communicating with VSAM."

This section describes the components of an alternate index and explains how they are related to the base data set. It defines new terms associated with alternate indexes, describes alternate-index records, keys, and pointers and describes how alternate indexes are maintained.

Base Clusters and Alternate-Index Clusters

In terms of access, an alternate index performs the same function as the prime index of a key-sequenced data set. The data set over which the alternate index is built is the *base cluster.* It can be a key-sequenced or an entry-sequenced data set, but not a relative record or a reusable data set.

In structure, the alternate index is similar to a cluster. It consists of an index component and a data component. The index component is identical in structure, format, and function to the prime index of a key-sequenced cluster. Likewise, the format of the alternate-index data component is identical to the format of the data portion of a key-sequenced data set. Therefore, each entry in the sequence set of an alternate-index index component points to a control interval in the alternate-index data component.

When building an alternate index, you can use as the *alternate key* any field in the base data set's records having a fixed length and a fixed position within each record. The alternate key must be in the first segment of a spanned record. For each alternate key, the data component of the alternate index contains a unique record. This record consists of the alternate key itself, followed by a pointer that is the prime key or RBA of the base data record that contains the alternate key. If more than one base data record contains the alternate key then the alternate index record contains a pointer to each base data record. These duplicate, or *nonunique* keys are discussed in the section "Alternate Keys" in this chapter.

Alternate-Index Paths

A *path* logically relates a base cluster and each of its alternate indexes. It provides a way to gain access to the base data through a specific alternate index. You define a path through Access Method Services. You must name it and you can give it a password, if you choose. The path name subsequently refers to the base cluster/alternate-index pair. This means that when you refer to a path (by way of the OPEN macro, for example), both the base cluster and the alternate index are affected (opened). Figure 11 shows how two paths can relate two alternate indexes to a single base cluster.

Alternate-Index Records

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

System Header Information

System header information is fixed length and indicates:

- Whether the alternate index record contains (1) prime keys or RBA pointers and (2) unique or nonunique keys
- The length of each pointer
- The length of the alternate key
- The number of pointers

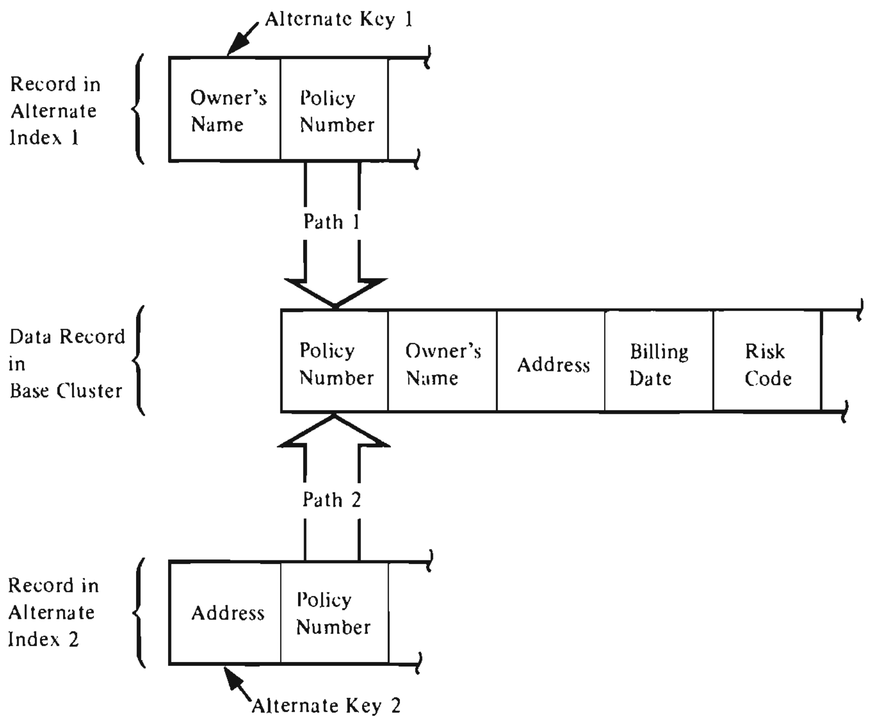


Figure 11. Two Alternate Indexes Over a Single Key-Sequenced Data Set

Alternate Keys

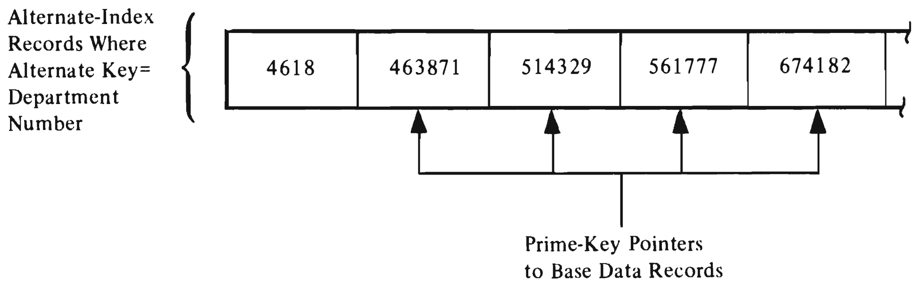
Unless the base data records span control intervals, any field in the base data records that has a fixed length and a fixed position within the record can be an alternate key. The alternate key must be in the first control interval of a spanned record. When an alternate index is created, the alternate keys are extracted from the base data records and ordered in collating sequence. If you build several alternate indexes over a base cluster, the alternate key fields of the different alternate indexes may overlap each other in the base data records. They can also overlap the prime key.

Keys in the index component of an alternate index or of a key-sequenced base cluster are compressed. Keys in the data component of an alternate index are not compressed. That is, the entire key is represented in the alternate-index data record.

An alternate key may refer to more than one record in the base cluster. For example, if an alternate index is established by department number over a payroll data set organized by employee number, there will be several employees with the same department number, as shown in Figure 12. In other words, there will be several prime-key pointers (employee numbers) in the alternate-index record: one for each occurrence of the alternate key (department number) in the base data set. When multiple pointers are associated with a given alternate key value, the alternate key is said to be *nonunique;* if only one pointer is associated with the alternate key, it is *unique.*

Base Data Records Where Prime Key= Employee Number

Employee Number	Name	Department Number	Other Information
463871	Martin, AB	4618	...
488797	Downs, CD	1201	...
514329	Michaels, EF	4618	...
561777	Price, GH	4618	...
568597	Sonders, IJ	2436	...
674182	West, KL	4618	...
⋮	⋮	⋮	...



Alternate-Index Records Where Alternate Key= Department Number
4618
463871
514329
561777
674182
Prime-Key Pointers to Base Data Records

Figure 12. Nonunique Alternate Keys

Alternate-Index Pointers

An alternate index uses prime keys if the base cluster is a key-sequenced data set and RBAs if the base cluster is an entry-sequenced data set.

For a nonunique key, like department number in Figure 12, multiple pointers are associated with it. The pointers are ordered by their arrival times. That is, if a base data record is updated with a key change (for example, an employee number in Figure 12 is changed), or if a new record is inserted with the same alternate key value (department number in Figure 12), the new prime-key pointer is added to the end of the alternate-index record. In the case of a key change, the old pointer is deleted.

A prime-key pointer has the same length as the prime key field of the base data record it points to. The maximum number of pointers that can be associated with a given alternate key is 32767, provided the maximum possible record length for spanned records is not exceeded.

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 maintenance 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, VSAM updates the alternate index immediately when there is a change to the associated base data cluster. VSAM opens all the UPGRADE alternate indexes for a base cluster whenever the base cluster is opened for output (but not control interval processing).

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 the upgrade fails because of a logical error, VSAM attempts to nullify any modifications made to the base data or to other alternate indexes, and the request that caused the upgrade is rejected.

If you specify NOUPGRADE when the alternate index is defined, you must provide a way to reflect insertions, deletions, and changes made to the base cluster in the associated alternate index.

When a path is opened for update, JCL allocates the base cluster and all the alternate indexes in the upgrade set. If allocating the alternate indexes is unnecessary, you can specify NOUPDATE and cause JCL to allocate only the base cluster. VSAM, in that case, does no automatic upgrading.

How Are VSAM Data Sets Created?

This short discussion on creating data sets is intended merely to introduce the following description of data access and VSAM catalogs. See "How Is Access Method Services Used?" in the chapter "Communicating with VSAM" for a more detailed discussion of defining data sets and loading records into them.

To define a VSAM data set, you use Access Method Services to allocate storage space for it and catalog it in either the master catalog or a user catalog. You can load data records into a data set by having Access Method Services copy them from a sequential, an indexed sequential, or another VSAM data set, or you can load them with your-own processing program.

In What Ways Can VSAM Data Sets Be Processed?

VSAM allows both sequential and direct access for each of its three types of data sets. Sequential access of a record depends on the position, with respect to the key, the relative byte address of the previously processed record, or the relative record number; direct access does not. During sequential access, records retrieved by key are in key sequence, records retrieved by RBA are in entry sequence, and records retrieved by relative record number are in relative record number sequence. To retrieve records after initial positioning, you don't need to specify a key, an RBA, or a relative record number. VSAM automatically retrieves or stores the next record in order, either next in key sequence, next in entry sequence, or next in relative record number sequence, depending on whether you're processing by key, by RBA, or by relative record number.

With direct access, the retrieval or storage of a record is not dependent on the key, the RBA, or the relative record number of any previously retrieved record. You must fully identify the record to be retrieved or stored by key, by RBA, or by relative record number.

GET-previous processing is a variation of normal keyed or addressed sequential processing. Instead of retrieving or updating the next record in ascending sequence (relative to current positioning in the data set), GET-previous processing returns or updates the next record in descending sequence.

VSAM allows a processing program or its subtasks to process a data set with multiple concurrent sequential and/or direct requests, each requiring that

VSAM keep track of a position in the data set, with a single opening of the data set. Access can be to the same part or to different parts of a data set. See "How Is Shared Data Protected?" in "Protecting Data with VSAM" for information about how VSAM provides for the protection of shared data.

A VSAM data set that does not have an alternate index can be used as a work file. That is, you can treat a filled data set as if it were empty and use it again and again, regardless of its old contents. To reuse a data set, you need only to define it as reusable and specify that it be reset when you open it.

VSAM provides programmers of utilities and systems with *control-interval access*. It retrieves and stores the contents of a control interval, rather than a single data record. With control-interval access, access is gained sequentially or directly by RBA and you can manage your own buffers or let VSAM do that for you, unless you select the high-performance option. This option reduces the number of instructions that must be executed in systems that demand greater speed. With this option, you must manage your own buffers and issue only direct-access requests. Detailed information about control-interval access is included in *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications*. The control format of the information may change in future releases of VSAM.

For a key-sequenced data set, the primary form of access is keyed access, using an index. For an entry-sequenced data set without an alternate index, the only forms of access are addressed and control interval access, using the RBA determined for a record when it was stored in the data set. For a relative record data set, the only forms of access are keyed and control interval access, using the relative record number as a key.

You can also use addressed access to process the data or index component of a key-sequenced data set. This may be useful when recovering from an index failure. Keyed insertion and deletion may change the RBAs of records, so you should provide a routine to record those changes during processing. VSAM will exit to that routine at the appropriate time.

When your processing program retrieves a record, VSAM reads the contents of the record into virtual storage (unless the contents have been read in previously). VSAM does not require the processing program to deblock records. VSAM indicates the length of the data record to your program and either places the record in your program's work area or gives your program the record's address in VSAM's I/O buffer. You need not concern yourself with any physical attributes of stored records.

If you must manage your own I/O buffers, you may. For additional information, see "RPL: Defining the Request Parameter List" in the chapter "Communicating with VSAM." Detailed information on how to manage your own I/O buffers is in *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications.*

Keyed Access for Key-Sequenced and Relative Record Data Sets

Keyed access is for key-sequenced and relative record data sets. The relative record numbers of the records in a relative record data set are treated as keys. Keys or relative record numbers are specified and returned in the area pointed to by the ARG field of the RPL.

Keyed access provides for retrieval, update, insertion, addition, and deletion. Each of these actions can be sequential, skip sequential, or direct when you are processing records in ascending key sequence. The actions can be sequential or direct when you are processing in descending key sequence.

In forward-sequential processing, records are retrieved or stored in ascending key or relative record sequence, starting from the beginning of the data set or another position that you select. You do not have to supply a search argument for VSAM to process the records. With direct processing, records are retrieved or stored by the search argument (key or relative record number) you supply. Records can be processed in any order, without regard to the sequence of records processed before or after. During skip-sequential processing, you can retrieve or store a group of records sequentially (in ascending sequence) and then skip to a different part of the data set and process another group of records sequentially. Skip sequential combines features of both sequential and direct processing.

When you specify GET-previous processing, VSAM will return the previous record relative to current positioning rather than the next record in the data set. You can select previous processing for POINT, GET, PUT (update only), and ERASE operations. GET-previous processing is not permitted with control-interval or skip-sequential processing.

Keyed Retrieval

Keyed sequential access for a key-sequenced data set depends on where the previous macro request positioned VSAM with respect to the key sequence defined by the index. When your program opens the data set for keyed access, VSAM is positioned at the first record in the data set in key sequence to begin keyed sequential processing. The POINT macro instruction positions VSAM at the record whose key you specify. If the key is a leading portion of the key field, a *generic key*, the record positioned to is the first of the records having the same generic key. A subsequent GET macro retrieves the record VSAM is positioned at. The GET then positions VSAM at the next record in key sequence. The POINT macro can position either forward or backward in the data set, depending on whether FWD or BWD was specified for the OPTCD operand.

When you are processing by way of 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 nonunique alternate key, the 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. 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.

Keyed sequential retrieval for a relative record data set causes the records to be returned in ascending or, if you are retrieving the previous record, descending numerical order, based on the current positioning for the data set. Positioning is established in the same way as for a key-sequenced data set, and the relative record number is treated as a full key. If a deleted record is encountered during sequential retrieval, it is skipped over and the next record is retrieved. The relative record number of the retrieved record is returned in the ARG field of the RPL.

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 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 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.

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 ARG operand 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 return code (duplicate key) is set in the RPL. To retrieve the remaining records with the same alternate key, specify the NSP option when retrieving the first record and then switch to sequential processing.

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 ARG operand of the RPL macro. If you request a deleted record, the request will cause a no-record-found logical error.

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

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.

Keyed Storage

To store records in ascending key sequence throughout a data set, you can use sequential, skip-sequential, or direct access. For sequential or skip-sequential processing, VSAM scans the sequence set of the index; for direct processing, VSAM searches the index from top to bottom.

A PUT macro instruction stores a record. A PUT for update following a GET for update stores the record that the GET retrieved. To update a record, you must previously have retrieved it for update.

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, thereby reducing 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.

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).

Sequential insertion in a relative record data set causes a record to be assigned the next available number in sequence, which is the next available relative record number greater than the position established by a previous record. The assigned number is returned in the ARG 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 ARG field of the RPL. You must insert the record into a slot that does not contain a record.

You can insert and update data records in the base cluster by way of a path, provided the PUT request does not result in nonunique alternate keys in an alternate index which you have defined with the UNIQUE parameter. If the alternate index is in the upgrade set (that is, you specified UPGRADE when you defined the alternate index), the alternate index is modified automatically when you insert or update 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.

Keyed Deletion

An ERASE macro instruction that follows a GET for update deletes the record that the GET retrieved. A record is physically erased in the data set when you delete it. 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.

You can erase a record from a relative record data set after you have retrieved the record for update. 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.

Addressed Access for Key-Sequenced and Entry-Sequenced Data Sets

For an entry-sequenced data set, the only forms of access are addressed and control interval access, using the RBA determined for a record when it was stored in the data set. Control-interval access is described in *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications.* You can also use addressed access for both key-sequenced and entry-sequenced data sets when you want to process the previous record. The previous record is the one with the next lower RBA. Positioning is established in the same way it is for keyed retrieval, except that to start processing at the end of the data set, you issue a POINT macro with OPTCD=LRD. During the processing of previous records, you cannot add or insert records.

Addressed access can be either sequential or direct for both key-sequenced and entry-sequenced data sets, but the processing allowed for a key-sequenced data set is different from that allowed for an entry-sequenced data set. With a key-sequenced data set, addressed access can be used to retrieve records, update their contents, and delete records. (The length of a record and the contents of its key field cannot be changed.) Records cannot be added because VSAM will not allow changes to the data set which could cause the index to change. With an entry-sequenced data set, addressed access can be used to retrieve records, update their contents (but not change their length), and add new records to the end of the data set. Records cannot be physically deleted because that would change the entry sequence of the data set (RBAs of the records).

The discussion of free space in a key-sequenced data set pointed out that keyed insertion, deletion, or update (length changing) of records can change their RBAs. Therefore, to use addressed access to process a key-sequenced data set, you may have to keep track of RBA changes. VSAM passes back the RBA of each record retrieved, added, updated, or deleted.

Addressed Retrieval

Positioning for addressed sequential retrieval is done by RBA rather than by key. When a processing program opens a data set for addressed access, VSAM is positioned at the first record in the data set in entry sequence to begin addressed sequential processing. A POINT positions VSAM for sequential access beginning at the record whose RBA you have indicated. A sequential GET causes VSAM to retrieve the data record at which it is positioned and positions VSAM at the next record in sequence.

With direct processing, you may optionally specify that GET position VSAM at the next record in either a forward or backward direction. Your program can then process the following records sequentially in the desired direction.

Addressed sequential access retrieves records in a forward or backward direction. If addressed sequential retrieval is used for a key-sequenced data set, records will not be in their key sequence if there have been control interval or control area splits.

Addressed direct retrieval requires that the RBA of each individual record be specified, since previous positioning is not applicable. The address specified for a GET or a POINT must correspond to the beginning of a data record; otherwise, the request is invalid.

Addressed Storage

VSAM does not insert new records into an entry-sequenced data set, but adds them at the end. With addressed access of a key-sequenced data set, VSAM does not insert or add new records.

A PUT macro instruction stores a record. A PUT for update following a GET for update stores the record that the GET retrieved. To update a record, you must previously have retrieved it for update. You can update the contents of a record with addressed access, but you cannot alter the record's length. Neither can you alter the prime key field of a record in a key-sequenced data set.

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.

Addressed Deletion

The ERASE macro can be used only with a key-sequenced data set to delete a record that you have previously retrieved for update.

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.

What Are the Master Catalog and User Catalogs For?

A master catalog is required with VSAM, and any number of user catalogs are optional. Almost everything that is true of the master catalog is true of user catalogs, but user catalogs have special uses and there are significant differences between the VS1 and VS2 catalogs that we will discuss after we consider the general functions of a VSAM catalog.

A VSAM Catalog's Use in Data and Space Management

VSAM catalogs are a central information point for all VSAM data sets and the direct-access storage volumes containing them. The information describing a volume and the data sets on it is extensive enough to enable VSAM to allocate and deallocate data sets on the volumes without the volumes being mounted on a device of the system. The catalogs also provide VSAM with information needed to authorize access to data sets, compile usage statistics on them, and relate RBAs to physical locations. Defining a VSAM data set automatically builds the appropriate catalog entry containing all the necessary information.

All VSAM data sets on a volume must be cataloged in the same VSAM catalog, and that catalog must be the one that owns the volume. This may be either the master catalog or a user catalog. A VSAM data set has an entry in only one catalog. See "How is Access Method Services Used?" in the chapter "Communicating with VSAM" for a description of how volume ownership is relinquished.

Information Contained in the Records of a Catalog

Besides data set records, a VSAM catalog has records describing direct-access volumes in terms of the allocation of data spaces and the location of available space. VSAM can allocate and deallocate space on cataloged volumes that are not mounted. However, when allocating space to a data set, if there is not sufficient space available in the data space or data spaces on a volume, you must use the Access Method Services DEFINE space command to get the additional space the data set needs.

Information in a Data Set Record

Data set records provide the information required to make the connection between a data record's RBA and its physical location in terms of a storage volume's physical attributes. Besides the type of storage device and a list of volume serial numbers, a VSAM catalog keeps other data set information, including:

- A pointer to the location of each extent of the data set
- Statistics on the results of operations performed on the data set and its records, such as the number of insertions and deletions and the amount of free space remaining
- Attributes of the data set determined when it was defined, such as control-interval size, physical record size, number of control intervals in a control area, and, for a key-sequenced data set, location of the key field
- Password protection information
- An indication of the connection between: the index and the data components of a key-sequenced data set; the index and data components of an alternate-index cluster; the alternate index and the base cluster of a path; and an alternate-index upgrade set and its base cluster
- Information used to determine whether a key-sequenced data or index component has been processed without the other
- Information about the volume(s) on which the data set is stored

Information in a Volume Record

Volume information in a VSAM catalog provides the information required to keep track of data spaces and free storage areas. A VSAM catalog contains this sort of volume information:

- The volume serial number and device characteristics
- The location of data spaces on a volume
- The location and size of free areas available for allocation to data sets

From this information, you can derive:

- The count of data spaces and data sets on a volume
- The location of data sets within data spaces on a volume
- An indication of the data spaces associated with a data set

The Special Uses of User Catalogs

User catalogs can improve VSAM reliability and facilitate volume portability.

Improving Reliability

User catalogs are useful for improving reliability. By putting the catalog information of some of your data sets and storage volumes into user catalogs, you decentralize control, allow for the partitioning of applications, and at the same time achieve increased reliability.

Moving Volumes from One Operating System to Another

Because all VSAM data sets must be cataloged, moving a volume from one operating system to another requires that catalog information describing the volume and the data sets on it be moved along with the volume.

If you want to be able to move a volume or volumes from one OS/VS system to another, or from an OS/VS system to a DOS/VS system, define a user catalog on one of the volumes and define the volumes and the VSAM data sets on them in the user catalog. You can then transport the volumes by demounting them and removing them from the first system, taking them to the second system, and remounting them. You use Access Method Services to disconnect the user catalog from the master catalog of the first system and to connect a pointer to it in the master catalog of the second system. Any number of user catalogs can be used in this way.

You can also move individual data sets from one system to another by using Access Method Services, but the use of user catalogs for single volume portability is the most convenient way to achieve data set portability. For additional information, see "Moving Data Sets from One Operating System to Another" in the chapter "Communicating with VSAM."

How Does the VS1 Master Catalog Differ from the MVS Release 3 Master Catalog?

In OS/VS1, the system contains a VSAM master catalog as well as a system catalog. In MVS Release 3, the system catalog is a VSAM catalog that also serves as the VSAM master catalog. The differences are important, particularly in the areas of naming conventions and search strategies.

The VS1 VSAM Master Catalog

The VS1 system catalog points to the VSAM master catalog, which can contain catalog entries for VSAM and nonVSAM data sets (except for those belonging to generation data groups) and pointer entries for any number of optional user catalogs. Figure 13 illustrates how data sets can be cataloged among the system catalog, the master catalog, and user catalogs.

VSAM catalogs are searched before the system catalog, for VSAM data sets and data sets of other access methods. When you execute a program to process a data set, the order in which the catalogs are searched is:

1. Any user catalog or catalogs specified for the job step.
2. Any user catalog or catalogs specified for the job when none is specified for the job step.
3. The master catalog.
4. The system catalog.

Use caution in naming your data sets. Because the VSAM catalog is always searched first, it is possible to lose access to a data set cataloged in the system catalog if it has the same name as a data set in the VSAM catalog.

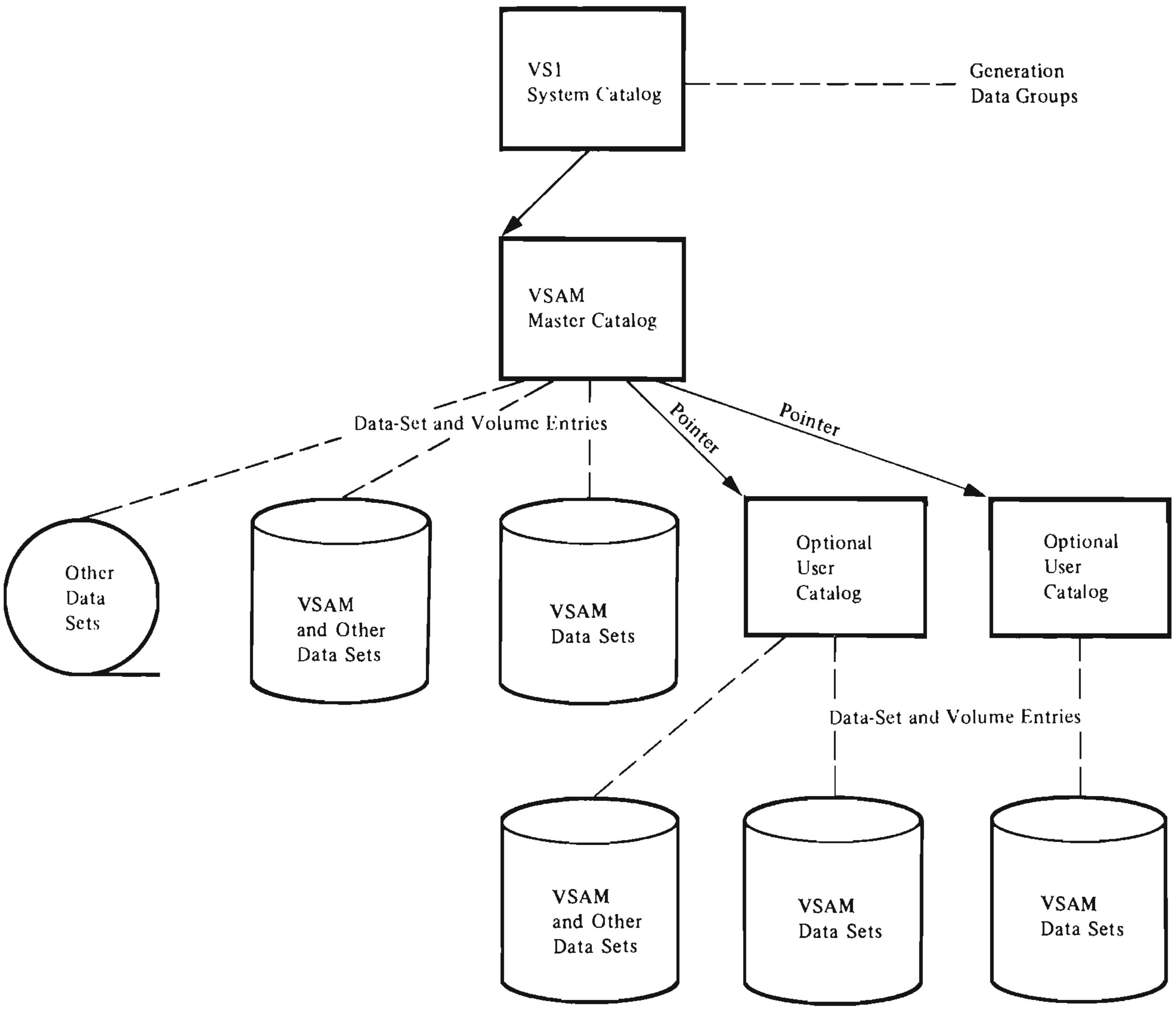


Figure 13. Cataloging VSAM and NonVSAM Data Sets in a VSAM Catalog

The MVS Release 3 Master Catalog

In MVS Release 3, the system catalog is the VSAM master catalog. It can handle both OS and VSAM data sets. As in OS, you can gain access to only one catalog per system at system initialization, and that catalog, called the master catalog, must contain entries for all the system data sets. Figure 14 compares the OS system catalog and the MVS master catalog.

The master catalog is established at system generation time, and without it, you can't define user catalogs, data spaces, or data sets. The volume on which the master catalog is defined must be permanently mounted.

The master catalog can contain pointers to OS catalogs (CVOLs), VSAM and other data sets, optional VSAM user catalogs, and generation data groups in nonVSAM data sets. Figure 15 illustrates how data sets and catalogs might be arranged within a basic MVS catalog structure.

OS System Catalog	MVS Master Catalog
Contains only OS data set entries	Contains entries for OS data sets, VSAM data sets, VSAM user catalogs, and OS CVOLS
One catalog during initialization	One catalog during initialization
Must be on IPL volume	Need not be on IPL volume
System controls the search strategy	You control the search strategy
All catalogs named SYSCTLG	All catalogs can be named by the user

Figure 14. Comparison of the OS System Catalog and the MVS Master Catalog

You cannot move the master catalog from one system to another by using Access Method Services commands. You can, however, make a copy of the master catalog, then move the copy to another system. If your master catalog is recoverable, you can use Access Method Services commands to repair damaged entries that result from system failure.

A master catalog cannot be used simultaneously as the VSAM master catalog for two MVS systems (that is, the master catalog cannot be shared between two systems as each system's master catalog). However, one system's master

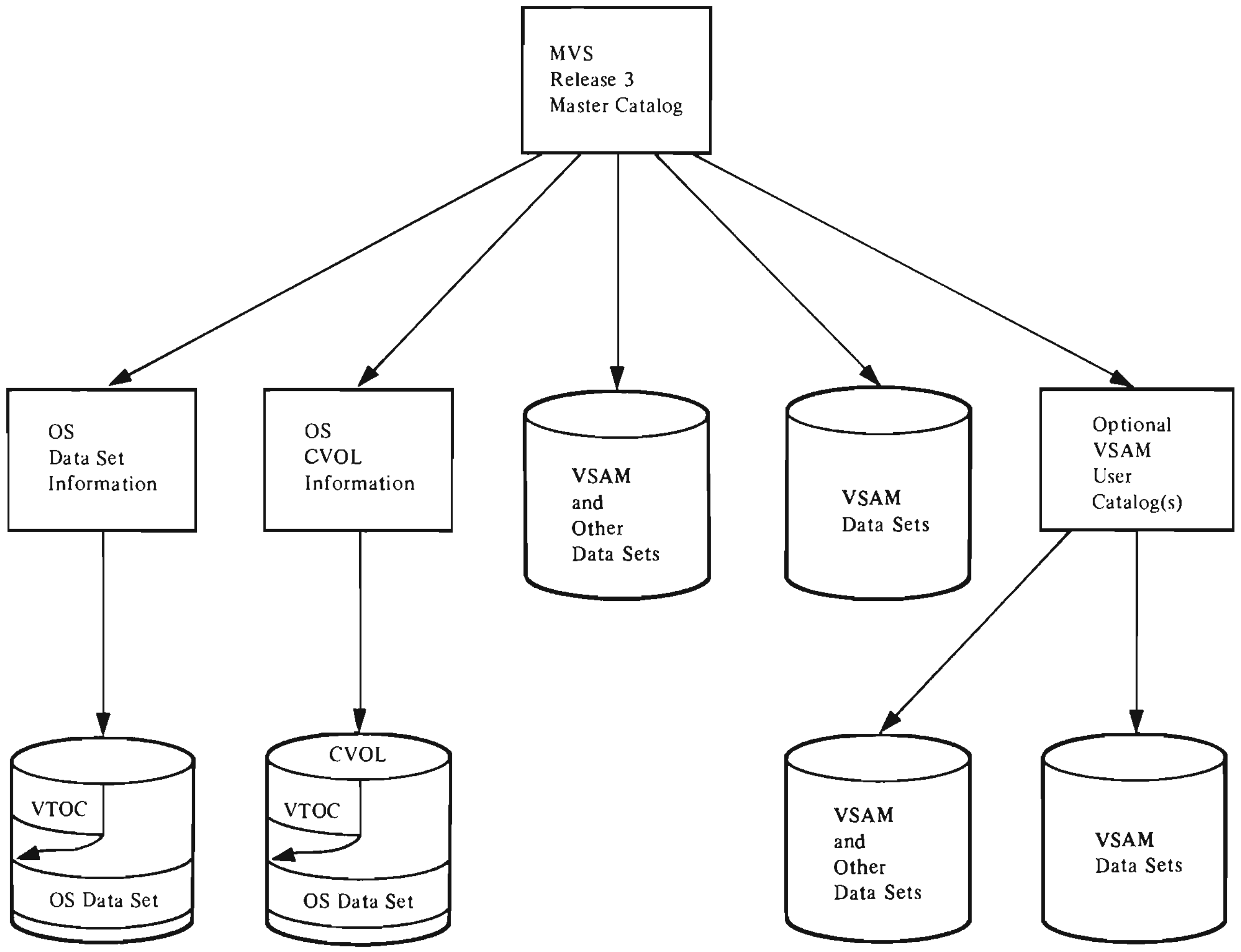


Figure 15. Cataloging VSAM and NonVSAM Data Sets in the MVS Master Catalog

catalog can be used as a user catalog on another MVS system that includes VSAM when the catalog is on a shared direct-access device. When a catalog is used simultaneously by two or more systems, all of the catalog's volumes must be on shared direct-access devices. If you do this, take care to assign passwords to each of the catalog's page spaces and system data sets to prevent their accidental or unauthorized use.

When you define a catalog, you can use JCL to cause the volume on which the catalog is to reside to be mounted or you can rely on dynamic allocation. For information and examples of how JCL and dynamic allocation are used to acquire resources, see *OS/VS2 Access Method Services* and *OS/VS2 JCL.*

In MVS, alias names can be assigned to a nonVSAM data set entry, a catalog connector entry (CVOL), and a user catalog. Such an entry contains a pointer to the beginning of a chain of alias entries. Each alias entry contains three pointers: one to the nonVSAM or CVOL entry, one to the next alias entry, and one to the previous alias entry.

When you execute a program to process a data set, the order in which the catalogs are searched is:

1. Any user catalog or catalogs specified for the job step.
2. Any user catalog or catalogs specified for the job when none is specified for the job step.
3. The master catalog, unless the data set is qualified (contains periods) and the qualifier (the characters up to the first period) is the name or the alias name of a catalog. In that case, that catalog is searched rather that the master catalog.

Because of this order of search, a qualified data set name and an unqualified data set name cannot exist in the same catalog, if the unqualified name is the same as the first qualifier of the qualified data set name. For example, the master catalog could not contain the data set 'ABC.123' and an alias 'ABC' for a CVOL or user catalog.

How Is the VSAM Catalog Structured?

A VSAM catalog is structured as a VSAM key-sequenced cluster with two key ranges. Like a VSAM key-sequenced cluster, the catalog consists of an index component and a data component. The space the catalog occupies is divided into fixed-length control intervals of 512 bytes each. Figure 16 illustrates the parts of the catalog described in the following paragraphs.

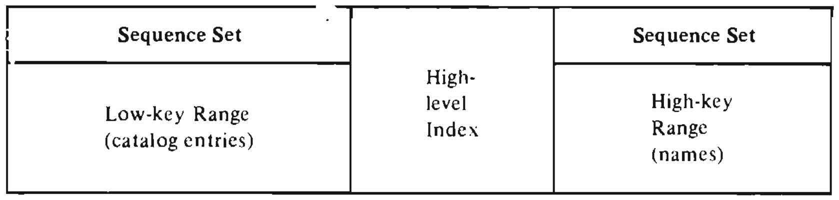


Figure 16. Catalog Structure

The Data Component

The catalog's data component is divided into two areas: a high-keyrange and a low-keyrange. One high-keyrange record and one low-keyrange record (plus, possibly, one or more extension records in the low key-range) exist for each cataloged object:

- A high-keyrange record contains the full 44-byte name of the cataloged object and the 3-byte control interval number of the object's low-keyrange record. Each high-keyrange record is 47 bytes long; up to 10 records can exist in a control interval in the high-keyrange.
- A low-keyrange record contains the information necessary to describe and locate the cataloged object. Each low-keyrange record is 505 bytes long; only one record exists in a control interval in the low-keyrange. Low-keyrange records are described in detail in *OS/VS2 Catalog Management Logic.*

The high-keyrange and low-keyrange also contain records that describe the catalog itself.

Control intervals that contain data records are grouped into control areas. A control area consists of control intervals that contain high-keyrange records or low-keyrange records; high-keyrange records do not exist in the same control area with low-keyrange records.

The Index Component

The catalog's index component consists of 505-byte index records, one per control interval. Control intervals containing index records are not grouped into control areas.

Each index record consists of one or more variable-length index entries. Each entry contains a compressed key value and a pointer to a lower level in the catalog (that is, a control interval in either a lower index level or in one of the data-keyranges). A compressed key is the 44-byte cataloged object's entryname, minus characters from the front and back of the entryname that are not needed to distinguish the key value (entryname) from the preceding and following key values. (See *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications* for details on indexes, index processing, and key compression.)

The index has two parts—an index set and a sequence set:

- The sequence set is the lowest level of the index. A sequence-set record exists for each control area (that is, for each group of data control intervals). The sequence-set record contains an entry for each control interval in the control area. Each entry contains a pointer to a control interval and the control interval's highest key value (compressed entryname).
- The index set is all levels of the index higher than the sequence-set level. An index-set record contains entries that point to lower-level index-set records, or that point to sequence-set records. Each entry contains a pointer to the lower-level record and the highest key value (compressed entryname) of the keys in the lower-level record. Since a sequence-set record contains the highest key value in each control interval within a control area, an entry in the next higher level record in the index set contains the highest key value of objects represented within a control area.

Size of a Control Area and Location of the Sequence Set

Because a sequence-set record contains one entry for each control interval in the control area, the size of the control area is limited by how many entries can fit in a sequence-set record. The length of an entry varies (because of the variable-length compressed entryname), but approximately 40 entries can fit in a sequence-set record. In addition, a control area exists as a whole number of contiguous tracks on a direct-access device. A catalog's control interval does not span tracks. Therefore, the size of a control area depends on the number of entries that will fit into a sequence-set record, and on the type of direct-access device:

- For a 3330, 3330 Model 11, or 2305 Model 2, a control area is two tracks and contains 40 data control intervals (20 control intervals per track).
- For a 3350, a control area is two tracks and contains 54 data control intervals (27 control intervals per track).
- For a 2305 Model 1, a control area is three tracks and contains 45 data control intervals (15 control intervals per track).
- For a 3340/3344, a control area is four tracks and contains 48 data control intervals (12 control intervals per track).
- For a 2314/2319, a control area is four tracks and contains 44 data control intervals (11 control intervals per track).

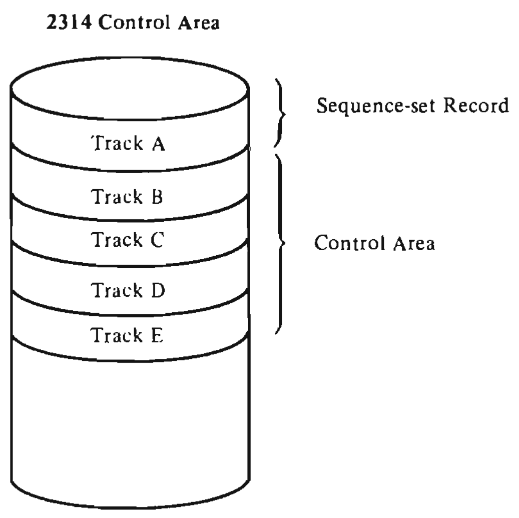
The track preceding each control area contains the control area's sequence-set record. The sequence-set record's track is also divided into control intervals (the number of control intervals per track depends on the device type). The sequence-set record is replicated (that is, it is copied in each control interval on the track) to improve performance by reducing the amount of rotational delay required when it is accessed. Figure 17 shows how the control area and sequence-set record tracks are related.

Allocation of Catalog Space

When a catalog is created, it is built in the first data space on the volume. When you define the catalog, you can specify an exact amount of space for it or you can allow Access Method Services to determine the total amount of catalog space. When Access Method Services determines the total amount of catalog space, you specify the amount of space necessary for the catalog entries (shaded part of Figure 18) and Access Method Services determines how much space is needed for the rest of the catalog. See "How Space Is Assigned to a Catalog" in the chapter titled "Creating and Cataloging Objects" for more details.

Catalog space is allocated as an amount of tracks (even when you specify an amount of cylinders or records when you define the catalog). The amount of tracks is a multiple of allocation units. An allocation unit is the size of a data control area plus one track (the adjacent track for the control area's sequence-set record). Therefore, the size of an allocation unit is:

- Three tracks for a 3330, 3330 Model 11, 3350, or 2305 Model 2 (two tracks for the control area and one track for the sequence-set record).
- Four tracks for a 2305 Model 1 (three tracks for the control area and one track for the sequence-set record).
- Five tracks for a 2314/2319 or 3340/3344 (four tracks for the control area plus one track for the sequence-set record).



Each track (on a 2314) contains 11 control intervals:

- **On Track A, the control intervals contain a replicated sequence-set record.**
- **On Tracks B, C, D, and E, each control interval contains up to ten high-keyrange records or one low-keyrange record.**

Figure 17. Physical Relationship of a Sequence-Set Record and the Control Area It Governs (on a 2314 Direct-Access Storage Device)

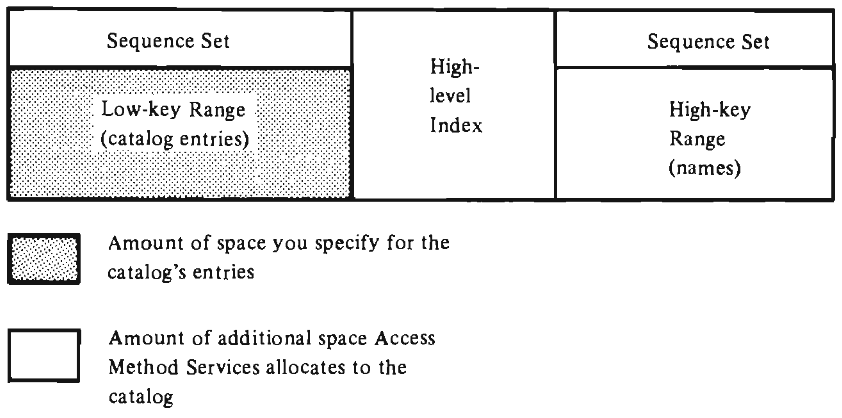

Sequence Set
Low-key Range (catalog entries)
High-level Index
Sequence Set
High-key Range (names)
Amount of space you specify for the catalog's entries
Amount of additional space Access Method Services allocates to the catalog

Figure 18. Space Allocation for the Catalog

When the amount of space you specify (in the DEFINE command) is not a multiple of the device's allocation unit size, the amount of space is rounded downward to a multiple and the additional tracks aren't allocated.

One allocation unit is always allocated to the index-set part of the index. Ten percent of the remaining allocation units (or at least one allocation unit) is allocated to the high-keyrange and its sequence-set record(s). The remaining allocation units are allocated to the low-keyrange and its sequence-set record(s). The minimum catalog size is three allocation units: one allocation unit each for the index set, the high key-range, and the low-keyrange.

For example, when you define a catalog you specify 1 cylinder of space on a 3330 direct-access device for the entire catalog (1 cylinder = 19 tracks):

- The request is rounded down to 18 tracks, which is six allocation units of three tracks each (the nineteenth track is not used and not available unless the catalog's data space is suballocated).
- One allocation unit (three tracks) is assigned to the catalog's index set (that is, the high-level part of the catalog's index).
- Ten percent of the remaining 15 tracks is to be allocated to the catalog's high-keyrange. However, since the minimum (one allocation unit, or three tracks) is larger than ten percent of the remaining tracks, or 1.5 tracks, three tracks are allocated to the catalog's high-keyrange. The first track contains a sequence-set record (occupying one control interval) that is replicated around the track. The next two tracks (one data control area), containing 40 control intervals, are available for a maximum of 400 high-keyrange records. In other words, the high-keyrange can point to 400 catalog entries before it must be extended (a catalog entry, however, might include more than one catalog record, or control interval, in the low-keyrange).
- The remaining space available to the catalog, 12 tracks or four allocation units, are assigned to the catalog's low-keyrange. Each allocation unit includes one track for the replicated sequence-set record and one data control area (two tracks containing 40 control intervals). Therefore, 8 tracks containing 160 control intervals (that is, four data control areas) are available for low-keyrange records. The catalog's self-describing entries usually occupy the first 13 catalog records (actually, the number of catalog records for the catalog's self-describing entries varies from 12 to 15, depending on the type of device containing the catalog).

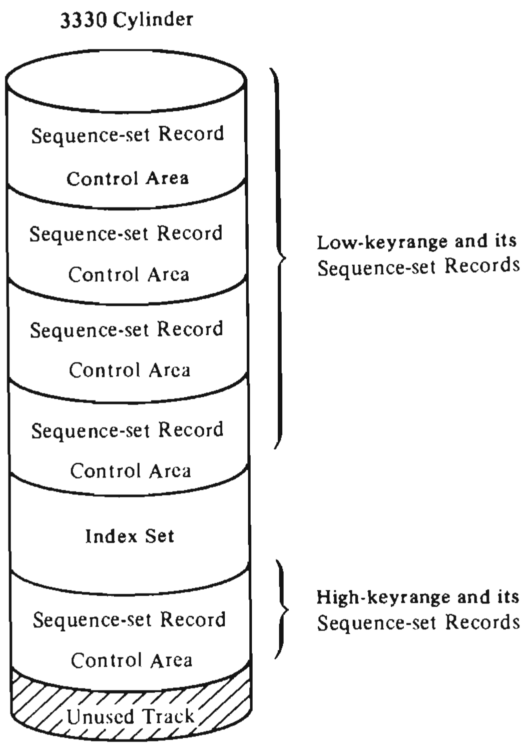
  The catalog's self-describing entries are pointed to by two high-keyrange records: one contains the catalog's entryname, the other contains the catalog volume's serial number.

Figure 19 illustrates the allocation of one cylinder of space for the catalog described in the above example.

Utilization of Catalog Space

Because the index-set, sequence-set, and low-keyrange control intervals each contain one (and only one) catalog record, the space allocated to these parts of the catalog can be completely filled before a secondary allocation of space is required. However, the catalog's high-keyrange control intervals can contain up to 10 records each, and are subject to control interval splitting and control area splitting:

- Control interval split: When VSAM Catalog Management (that is, the part of VSAM that manipulates catalog records) tries to add a record to a high-keyrange control interval that contains 10 records, the control interval will be split into two control intervals. Five of the control interval's records are moved into an unused control interval, so that after the split each control interval contains five high-keyrange records. The new record is then added to the appropriate control interval. Each control interval now has sufficient free space to accept more high-keyrange records, but the free space might not be used until entries with the appropriate key values are added to the catalog.



- Each allocation unit = 3 tracks.
- Each sequence-set record occupies one track (the sequence-set record is contained in one control interval, and is replicated around the track).
- Each control area = 2 tracks = 40 control intervals.

Figure 19. Allocation of a One-Cylinder Catalog on a 3330 Direct-Access Storage Device

- Control area split: When all control intervals in one of the high-keyrange's control areas are in use (that is, the control area no longer contains any free control intervals to use when a control interval split occurs) and a control interval must be split, then the control area is split. Note that, before the control area is split, all of its control intervals except one might be only half-full. When the control area is split, half of the control intervals are moved into an unused control area. The appropriate control interval is split and the high-keyrange record is added in key sequence to the appropriate control interval. After the control area is split, then, only 25 percent of the two control areas might contain data records (although this is a worst-case example; it illustrates how space might be under-utilized in the catalog's high-keyrange).

Whenever a control interval or control area is split, the appropriate sequence-set and index-set records are updated or split to reflect the new data structure.

For example, suppose you want to create a VSAM user catalog that contains the entries from an OS CVOL. On a 3330 direct-access device, two allocation units are assigned to a catalog's high-keyrange and its associated sequence-set records. Since a 3330 track contains 20 control intervals and up to 10 high-keyrange records can exist in a control interval, the high-keyrange can potentially contain 800 records (2 allocation units x 2 tracks per allocation unit x 20 control intervals per track x 10 records per control interval). After the catalog is created, you issue the CNVTCAT command to convert the OS control volume's (CVOL's) entries to VSAM catalog entries. The CVOL's entries are converted one at a time and are added to the catalog in ascending key sequence (the catalog was empty before the CNVTCAT command was issued). When a control interval must be split, the record causing the split will be added to the control interval with the higher keys (that is, with the entrynames of higher key sequence). The following splitting occurs:

- As each control interval in the first control area becomes full and another entry is added, the control area is split. Since the CNVTCAT command adds entries to the VSAM catalog in ascending key sequence, each high-keyrange control interval contains five records after the split. When the next-to-last control interval is split, there are almost 200 records in a control area that might contain 400 records.
- When the last control interval in the first control area becomes full and another entry is added, the first control area must be split. Half of the control intervals in the first control area are moved into the second control area. The first control area now contains 100 records and, because the OS CVOL's entries are converted and added to the user catalog in ascending key sequence, no further records are added to the first control area.
- When control intervals in the second control area become full, they are split as entries are added. Each control interval in the second control area thus contains five high-keyrange records (except the current control interval, which might contain more than five until it is split). The second control area can contain a maximum of 205 records before it must be split (that is, 5 records in all control intervals except the last one (50 percent utilization), and 10 records in the last control interval (100 percent utilization)).

In this example, 305 entries were converted from the OS CVOL and added to the VSAM catalog. The catalog's high-keyrange must be extended to allow any further OS CVOL entries to be converted and added in ascending key sequence. If the catalog was defined without a secondary space allocation amount specified, the CNVTCAT command will terminate with a message to the user that indicates the catalog is full. However, the catalog's high-keyrange now contains ample free space to allow you to add entries to it in a random sequence (that is, entries that are added to the catalog as a result of DEFINE commands).

This example illustrates the importance of specifying a secondary space allocation amount when you define your catalog, especially when you intend to use the catalog to receive converted entries from an OS CVOL.

COMMUNICATING WITH VSAM

This chapter introduces programmers to communicating with VSAM by using the commands of Access Method Services, the macros for connecting a processing program to a data set and gaining access to it, the macros used in data base/data communication (DB/DC) applications, and the JCL parameters affected by VSAM. An application programmer doesn't need to know the format of control blocks, as he does with some other access methods: he just specifies the name of the action he wants.

How is Access Method Services Used?

Access Method Services is a multifunction service program that you use to define a VSAM data set and load records into it, build an alternate index, convert OS catalogs to VSAM master or user catalogs, convert a sequential or an indexed sequential data set to the VSAM format, list VSAM catalog records or records of a data set, copy a data set for reorganization, create a backup version of a data set, recover from certain types of damage to a data set, and make a data set portable from one operating system to another.

You tell Access Method Services what to do by giving a command and descriptive parameters through an input job stream or by calling it in a processing program and passing it a command statement. In OS/VS2 you can also execute Access Method Services from a TSO (Time Sharing Option) terminal, either by executing a program that calls it, by executing it directly and giving commands and parameters through an input data set, which can come from a TSO terminal, or by entering one of the TSO commands that is identical to Access Method Services commands. For more information about the use of TSO with VSAM, see "How Can the Time Sharing Option (TSO) Be Used with VSAM?" in the chapter "Preparing for VSAM."

A set of conditional statements (IF, ELSE, DO, END, SET) allows you to alter the sequence of execution of a series of commands by testing or resetting codes that Access Method Services sets to indicate the completion status of each command.

There are commands and groups of commands in Access Method Services for:

- Defining and deleting data sets and listing catalog records
- Building alternate indexes
- Copying and listing data sets
- Moving data sets from one operating system to another
- Recovering data
- Converting OS catalogs to VSAM catalogs

Complete descriptions of all the Access Method Services commands are in *OS/VS1 Access Method Services* and *OS/VS2 Access Method Services.*

Defining and Deleting Data Sets and Listing Catalog Entries

You must use Access Method Services to define all VSAM data spaces, data sets, indexes, and catalogs. It makes entries for them in a VSAM catalog and allocates space for them. Four commands enable you to define data sets, alter the definitions, allocate and free auxiliary-storage space, and list catalog records: DEFINE, ALTER, DELETE, and LISTCAT.

DEFINE: Defining a Data Set and Allocating Space

To define a catalog, data space, key-sequenced data set, entry-sequenced data set, relative record data set, reusable data set, alternate index, path, and (in VS2) a generation data group and an alias, you specify the DEFINE command, the object to be defined, and, optionally, the catalog that is to contain an entry defining it. You also use DEFINE to catalog data sets of other access methods in a VSAM catalog.

There are parameters for specifying initial auxiliary-storage allocation, amount of space for extensions, erasure of data in a deleted data set, passwords and other authorization information, size and other attributes of data records, minimum amount of virtual-storage space for I/O buffers, percentages of free space in control intervals and control areas of a key-sequenced data set and other performance options, retention period, name of module to receive control if processing error occurs, identification of the owner of the data set defined in the entry, how a data set can be shared across regions or systems, data-set preformatting options, and whether write operations are to be verified.

You specify the amount of auxiliary-storage space for the object you are defining as the number of data records that the object is to contain or as a number of physical units, such as tracks or cylinders. Specifying the number of records, independent of type of storage device, leaves the calculation of the number of physical units of space up to VSAM. It calculates the size of the control interval and control area to be used. You may specify the control-interval size, and VSAM will use it so long as the size falls within the acceptable limits that VSAM calculates.

You can define a catalog with a recovery attribute that makes it possible to recover from certain types of catalog damage. See the chapter "Protecting Data with VSAM" for further information.

When you define a key-sequenced data set, you may specify that its space is to be allocated on volumes according to ranges of key values. The space for each range is extended separately on the same volume when additional space is required; otherwise on a candidate volume.

For convenience, you may specify an existing catalog entry as a model for a new entry, so long as they are of the same type (entry-sequenced data set, key-sequenced data set, alternate index, path, or user catalog). The information in the model will be used in the new entry unless you override it.

ALTER: Modifying a Catalog Entry

Many of the attributes that you define, either explicitly or by default, when you create a catalog record may be modified subsequently by way of the ALTER command, most of whose parameters are the same as the DEFINE parameters. You can change the name of a data set, the key position and record size of an empty VSAM data set, the indication of whether to erase the data in a deleted data set, passwords and other authorization information, minimum amount of virtual-storage space for I/O buffers (which you may increase, but not decrease), percents of free space in new control intervals and control areas of a key-sequenced data set, retention period, *name* of the owner of a data set, the indication of how to share a data set, the indication of whether to verify write operations, and the indication of whether VSAM is to maintain an alternate index. Only empty alternate indexes can be changed to UPGRADE.

Certain attributes of the data set, such as control-interval size and placement of the index in auxiliary storage relative to a key-sequenced data set, cannot be modified. Changing these attributes amounts to a reorganization of the data set and requires that you define a new data set and copy the old data set into it.

You can use the ALTER command to remove a damaged volume. VSAM removes all VSAM data spaces from the volume, rewrites the volume's VTOC, and relinquishes ownership of the volume. NonVSAM data sets on the volume and the catalog that owns the volume are not affected. This function should be used only when you can't gain access to the catalog that owns the volume.

DELETE: Removing a Catalog Entry and Freeing Space

The DELETE command enables you to remove the entry for any previously defined object and, in effect, cause it to cease to exist. The space is freed for use by new objects and, if the erase option is specified in the entry or in the command, overwritten with binary 0s.

You must use Access Method Services to delete data spaces, data sets, indexes, and catalogs: you cannot delete them by way of the JCL disposition parameter or operating-system utilities. Deletion of an alternate index causes its data and index component and the related path to be deleted, deletion of a base cluster causes all related alternate indexes and paths to be deleted, and deletion of a path causes no other object to be deleted.

You can also use the DELETE command to force the deletion of VSAM volumes and catalogs, regardless of their contents. When a volume is specified, all VSAM data spaces are removed from the VTOC and VSAM relinquishes ownership of the volume. In addition, all catalog entries that refer to this volume are marked unusable for all purposes except deletion. You cannot force the deletion of a volume that contains a VSAM catalog. When a VSAM catalog is specified, all the VSAM data spaces are removed from all the volumes owned by the catalog. An active master catalog cannot be deleted this way.

The catalog relinquishes ownership of a volume when no nonempty data spaces remain on the volume following a DELETE SPACE command. In the case of data sets with the unique attribute, the catalog does not relinquish ownership until each data set has been deleted, followed by a deletion of the data spaces.

In VS2 systems only, the DELETE command can be used to remove VSAM entries from a VSAM catalog when a volume owned by the catalog has lost its VSAM space or is inaccessible. Loss of data can be caused by physical damage to the volume, an improper restore, or a volume cleanup operation (see ALTER command, above). This function should be used only when you cannot gain access to the volume.

LISTCAT: Listing Catalog Entries

The LISTCAT command enables you to list individual entries, all entries of a particular type (cluster, alternate index, path, etc.), or all entries of a given catalog. You see the entire entry, except that passwords in an entry are not listed unless you specify the master password for the data set defined by the entry or the master password for the catalog itself.

Building an Alternate Index

The BLDINDEX command is used to build an alternate index over a single base cluster. A base cluster can have more than one alternate index.

BLDINDEX: Building an Alternate Index

One or more alternate indexes can be built over a defined, nonempty key-sequenced or entry-sequenced data set. The alternate index must have been previously defined. If the data set is nonempty and defined with the reusable attribute, BLDINDEX will reuse it.

There are parameters for specifying the names and passwords of the base cluster and alternate index, job control statements to be used when external sorts are performed, and the name and password of the catalog in which sort work files will be defined.

You can supply more than one alternate index name and build additional alternate indexes at the same time.

Copying and Listing Data Sets

The REPRO and PRINT commands enable you to copy and list sequential, indexed sequential, and VSAM data sets.

REPRO: Converting and Reorganizing Data Sets

The REPRO command instructs Access Method Services to get records from a sequential, indexed sequential, or VSAM data set and put them into a sequential or VSAM data set. You may use it to convert an indexed sequential data set to a key-sequenced data set with an index. First, define a new key-sequenced data set and its index. Then copy the indexed sequential data set into the key-sequenced data set. Access Method Services converts data records to the VSAM format and builds an index.

You can reorganize an old data set by copying it into a newly defined data set of the same type. With key-sequenced data sets, you can optionally specify different percents of distributed free space and different performance options for the new data set when you define it. Copying the old key-sequenced data set into the new one redistributes free space, makes the entry sequence of the data records the same as their key sequence, and builds a new index.

The data set into which records are copied may either be newly allocated (by way of the DEFINE command) or contain records already. Records copied into a key-sequenced data set are merged with any existing records; records are added at the end of an entry-sequenced data set. When copying into a relative record data set from a relative record data set, the records are placed in the same relative position as they were in the input data set. You may specify a range of records to be copied by number of records, by key or address in an indexed sequential or a key-sequenced data set, or by relative record number in a relative record data set.

The REPRO command also provides the catalog unload/reload function that is used to backup catalogs and recover from failures that make them inaccessible.

Catalog backup (unload) allows you to copy a VSAM catalog to a SAM (sequential access method) data set or to a VSAM key-sequenced or entry-sequenced data set. The copy, which preserves the contents of the catalog, cannot be used as a catalog. To recover (reload) a catalog, copy the unloaded version into an existing VSAM catalog. An existing catalog can be one you defined for this purpose, or it can be an earlier or later version of the unloaded catalog.

The greater the difference between the backup catalog and the active catalog, the more difficult it will be to regain access to all of your data. Therefore, you should make backup copies frequently. The closer your backup copy comes to matching the active catalog, the more successful any recovery operation will be. For additional information, see "Protecting the Catalog" in the chapter "Protecting Data with VSAM."

In OS/VS2, you can also use REPRO to copy one VSAM catalog to another.

PRINT: Listing Data Records

The PRINT command instructs Access Method Services to list some or all of the records of a sequential, indexed sequential, or VSAM data set or catalog in one of three formats: each byte as 2 hexadecimal digits, each byte as a single character, or a combination of these two, side-by-side. You may specify a range of records for listing as you do for copying.

Moving Data Sets from One Operating System to Another

We discussed volume portability between OS/VS systems and between OS/VS and DOS/VS systems in "The Special Uses of User Catalogs" in the chapter "Getting to Know What VSAM Is and Does." The EXPORT and IMPORT commands allow you to transport individual data sets between OS/VS systems or between OS/VS and DOS/VS systems. Figure 20 compares volume and data-set portability. Data portability is achieved by moving volumes or by moving individual data sets.

EXPORT: Extracting Catalog Information and Making a Data Set Portable

The EXPORT command instructs Access Method Services to copy an entry-sequenced data set, a relative record data set, or a key-sequenced data set and its index (other than a VSAM catalog) in the format of a sequential data set onto a storage volume to be transported to another operating system. The transporting volume may be magnetic tape or disk. Access Method Services also extracts information from the catalog entry that defines the object to be transported and copies it onto the transporting volume. The information is used to define the object automatically in a VSAM catalog in the other operating system.

EXPORT can be used to make a backup copy of a data set. Should that data set become inaccessible, you can use the IMPORT command to introduce the exported copy back into the system.

Exportation is either permanent or temporary. In permanent exportation, Access Method Services deletes the catalog record and frees the storage space; in temporary exportation of an object, both the sending and the receiving operating systems have a copy of it, and you may specify that one or both of the copies are not to be modified. A copy so protected can only be read. You may free the copy for full access with the ALTER command.

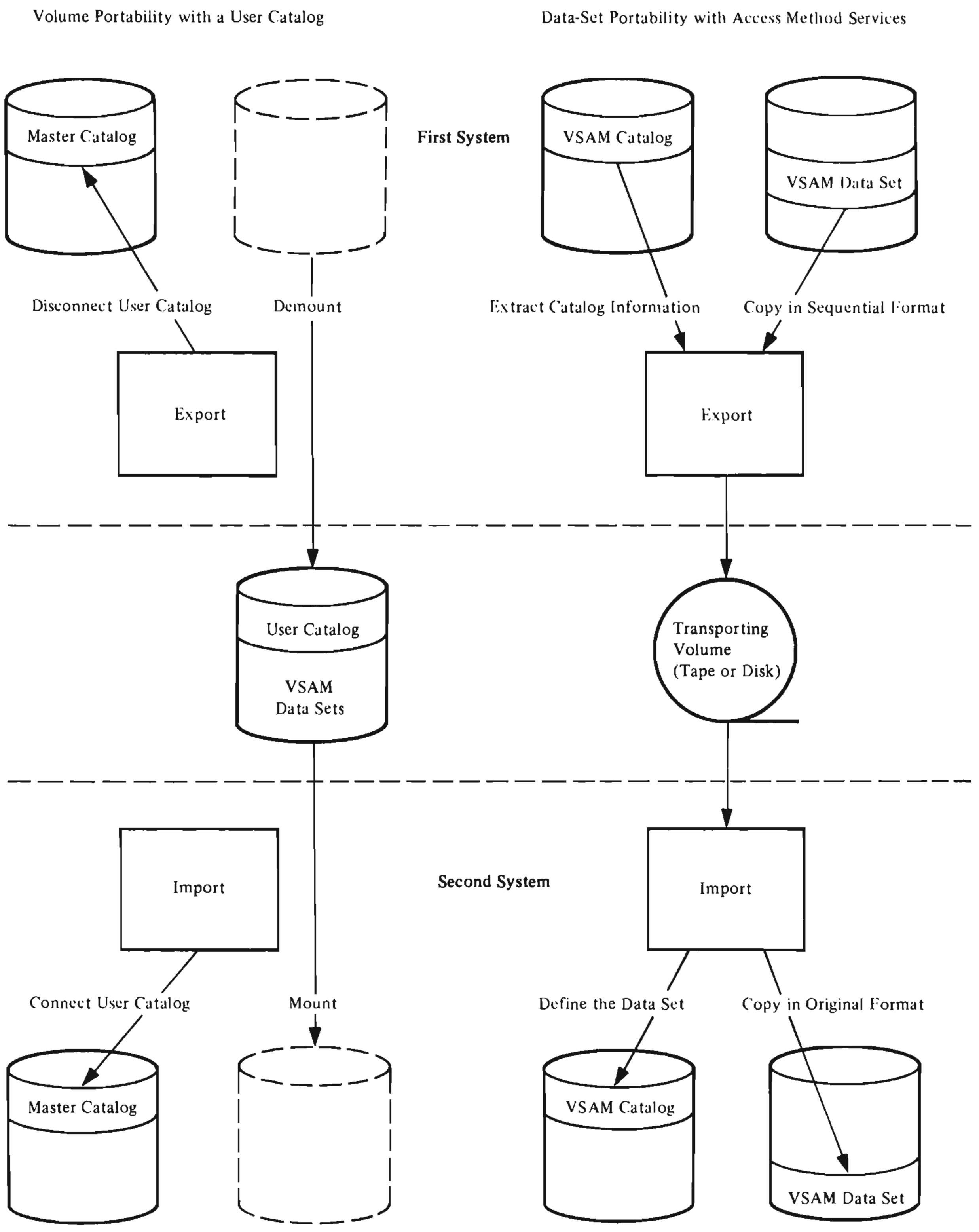


Figure 20. Comparison of Volume Portability and Data-Set Portability

You use EXPORT to disconnect a user catalog from a master catalog when you are moving the user catalog to another system. The user catalog is not copied, but remains on its original volume in its original form.

Paths are not exportable by themselves but are included in exports of alternate indexes or clusters; alternate indexes are exported as key-sequenced data sets. To permanently export a cluster and the alternate indexes associated with it, first export the alternate indexes and then the cluster.

IMPORT: Loading a Portable Data Set and Its Catalog Information

The IMPORT command instructs Access Method Services to define the entry-sequenced data set, the relative record data set, or the key-sequenced data set and its index on the transporting volume in the catalog that you specify, using the catalog information extracted in exportation. The object itself is stored in its VSAM format in a data space that is defined in the specified catalog.

You use IMPORT to define a pointer to a user catalog in the master catalog. The user catalog is not copied, but remains on its original volume in its original form.

You can use the EXPORT and IMPORT commands to prepare a backup version of an entry-sequenced data set and its catalog record, a key-sequenced data set, its index, and their catalog records, or a relative record data set and to load the backup copy if it is needed. When you import a backup copy, the catalog record is regenerated.

To import a cluster and the alternate indexes associated with it, first import the cluster and then the alternate indexes. IMPORT will automatically reestablish all the paths that existed when the data sets were exported.

Use the IMPORT command to introduce back into the system the backup data sets produced by the EXPORT command.

When exporting data sets from one device type and importing them to another device type, you can delete and redefine your data set with space parameters that are appropriate to the new device. The new data set must be empty. Also, it must be the same type of data set, and if indexed, it must have the same key length and position as the old data set. IMPORT uses this empty data set rather than defining a new one based on exported catalog information.

Recovering Data

Access Method Services provides the VERIFY, EXPORTRA, IMPORTRA, and LISTCRA commands to test and reestablish a data set's integrity, recover a cataloged object, reestablish objects in the catalog, and list the contents of a catalog recovery area.

VERIFY: Testing and Reestablishing a Data Set's Integrity

The end of a data set is indicated by an end-of-file indicator at the end of the data set and by information in the data set's catalog record. The end may be improperly indicated in the catalog if an error prevented VSAM from closing the data set. You can instruct Access Method Services to close the data set. It modifies the catalog information, if necessary, to correspond with the data set.

EXPORTRA: Exporting Objects Using the CRA

For data sets cataloged in a catalog that was defined with the recoverable attribute, critical catalog information is recorded in CRAs (catalog recovery areas) that are present on each owned volume. If the VSAM or nonVSAM data is not addressable via the catalog, the EXPORTRA command can be used to gain access to both the recovery area data and the VSAM data to create a copy of the data. The recovered information can be introduced back into the system by the IMPORTRA command.

The EXPORTRA command can be used to recover objects on the basis of multiple recovery areas (volumes), a single volume, or sets of individual data sets. EXPORTRA can also be used to recover nonVSAM objects and all VSAM objects except page spaces. To do a selective recovery, use the LISTCRA command to determine data set names and their associated volumes.

IMPORTRA: Reestablishing Objects in the Catalog

The IMPORTRA command can be used to reestablish in a VSAM catalog all the nonVSAM and VSAM objects (except page spaces) rendered portable by EXPORTRA.

Before you begin, you must provide a stable base in which IMPORTRA can operate. In particular, you should have available either a new or restored VSAM catalog. IMPORTRA requires that the volumes occupied by the exported VSAM data sets be available for mounting.

LISTCRA: Listing a Catalog Recovery Area

The LISTCRA command can be used to list the entire contents of a given catalog recovery area or to list those entries that differ from those in their associated catalog. The content of the list is either the data set names and volumes or a dump of the records.

LISTCRA can be used to determine what corrective actions are required.

Converting an OS Catalog into a VSAM Catalog

Access Method Services provides a command that converts entries in an OS catalog into entries in an existing VSAM master or user catalog.

CNVTCAT: Converting an OS Catalog into a VSAM Catalog

After the VSAM catalog that is to receive the converted entries has been defined, you issue the CNVTCAT command to convert OS catalog entries into VSAM catalog entries.

There are parameters that enable you to specify the names of the OS catalog to be converted, the VSAM catalog that is to receive the converted entries, and the master catalog into which any aliases for user catalogs are to be placed. If any of the catalogs that are to receive entries are protected by passwords, you must supply the update or higher level password. You can also indicate whether entries are to be listed after they are converted.

For additional information and coded examples of the CNVTCAT command, see *OS/VS1 Access Method Services* or *OS/VS2 Access Method Services.*

What Are the Macros for Processing a VSAM Data Set?

You code the VSAM macros in a processing program to gain access to your data. There are macros for:

- Connecting and disconnecting a processing program and a data set. These prepare a bridge for VSAM between the program and the data.
- Specifying parameters that relate the program and the data. These identify the data set and describe the kind of processing to be done.
- Manipulating the information relating the program and the data. These are used to specify changes in processing.
- Requesting access to a data set. These initiate the transfer of data between auxiliary and virtual storage.
- Gaining access to index control intervals.
- Sharing resources.

Connecting and Disconnecting a Processing Program and a Data Set

You use the OPEN macro to connect a processing program to a data set, so VSAM can satisfy the program's requests for data; you use CLOSE to complete processing and free resources that were obtained by the Open routine.

OPEN: Connecting a Processing Program to a Data Set

VSAM uses its own authorization routine and one that you have provided to verify a program's authority to process a data set.

Open constructs VSAM control blocks and, by examining the DD statement indicated by the ACB macro and the volume information in the catalog, calls for the necessary volumes to be mounted and checks whether each volume matches its catalog information. If you are opening a key-sequenced data set, an alternate index, or a path, Open checks for consistency of updates of primary index and data components. If the data set and its index have been updated separately, a warning message is issued to indicate a timestamp discrepancy.

CLOSE: Disconnecting a Processing Program from a Data Set

The Close routine completes any operations that are outstanding when a processing program issues a CLOSE macro for a data set. For instance, VSAM buffers index records and data records, so the contents of a control interval may need to be stored or an index record updated and stored.

Close updates the catalog for any changes in the attributes of a data set. The addition of records to a data set may cause its end-of-file indicator to change, in which case Close updates the end-of-file indicator in the catalog. These end-of-file indicators help ensure that the entire data set is accessible. If an error prevents VSAM from updating the indicators, the data set is flagged as not properly closed. When a processing program subsequently issues an OPEN macro, it is given an error code indicating the failure. For more information on correcting this condition, see the discussion of the Access Method Services VERIFY command and "Method of Indicating the End of a Data Set" in the chapter "Protecting Data with VSAM."

Close restores control blocks to the status that they had before the data set was opened and frees the virtual-storage space that Open used to construct VSAM control blocks.

You can issue a CLOSE macro (TYPE=T) to update the catalog. Processing may continue without reopening the data set.

Specifying Parameters That Relate the Program and the Data

To open a data set for processing, you must identify the data set and the types of processing to be done. You use the ACB macro to specify a data set you want to process and the types of access you want to use. The GENCB macro can be used in place of the ACB, EXLST, or RPL macro to generate processing specifications during the execution of a processing program, rather than during assembly or compilation of the program.

ACB: Defining the Access-Method Control Block

You use the ACB macro to define a control block for each data set that your processing program will gain access to. You give the name of the JCL DD statement that describes the data set, so the Open routine can connect the program to the data. If you use more than one ACB for a given cluster, VSAM optionally uses the same set of control blocks for all requests to the specified data set.

You can share control block structures among multiple ACBs by specifying the same ddname or by processing the same base data set and specifying the DSN option in the ACB macros.

The other information that you specify enables Open to prepare for the kind of processing to be done by your program:

- The address of a list of exit-routine addresses that you supply. You use the EXLST macro, described next, to construct the list.
- For processing concurrent requests, the number of requests that are defined for processing the data set. The control blocks for the set of concurrent strings you specify are allocated on contiguous virtual storage. If the number you specify is not sufficient, OS/VS will dynamically extend the number of strings as needed by concurrent requests for this ACB. Strings allocated by dynamic extension are not necessarily on contiguous storage.
- The size of the virtual-storage space for I/O buffers and the number of I/O buffers that you are supplying for VSAM to process data and index records. A minimum of two buffers is required for data control intervals for a single request for an entry-sequenced data set. A minimum of three buffers is required for a key-sequenced data set, two for data control intervals and one for index records. For concurrent requests that require VSAM to keep track of multiple positions in a data set, each additional request requires a minimum of one buffer for control intervals and one buffer for index records. For example, three concurrent requests requires a minimum of four buffers for control intervals and three buffers for index records. (These numbers do not apply to the shared resources option because those buffers are handled via buffer subpools.)
- The password that is required for the type of processing desired.

- The processing options to be used: keyed, addressed, or control interval, or a combination; sequential, direct, or skip sequential access, *or a* combination; retrieval, storage, or update (including deletion), or a combination; shared or nonshared resources.
- Address and length of an area for error messages from VSAM.

EXLST: Defining the Exit List

You use the EXLST macro to specify the addresses of optional exit routines that you may supply for analyzing physical and logical errors, end-of-data-set processing, noting RBA changes, and writing a journal. Any number of ACB macros in a program may indicate the same exit list for the same exit routines to do all the special processing for them, or they may 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 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.

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 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.

Writing a journal. To journalize the transactions against a data set, you may specify a journal routine, which VSAM exits to before moving your data to the control-interval buffer. 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 transmitting to auxiliary storage the contents of a control interval in which there is an RBA change.

RPL: Defining the Request Parameter List

The RPL macro defines the request parameter list, or the list of parameters required for a particular request for access. It identifies the data set to which the request is directed by naming the ACB macro that defines the data set.

You can use a single RPL macro to define parameters that apply to all of the requests (GET, PUT, POINT, and ERASE, described under "Requesting Access to a Data Set") for access to a data set. You use the MODCB macro (described following GENCB) to modify some of the parameters to change the type of processing. For example, you can change from direct to sequential or from update to nonupdate processing.

For concurrent requests that require VSAM to keep track of more than one position in a data set, you may use up to 255 RPL macros to specify requests that your processing program or its subtasks can issue asynchronously to gain access to the same data set concurrently. The requests can be sequential or direct or both, and they can be for records in the same or different parts of the data set.

You need specify only the RPL parameters appropriate to a given request, as follows:

Address of the next request parameter list in a chain. You can chain request parameter lists together to define a series of actions for a single GET or PUT. For example, each request 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 as a single request (chaining request parameter lists is not the same as processing concurrent requests that require VSAM to keep track of multiple positions in a data set).

Processing options for a request. A request is to gain access to a data record or a control interval. Access may be gained by address (RBA) or by key. Addressed access may be sequential or direct; keyed access may be sequential, skip sequential, or direct. Access may be forward or backward. Access may be for updating or not updating. A nonupdate direct request to retrieve a record can optionally cause positioning at the following record for subsequent sequential access. The characteristics that may be specified are summarized, as follows:

- A request (including a request defined by a chain of request parameter lists) is either synchronous, so that VSAM does not give control back to your program until the request is completed, or asynchronous, so that your program may continue to process or issue other requests while the request is active and later use the CHECK macro to suspend processing until the request has been completed.
- For a keyed request, you specify either a generic key or a full key to which the key field of the record is to be matched. A generic search argument is matched for a less-than-or-equal comparison to the key field, and a full argument is matched for either an equal or a less-than-or-equal comparison to the key field.
- For retrieval, a request is either for a data record to be placed in a work area in the processing program or for the address of the record within VSAM's I/O buffer to be passed to the processing program. For all other requests (requests that involve updating or inserting) the work area contains the data record.
- For a request to gain direct access to a control interval, you specify the RBA of a 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 to be intermixed. If you provide your own buffers, intermixing is not allowed.

Address and size of the work area to contain a data record. You must provide a work area. It contains a data record or the address of the record within VSAM's I/O buffer. Having a work area that is too small is considered a logical error.

Length of the data record being processed. For storage, your processing program indicates the length to VSAM; for retrieval, VSAM indicates it to your program.

Length of the key. This parameter is required only for processing by generic key. For ordinary keyed access, the full key length is available to the Open routine from the catalog.

Address of the area containing the search argument. The search argument is either a key value or an RBA.

Address and length of an area for error messages from VSAM. Your routine for analyzing physical errors receives messages in this area.

GENCB: Generating Control Blocks and Lists

You use the GENCB macro in place of an ACB, EXLST, or RPL macro to generate an access-method control block, exit list, or request parameter list during the execution of your processing program, rather than producing it with the corresponding macro. You code GENCB the same as the other macros, but it enables you to generate one or more copies of a control block or list, and with GENCB, you can code parameter values in more ways—such as by putting a value in a register.

Manipulating the Information Relating the Program and the Data

The MODCB, SHOWCB, and TESTCB macros are for modifying, displaying, and testing the contents of an access-method control block, exit list, or request parameter list. The SHOWCAT macro is for displaying selected fields of the VSAM catalog.

MODCB: Modifying the Contents of Control Blocks and Lists

You use the MODCB macro to specify a new value for fields in an access-method control block, exit list, or request parameter list in the same way you defined them originally. For example, to use a single request parameter list to directly retrieve the first record having a certain generic key and then to sequentially retrieve the rest of the records having that generic key, you would use MODCB to alter the request parameter list to change from direct to sequential access.

SHOWCAT: Displaying Fields of the VSAM Catalog

Your program can use the SHOWCAT macro to retrieve selected data from the VSAM catalog. The information, which is based on a specified object (cluster, alternate index, path, etc.), is returned in a work area that you must supply. SHOWCAT is described in *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications.*

SHOWCB: Displaying Fields of Control Blocks and Lists

SHOWCB allows you to examine the contents of fields in an access-method control block, exit list, or request parameter list. VSAM gives the contents to you in an area you provide and in the order you specify the fields. You may display the contents of fields additional to those that you define in the macros. For example, when a data set is open, you can display various counts, such as number of control-interval splits, number of deleted records, and number of index levels.

TESTCB: Testing the Contents of Control Blocks and Lists

The TESTCB macro enables you to test the contents of a field or combination of fields in an access-method control block, exit list, or request parameter list for a particular value and alter the sequence of your processing steps as a result of the test.

Requesting Access to a Data Set

All of the preceding macros are for preparing to process a data set. The request macros, GET, PUT, POINT, and ERASE, initiate an access to data. Each use of one of these macros requires a request parameter list (or chain of request parameter lists) that fully defines the request: the only parameter that is specified with a request macro is the identity of the request parameter list.

The CHECK macro synchronizes a request initiated by a macro in the asynchronous form. In asynchronous processing, VSAM gives control back to your program before completion of the request. You use CHECK to suspend processing, if necessary, until the request has been completed and to schedule any routines to handle unusual conditions. You use the ENDREQ macro to terminate a request that is not required to be completed or to free VSAM from keeping track of a position in a data set.

The options for using GET, PUT, POINT, and ERASE are outlined in the discussion of the RPL macro, and the use of each macro is discussed in the section "In What Ways Can VSAM Data Sets Be Processed?" in the chapter "Getting to Know What VSAM Is and Does."

Requesting Access to Index Records

Two macros enable you to gain access to the contents of records in the index component of a key-sequenced data set.

GETIX and PUTIX: Retrieving and Storing Index Records

To issue the GETIX and PUTIX macros you supply only the address of the RPL. A PUTIX must be preceded by a successful GETIX.

These macros are described in *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications.*

Using Shared Resources

VSAM provides macros that build and delete a pool of shared channel program areas, buffers, and control blocks; write a buffer; search a buffer pool for a range of RBAs; and mark a given buffer for output.

Complete descriptions of these macros are in *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications.*

BLDVRP: Building a VSAM Resource Pool

To build a shared resource pool, you issue the BLDVRP macro before you open any ACB in which the shared resources option is specified. There are two sharing options: local shared resources (LSR) specifies that the pool is to be shared within a single region and global shared resources (GSR) means that the pool is to be shared across the entire system. LSR is valid in both VS1 and MVS Release 3; GSR, which requires system authorization, is valid in MVS Release 3 only. These options are especially useful when many VSAM data sets are open, when the amount of activity in a given data set is difficult to predict, and when a single transaction requires access to several data sets.

DLVRP: Deleting a VSAM Resource Pool

After all the data sets having access to shared resources are closed, the DLVRP macro is used to delete the shared pool. The shared pool cannot be released if any such data sets are still open.

WRTBFR: Writing a Buffer

VSAM normally performs write operations immediately after a direct request for storage. To improve performance, those writes can be deferred by so specifying in the ACB. You can specify the WRTBFR macro any time you want those previously deferred write operations to be done. Thus, you can synchronize your processing at any point you choose.

SCHBFR: Searching a Buffer Pool

The SCHBFR macro enables you to search the buffer pool for a particular range of RBAs. You can specify at which buffer in the pool the search is to begin. The RPL is assumed to specify control-interval access.

MRKBFR: Marking a Buffer for Output

You can issue the MRKBFR macro to identify a buffer to be marked for output and then either be retained or released for reuse.

How is JCL Used?

VSAM uses a minimum number of JCL parameters. It has two optional DD statements, JOBCAT and STEPCAT, for specifying catalogs and an optional JCL DD parameter, AMP, for overriding parameters specified by a processing program.

JCL is used in MVS to catalog, uncatalog, and delete nonVSAM data sets in a VSAM catalog. Also in MVS, you can invoke dynamic allocation of auxiliary storage. Although this publication does not describe this function, you can dynamically allocate VSAM data sets and user catalogs. Access Method Services also provides for the dynamic allocation of data sets. For information, see *OS/VS2 Access Method Services* and *OS/VS2 JCL.*

Defining a VSAM Data Set

When you define a data set, no DD statement is required if Access Method Services can allocate space for the data set from an existing data space. If a data space must be created to allocate space for the data set that you're defining. you need a DD statement (in OS/VS1) for OS/VS job management to provide device allocation: you specify storage unit, volume, and a disposition of OLD. You never specify space parameters (SPACE, SPLIT, SUBALLOC) or a disposition of NEW, DELETE, CATLG, or UNCATLG, since you use Access Method Services to define and delete all VSAM data sets.

Processing a VSAM Data Set

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 are sufficient to describe the data set. A key-sequenced data set is defined by a single DD statement.

To limit a data set to access by a single job step, you use a disposition of OLD. You use a disposition of SHR in the JCL of separate jobs to enable two or more job steps to share a data set, provided the data set's definition in the catalog specifies that sharing is permitted.

Specifying VSAM Catalogs

The master catalog is always available, without JCL specification. You make user catalogs available by describing them in DD statements with special names for a job or a job step: JOBCAT and STEPCAT. You describe a catalog sufficiently by giving its data-set name and a disposition of SHR. A user catalog may be either a STEPCAT or JOBCAT catalog; if both STEPCAT and JOBCAT user catalogs are specified, the STEPCAT catalog is available for the step for which it is specified, and the JOBCAT catalog is available for all steps for which no STEPCAT was specified. VSAM uses a data set's name as a search argument to search a catalog.

In VS2, you can minimize the use of JOBCAT and STEPCAT DD statements for your jobs when you name your data set with a qualified entryname whose first qualifier is the name or alias of the catalog in which the data set is defined. When the catalog is not identified with a DD statement or explicitly named (that is, with an Access Method Services command's CATALOG parameter), the OS/VS scheduler searches the master catalog for the data set's entryname. If the entryname is not found, the system uses the entryname's first qualifier as a search argument and attempts to locate either a user-catalog connector entry or a user catalog's alias entry. If the system finds a user-catalog connector entry (that is, an entry whose name or alias is the same as the entryname's first qualifier), the system searches that user catalog for the data set's entry, using the data set's full entryname. (If the system does this search, its performance will be affected.)

Using Other JCL Parameters

Some JCL parameters are ignored, are invalid, or bring about the wrong results if used with VSAM, and VSAM has a special JCL DD parameter, AMP.

JCL Parameters Not Used with VSAM

VSAM ignores parameters for defining tape data sets: data-set sequence number, NSL, NL, BLP, and AL. You may not use the parameters for a sequential data set, DATA, SYSOUT, and *, for specifying a VSAM data set. These DD names are invalid for VSAM data sets: JOBLIB, STEPLIB, SYSABEND, SYSUDUMP, and SYSCHK.

These DD parameters are also invalid: UCS, QNAME, DYNAM, TERM, and the forms of DSNAME for ISAM, PAM (partitioned access method), and generation data groups. VSAM does not use temporary data sets or concatenated data sets. VSAM does use concatenated STEPCATs and JOBCATs.

VSAM's Special DD Parameter: AMP

The VSAM DD parameter, AMP, has subparameters for specifying attributes that you can also specify by way of the ACB or the EXLST macros: size of virtual-storage space for I/O buffers, number of I/O buffers for data and index records, number of concurrent requests to be processed, and name of an exit routine for analyzing physical errors. AMP values override any values specified by way of the macros or any defaults supplied by the catalog.

To mount only some of the volumes on which a VSAM data set is stored, you must specify the DD parameters VOLUME and UNIT. Specifying those parameters to open a DCB (to be processed through the ISAM interface program) prevents a reference to the catalog and requires you to use another AMP subparameter (AMORG) to identify the data set as a VSAM data set. If you specify those parameters to open a VSAM ACB, AMORG is not required.

Another subparameter is used for specifying checkpoint/restart options. They are described in "How Are Programs Restarted Following a Failure?" in the chapter "Protecting Data with VSAM."

PREPARING FOR VSAM

This chapter indicates, for all prospective users of VSAM, the programming languages in which you can write programs to use VSAM, and the use of TSO (Time Sharing Option) and SMF (System Management Facilities) with VSAM.

The topic "How Can Existing Programs That Use ISAM Be Used with VSAM?" is for users of ISAM and may be ignored by other readers. It contains detailed information for programmers to decide whether existing programs that use ISAM can use the ISAM interface to process new key-sequenced data sets with indexes or key-sequenced data sets with indexes into which indexed sequential data sets have been converted.

What Programming Languages Can VSAM Be Used With?

| You can use the OS/VS assembler, PL/I, COBOL, and VS BASIC COBOL languages to gain direct access to VSAM data sets.

You can also code programs in PL/I and COBOL, using ISAM, to process VSAM data sets by way of the ISAM interface.

How Can the Time Sharing Option (TSO) Be Used with VSAM?

TSO is a subsystem of OS/VS2 that provides conversational time sharing from remote terminals. You can use TSO with VSAM and Access Method Services to:

- Execute Access Method Services commands directly as TSO commands
- Execute a program to process a VSAM data set
- Execute a program to call Access Method Services
- Dynamically allocate a VSAM data set and execute a program that uses VSAM macros to process the data set
- Allocate a VSAM data set by way of a LOGON procedure and execute a program that uses either VSAM or ISAM macros to process the data set

VSAM data sets must be cataloged in the master catalog or in a user catalog. The master catalog is allocated when the system is initialized; you can allocate and gain access to a user catalog by making it the STEPCAT of a LOGON procedure or by using the naming conventions.

For details about writing and executing programs and allocating data sets with TSO, see *OS/VS2 TSO Terminal User's Guide* and *OS/VS2 TSO Command Language Reference*.

How Can System Management Facilities (SMF) Be Used with VSAM?

SMF is an optional program of OS/VS that provides the means for gathering and recording information that can be used to evaluate system usage. VSAM supplies volume and data-set information to SMF.

The following records are written to the SMF data set to support VSAM:

- Record type 62, VSAM cluster or component opened, is written at the successful or unsuccessful opening of a VSAM data set, index, or cluster. It identifies the catalog in which the object is defined and the volumes on which the catalog and object are stored.
- Record type 63, VSAM cluster or component cataloged, is written after a data set or cluster is cataloged or its entry is altered with new space allocation information (that is, the entry's object is extended). It gives the catalog record for a newly defined object and the old and new versions of an altered catalog record.
- Record type 64, VSAM component status upon closing or reaching end of space on a volume, is written when a VSAM data set, index, or cluster is closed, when it becomes necessary to switch to another volume to continue processing, or when no more space is available on a volume. It indicates what condition occurred, identifies the volume(s) on which the object is stored, gives the extents of the object on the volume(s), and gives statistics about processing events that have occurred since the object was defined.
- Record type 67, VSAM entry deleted, is written when a data set or cluster is deleted. It identifies the VSAM catalog in which the object was defined and gives the catalog record.
- Record type 68, VSAM entry renamed, is written when a data set, index, or cluster is renamed. It identifies the VSAM catalog in which the object is defined and gives the old and new names.
- Record type 69, VSAM data space defined or deleted, is written when a data space is defined or deleted. It identifies the catalog in which the data space is defined and the volume on which it is (or was) allocated. It also gives the number of data spaces on that volume and the amount of space available in them.

For additional information about SMF, see *OS/VS1 System Management Facilities (SMF)* or *OS/VS2 System Programming Library: System Management Facilities (SMF).*

How Can Existing Programs That Use ISAM Be Used with VSAM?

This section is intended for users of ISAM who are converting to VSAM. VSAM's ISAM interface minimizes your conversion costs and scheduling problems by permitting programs coded to use ISAM to process VSAM data sets. To use the interface, you must convert indexed sequential data sets to VSAM data sets (for which you can use Access Method Services), convert ISAM JCL to VSAM JCL, and ensure that your existing ISAM programs meet the restrictions for using the interface.

Comparison of VSAM and ISAM

In most cases, if you use the performance options described in the chapter "Optimizing the Performance of VSAM," you can get better performance with VSAM while achieving essentially the same results that you can achieve with ISAM; you can also achieve results that you can't achieve with ISAM. The use of your existing ISAM processing programs to process key-sequenced data sets depends upon the extent to which VSAM and ISAM are similar in what they do, as well as upon the limitations of the ISAM interface itself. This subsection describes the similarities and differences between VSAM and ISAM in the areas that you are familiar with from using ISAM and indicates the functions of VSAM that have no counterpart in ISAM.

Comparison of VSAM and ISAM in Common Areas

A number of things that ISAM does are done differently or not at all by VSAM, even though the same practical results are achieved. The areas in which VSAM and ISAM differ are:

- Index structure
- Relation of index to data
- Deleting records
- Defining and loading a data set

These differences are described in the paragraphs that follow.

Index structure. Both a VSAM key-sequenced data set and an indexed sequential data set have an index that consists of levels, with a higher level controlling a lower level. In ISAM, either all or none of the index records of a higher level are kept in virtual storage. VSAM keeps individual index records in virtual storage, the number depending on the amount of buffer space you provide. It optimizes the use of the space by keeping those records it judges to be most useful at a particular time.

Relation of index to data. The relation of a VSAM index to the auxiliary-storage space whose records it controls is quite different from the corresponding relation for ISAM, with regard to overflow areas for record insertion. ISAM keeps a two-part index entry for each primary track that a data set is stored on. The first part of the entry indicates the highest-keyed record on the primary track. The second part indicates the highest-keyed record from that primary track that is in the overflow area for all the primary tracks on the cylinder and gives the physical location in the overflow area of the lowest-keyed record from that primary track. All the records in the overflow area from a primary track are chained together, from the lowest-keyed to the highest-keyed, by pointers that ISAM follows to locate an overflow record. Overflow records are unblocked, even if primary records are blocked. VSAM does not distinguish between primary and overflow areas. A control interval, whether used or free, has an entry in the sequence set, and after records are stored in a free control interval, it is processed exactly the same as other used control intervals. Data records are blocked in all control intervals and addressed, without chaining, by way of an index entry that contains the key (in compressed form) of the highest-keyed record in a control interval.

Deleting records. With ISAM, you mark records you want to delete, either for you to erase subsequently or for ISAM to drop, should they be moved into the overflow area; VSAM automatically reclaims the space in a key-sequenced data set and combines it with any existing free space in the affected control interval. Because of its use of distributed free space for insertions and deletions, VSAM requires less data-set reorganization than ISAM does. The ISAM interface allows you the option of marking records for deletion or erasing records.

Defining and loading a data set. You define all VSAM data sets in a catalog and allocate space for them by way of Access Method Services, rather than by way of JCL. You can load records into a data set with your own processing program or with Access Method Services, in one execution or in stages.

VSAM Functions That Go Beyond ISAM

VSAM has capabilities that ISAM doesn't have:

Skip sequential access. You can process a key-sequenced data set sequentially and skip records automatically, as though you were using direct access.

Concurrent request processing. Processing is extended by concurrent sequential or direct requests, or both, each requiring that VSAM keep track of a position in the data set, by means of a single access-method control block and without closing and reopening a data set.

Addressed sequential access. You can retrieve and store the records of a key-sequenced data set by RBA, as well as by key. With ISAM, you can position by physical address, but you must retrieve in a separate request.

Direct retrieval by generic key. With VSAM, you can retrieve a record directly, not only with a full-key search argument, but also with a generic search argument. ISAM enables you only to position at a record by generic argument: you must retrieve the record separately.

Alternate indexes. Rather than keep multiple copies of the same information organized in different ways for different applications, you can build one or more alternate indexes over key-sequenced and entry-sequenced data sets. Each alternate index provides a unique way to gain access to the same base data set.

Key-range allocation. With a multi-volume key-sequenced data set, you can assign data to various volumes according to the ranges of key values in the data records. For a data set that resides on three volumes, you might assign records with key A-E to the first volume, F-M to the second, and N-Z to the third. Because you know which volume contains what records, you can specify that only the volume(s) containing the records you want to process be mounted.

Secondary allocation of storage space. When you define a VSAM data set, you can specify the amount of auxiliary-storage space that is to be allocated automatically, when required, beyond the primary space allocation. You can specify the amount in terms of a number of data records or in terms of a number of tracks or cylinders.

Automatic data-set reorganization. VSAM partially reorganizes a key-sequenced data set by splitting a control area when it has no more free control intervals and one is needed to insert a record.

No abnormal terminations by Open. The VSAM Open routine does not abnormally end, but returns an explanatory message in all cases where it cannot carry out a request to open a data set.

How to Convert an Indexed Sequential Data Set to a Key-Sequenced Data Set

To convert an indexed sequential data set to a VSAM data set that you can process either with an ISAM program by way of the ISAM interface or with a VSAM program, you must convert the ISAM JCL to VSAM JCL and use Access Method Services to define a key-sequenced data set in a VSAM catalog and allocate space for it. You may use your ISAM load program by way of the ISAM interface to convert the data set, or you may use Access Method Services REPRO. If your ISAM program loads records in descending sequence, you might want to change it to ascending sequence (or use REPRO), since descending sequence causes more control-interval splits. You can also build alternate indexes for the data set after converting it. For more details about the procedure, see the discussion of the Access Method Services DEFINE and REPRO commands in the section "How Is Access Method Services Used?" in the chapter "Communicating with VSAM."

Figure 21 summarizes converting indexed sequential data sets to key-sequenced data sets and processing them either with programs that have been converted from ISAM to VSAM, with programs that still use ISAM, or with new VSAM programs. Most existing programs that use ISAM require little or no modification to use the ISAM interface to process VSAM data sets.

What the ISAM Interface Does

When a processing program that uses ISAM issues an OPEN that specifies a DCB describing an indexed-sequential data set that has been converted to or replaced by a key-sequenced data set, the Open routine detects the need for the ISAM interface and calls the interface's Open routine to:

- Construct control blocks and parameter lists that are required by VSAM
- Load the appropriate interface routines into virtual storage
- Initialize the ISAM DCB for the interface to intercept ISAM requests
- Take any DCB exit requested by the processing program

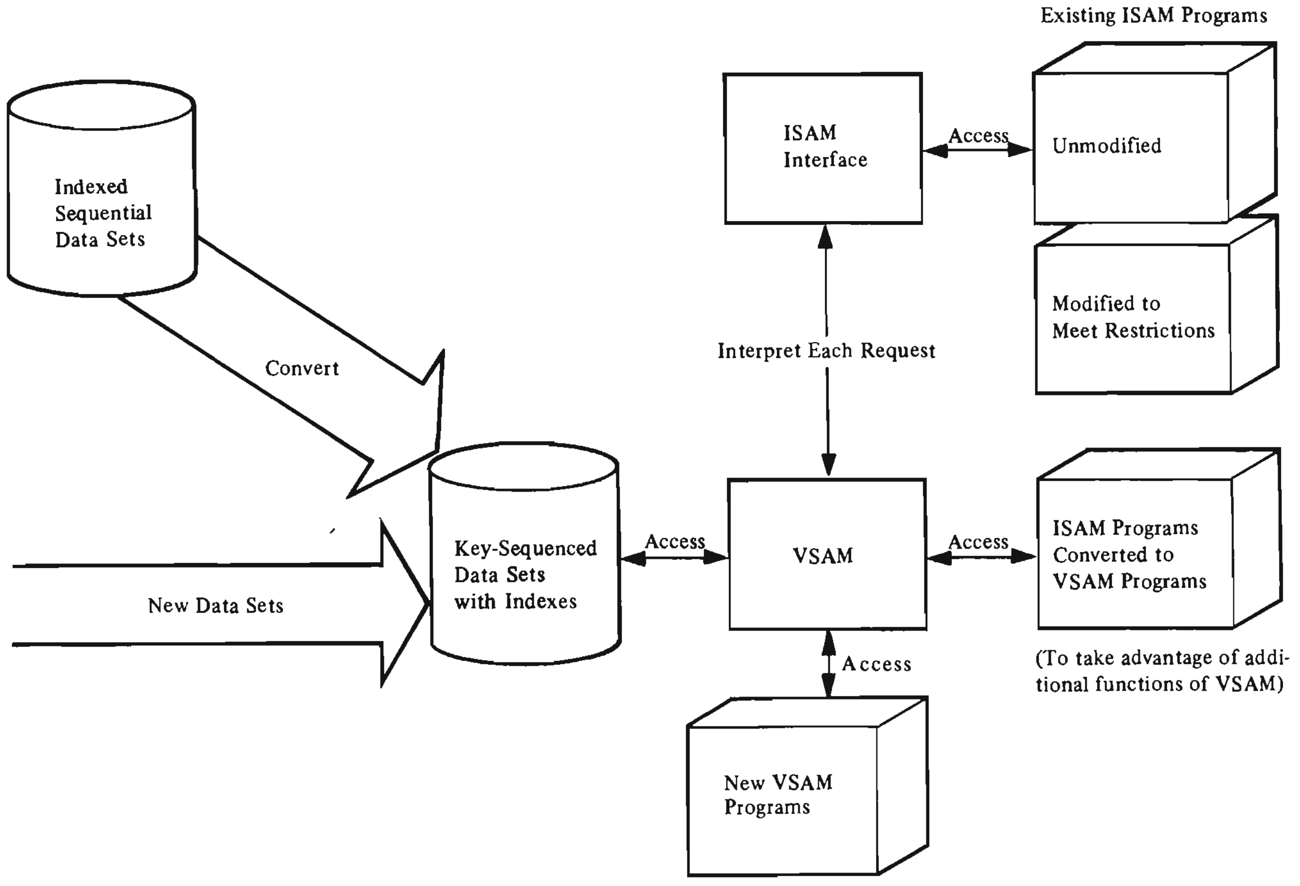
The interface intercepts each subsequent ISAM request, analyzes it to determine the equivalent keyed VSAM request, defines the keyed VSAM request in the request parameter list constructed by Open, and initiates the request.

All VSAM requests are handled synchronously; no VSAM CHECK macro is used. The ISAM CHECK macro merely causes exception codes in the DECB (data event control block) to be tested.

For processing programs that use locate processing, the 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.) The ISAM RELSE instruction causes no action to take place.

The 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.



Existing ISAM Programs
Indexed Sequential Data Sets
ISAM Interface
Access
Unmodified
Modified to Meet Restrictions
Interpret Each Request
Convert
Key-Sequenced Data Sets with Indexes
Access
VSAM
Access
ISAM Programs Converted to VSAM Programs
New Data Sets
Access
(To take advantage of additional functions of VSAM)
New VSAM Programs

Figure 21. Use of ISAM Programs to Process VSAM Data Sets

When the processing program closes the data set, the interface's Close routine issues a VSAM PUT macro for ISAM PUT locate requests (in load mode), deletes from virtual storage the interface routines loaded by Open, frees virtual-storage space that was obtained by Open, and gives control to VSAM Close.

Restrictions in the Use of the ISAM Interface

The ISAM interface enables programs that use ISAM to issue only those requests that VSAM or the interface can simulate. These are the restrictions for using the interface:

- The program must run successfully under ISAM. The ISAM interface does not check for parameters that are invalid for ISAM. See VSAM restrictions (e.g. OPEN macro (TYPE=J); temporary data sets).
- The program must use standard ISAM interfaces.
- If your program counts overflow records to determine reorganization needs, the count will be meaningless with VSAM data sets.
- You may share data among subtasks that specify the same DD statement in their DCB(s). But 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. Not all of the fields in a DCB can be counted on to contain valid information.

- The work area into which data records are read cannot 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 record length in the DECB is assumed to be the actual length of the record.
- 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.
- If the RECFM parameter is not specified in a processing program's DCB, you must specify it in the AMP parameter in the DD statement for the data set.
- A SYNAD routine must not issue VSAM macros or check for VSAM return codes. The ISAM interface translates all VSAM codes to appropriate ISAM codes. If your processing program already indicates a SYNAD routine, the routine specified in the AMP SYNAD parameter replaces it. You need not modify or replace a SYNAD routine that issues only a CLOSE, ABEND, SYNADAF, or SYNADRLS macro or examines DCB or DECB exception codes.

OPTIMIZING THE PERFORMANCE OF VSAM

This chapter is intended for programmers who will choose and implement the VSAM data set options that affect performance through the size of the control interval, the percents of distributed free space, and the handling of indexes and VSAM catalogs.

For more detailed information about options that affect VSAM performance, see *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*

How Can Control-Interval Size Be Used to Influence Performance?

A data set's control-interval size affects performance. As a general rule the larger the control interval, the better the sequential performance—for a number of reasons:

- Fewer index records required for a key-sequenced data set
- Fewer control-interval accesses, which is significant only for sequential or skip sequential access
- More efficient distribution of free space in a key-sequenced data set

You can request a particular control-interval size, but it must fall within the acceptable limits determined by VSAM, depending on the smallest amount of virtual-storage space you'll ever provide for I/O buffers and the size of your data records.

I/O-buffer size is important because VSAM transmits the contents of a control interval, and the amount of virtual-storage space for I/O buffers limits the size of a control interval. The amount of space for I/O buffers is the most flexible variable you have for influencing performance through control-interval size. The size and other attributes of your data records generally depend on the needs of your application.

How Does Distributed Free Space Improve Performance?

In the section "Key-Sequenced Data Sets" in the chapter "Getting to Know What VSAM Is and Does," we discussed the way VSAM uses distributed free space for the insertion of a record into a key-sequenced data set. It was pointed out that insertion can be achieved in a data set that hasn't any distributed free space, by means of a control-area split. Therefore, the decision to provide free space throughout the control intervals and control areas of a data set rests on considerations of performance. Free space in the immediate area into which a record is inserted speeds up the insertion and avoids control-area splitting, which may move a group of records to a different cylinder, away from the preceding and following records in key sequence.

The question that arises is: How much space do I provide? There is no one answer; the decision depends on how much inserting or lengthening of records you plan to do. Of course, if the data set will be for reference only, it will need no free space. If insertions into the data set are numerous, you might get the best performance by leaving half of the space free when you create the data set. In general, you should estimate the percent of growth and leave a proportionate amount of free space. For example, if you calculated 25% growth spread throughout the data set, you might leave 1/5 of the total space free, because the data set is now at 4/5 of its eventual size.

You may estimate that the growth of a data set will continue indefinitely. But if you attempted to leave enough free space for indefinite growth, you would end up with almost nothing but free space. So you have to decide how long a period of growth you want to provide for and count on reorganizing the data set at the end of that period to redistribute free space.

When you estimate data-set growth, remember that if records in a key-sequenced data set are deleted or shortened, VSAM makes the space thus freed available as free space within the control interval.

What Index Options Are There to Improve Performance?

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

- Index-set records in virtual storage
- Index and data set on separate volumes
- Sequence-set records adjacent to the data set
- Size of index control interval
- Replication of index records

Index-Set Records in Virtual Storage

To retrieve a record from a key-sequenced data set or store a record in it 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 space it is providing for VSAM to buffer index records. Enough space for one I/O buffer for index records is the minimum, but a serious performance problem with a space large enough for only one or two index records is that an index record may be continually deleted from virtual storage to make room for another and then retrieved again later when it is required. Ample space to buffer index records can improve performance by preventing this situation.

You ensure that index-set records will be in virtual storage by specifying enough virtual-storage space for I/O buffers for index records when you begin to process a data set. VSAM keeps as many index-set records in virtual storage as the space will hold. Whenever an index record must be retrieved to locate a record, VSAM makes room for it by deleting from the space the index record that VSAM judges to be the least useful under the circumstances then prevailing. It is generally the index record that belongs to the lowest index level then represented in the space and that has been in the space the longest.

Index and Data Set on Separate Volumes

You may place the index component of a key-sequenced data set on a separate volume from the data component, either on the same or on a different type of storage device.

Using different volumes eliminates the contention between gaining access to index records and gaining access to data records when you are using keyed access. The smaller amount of auxiliary-storage space required for an index makes it economical to use a faster storage device for it than for the data component.

Sequence-Set Records Adjacent to the Data Set

In using disk storage, you should minimize disk-arm movement. Having the sequence set accompany the data set is one way to reduce the movement for a key-sequenced data set. When you define the data set, you can specify that the sequence-set index record for each control area is to be on the track adjacent to each control area. This avoids two separate seeks when access to a data record requires VSAM to examine the sequence-set index record of the control area in which the data record is stored. One arm movement enables VSAM to retrieve or store both the index record and the contents of the control interval in which the data record is stored. When this option is taken, sequence-set records are replicated, as described next.

Size of Index Control Interval

The fourth option you might consider is ensuring that the index-set control interval is large enough to cover a full control area. Thus, the index-set control intervals may be larger than is actually required to contain the pointers to the sequence-set level; however, this option also keeps to a minimum the number of index levels required, thereby reducing search time and improving performance. This option does increase rotational delay and transfer time; therefore, you have to weigh the advantages and decide according to your own needs.

Replication of Index Records

The last option is the replication of an index record on a track of a direct-access storage volume as many times as it will fit. The object of replication is to reduce the time lost waiting for the record to come around to be read (rotational delay). Rotational delay is, on the average, half the time it takes for the volume to rotate. Replication of a record reduces this time. For instance, if ten copies of an index record fit on a track, rotational delay is, on the average, only one-twentieth of the time it takes for the volume to rotate.

This option costs auxiliary-storage space; it requires a full track of storage for each index record replicated. You have to weigh the relative values of auxiliary-storage space and processing speed.

You can replicate index records in these combinations of sequence set and index set:

- Sequence set separated from index set and only sequence-set records replicated
- Sequence set separated from index set but all index records replicated
- Sequence set and index set together and all index records replicated

Separating the sequence set from the index set is for placing the sequence set adjacent to the data, which is the previous option we discussed. Figure 22 illustrates replication of a sequence-set record that has been placed adjacent to its control area to avoid moving the arm separately for index and for data; the index record is replicated to reduce rotational delay.

How Can VSAM Catalogs Affect Performance?

Sharing Services with User Catalogs

A large number of requests for information from a VSAM catalog may result in some of the requests being answered more slowly than they would be if several catalogs had parts of the information. You might have the master catalog primarily contain pointers to user catalogs, which would contain entries for most data sets, indexes, and volumes. By decentralizing data set entries, you also reduce the time required to search a given catalog and minimize the effect of a catalog's being inoperative or unavailable.

Improving Catalog Performance in MVS

To improve catalog performance in MVS systems, you can:

- Mount the catalog volume on a nonshared DASD device.
- Preformat and initialize a new catalog before the first DEFINE operation. Although this takes some time, you realize improved performance on subsequent DEFINE operations into the catalog.
- Defining entries into a catalog that is not protected at the update level improves performance. The savings can be significant when you are using the CNVTCAT command to convert OS catalog entries into VSAM catalog entries.

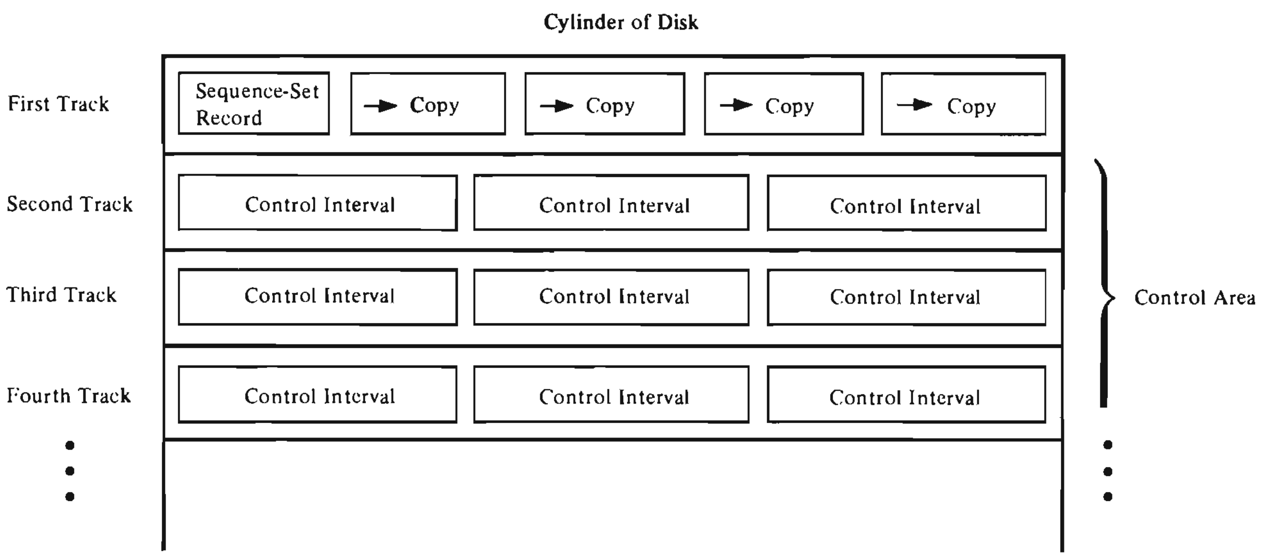


Figure 22. Replication of a Sequence-Set Index Record Adjacent to Its Control Area

PROTECTING DATA WITH VSAM

How safe is your data with VSAM? What provisions does VSAM make to ensure that data is not lost or destroyed by errors in the system, or sabotaged or pilfered by unauthorized persons? How easy is it to determine what the cause of a problem is and to do something about it? This chapter is intended for installation managers and system programmers interested in the answers to these questions.

The protection of data includes data integrity, or the safety of data from accidental destruction, and data security, or the safety of data from theft or intentional destruction. We'll discuss the attributes and options of VSAM that enhance data integrity, procedures for providing a backup copy of the catalog to protect data sets in the event of a catalog failure, protection of data shared by operating systems, regions, and subtasks, use of passwords and various authorization routines to prevent unauthorized access to your data, and methods of restart and problem determination.

How Does VSAM Achieve Data Integrity?

The attributes and options of VSAM that affect data integrity are:

- Method of inserting records into a key-sequenced data set
- Control-interval principle
- Method of indicating the end of a data set
- Verifying write operations

Method of Inserting Records into a Key-Sequenced Data Set

We discussed the method of inserting new records into a key-sequenced data set with an index in the section "Key-Sequenced Data Sets" in the chapter "Getting to Know What VSAM Is and Does." Free space distributed throughout used control intervals allows VSAM to insert a record into a control interval held in virtual storage by shifting records in it without an I/O operation. VSAM splits control intervals and control areas, when necessary, in a way that does not expose any data to loss, even if an I/O error occurs before the split is completed.

The order of operations for a control-interval split is:

1. Obtain a free (spare) control interval.
2. Move data from the control interval to be split to the new (spare) control interval.
3. Write the updated (spare) control interval.
4. Update the sequence-set control interval to reflect the use of free space and the new index entry for the new control interval.
5. Write the updated sequence-set control interval.
6. Update the old control interval and write it.

Control-Interval Principle

With a key-sequenced data set, the control interval is the unit pointed to by entries in a sequence-set index record. Only a record addition or a record insertion that splits a control interval or a control area causes a modification of the index. For instance, even though a record insertion might change the RBA of the record with the highest key in the control interval, the index entry is not altered, since the pointer in it is to the control interval, not to the record. Minimal index handling and modification lessen the chance of error.

Method of Indicating the End of a Data Set

VSAM combines two procedures for achieving data integrity:

- Preformatting the last control area of a data set
- Updating the catalog to indicate the RBA of the end of the data set and the highest-keyed record in the data set

Preformatting a Data Set

Preformatting the end of a data set as each control area comes into use ensures greater data integrity than formatting it only at the end of processing. VSAM formats a control area before using its control intervals by putting control information in them and putting an end-of-file indicator in the last control interval. The end-of-file indicator helps prevent data that has been added to a data set from being lost.

VSAM optionally preformats control areas when loading records into a data set and always preformats them when subsequently adding records to the data set. You have two options when loading records into a data set, whether you use the REPRO command of Access Method Services or your own processing program:

- The first option is to improve load speed: VSAM does not format the last control area of a data set until a CLOSE macro instruction is issued. An error that prevents further processing will result in the loss of all of the data that has been loaded.
- The second option is to improve the ability to recover from a failure and complete loading. Each time a control area is filled with records, VSAM formats the next control area before storing records in it. In this way each set of new records is protected against loss as it is added to the data set.

Updating the Catalog

The addresses kept in the catalog for the end of the data set enable VSAM to keep track of the physical end and, for a key-sequenced data set, the logical end of the data set. VSAM updates these addresses at intervals determined by a processing program's issuance of a temporary CLOSE macro instruction and at the end of data-set processing, when the data set is fully closed. By using the VERIFY command of Access Method Services, you can recover data in cases where VSAM was unable to close a data set properly and update the end-of-file indicator in the catalog. See the discussion of the VERIFY command in the chapter "Communicating with VSAM."

Verifying Write Operations

To improve the integrity of data written to auxiliary storage, you can request VSAM to verify each write operation for accuracy. Verification takes additional time, but it decreases the chance of introducing errors into the data set.

How Are Data Sets and Catalogs Protected?

Data integrity is not only ensured by standard provisions of VSAM, but can be improved by using various preventive and corrective measures. Some of these are Access Method Services facilities, which are described in *OS/VS1 Access Method Services* and *OS/VS2 Access Method Services.* These can be included within the design, use, and backup/recovery procedure of the VSAM structure.

Suggested preventive and corrective measures are:

- Using the VERIFY command to correct the catalog information in an attempt to regain access to a data set that was improperly closed.
- Using the optional WRITECHECK parameter of the DEFINE command to verify each write operation when writing data to auxiliary storage.
- Using the DEFINE SPACE command to dedicate use of volumes for VSAM data sets in order to segregate VSAM and nonVSAM recovery. You can dedicate a volume by defining a VSAM data space that occupies the whole volume.
- Minimizing or eliminating secondary allocations for data sets to overcome the complexity of using REPRO in catalog recovery stemming from secondary extents.
- Using the DEFINE USERCATALOG command to maximize the use of user catalogs and to limit the use of the master catalog. Compare the effect of the loss of a catalog when 10 data sets are cataloged in each of 10 catalogs, and 50 data sets are cataloged in each of two catalogs. The fewer the catalogs, the greater the disruption of daily operations in the event of loss of a catalog.
- Using the EXPORT command to create backup copies of data and associated catalog entries. The catalog entries can be reestablished in the catalog from which they were extracted or into a different catalog using the IMPORT command. The data set is reestablished by IMPORT without redefining it.
- Using the REPRO command to unload and reload catalogs.
- Using the optional RECOVERABLE parameter of the DEFINE MASTERCATALOG or DEFINE USERCATALOG command to create a catalog recovery area on each volume owned by that catalog. Only if a catalog is recoverable can the EXPORTRA, IMPORTRA, and LISTCRA commands be issued.
- Using the EXPORTRA command to recover data independent of the status of the catalog. The recovered data can then be imported into the system and catalog using IMPORTRA.
- Using the LISTRCRA command to list the contents of a catalog recovery area (CRA) and, if desired, to compare the contents of the CRA with the catalog. The COMPARE option is useful in detecting potential catalog

problems and in checking the validity of a catalog which was reestablished by means of the REPRO command or a volume restore.

- Using the ALTER REMOVEVOLUMES command to initialize a volume owned by VSAM when the catalog that owns the volume is inaccessible. VSAM indicators are reset and the VSAM space is returned to the VTOC as available space. There is no attempt to update a catalog during execution of the ALTER REMOVEVOLUMES command. It is assumed to be inaccessible.
- Using the DELETE SPACE (FORCE) command to remove information from both the VTOC and the catalog. When space is deleted using the FORCE option, the VTOC's VSAM volume ownership is given up, VSAM space is returned to the VTOC, the space definition in the catalog for that volume is deleted, and VSAM data sets on that volume are marked as unusable in the catalog. If you want to redefine the data sets, you must first delete them.
- Using the IEHDASDR DUMP/RESTORE system utility to create a backup copy of an entire volume and to restore that copy on a volume. The use of this utility in a VSAM environment requires special considerations because both the volume VTOC and the catalog contain space mapping information about the volume which has to be synchronized to ensure accessibility and to avoid damage to data. To avoid possible discrepancies between catalog and data set information, quiesce I/O operations and dump all related volumes at the same time. (See *OS/VS Utilities,* for details on how to use IEHDASDR.)

Your design for recovery should certainly consider all of the above measures. Most of them are discussed in the following sections.

Secondary Allocations of Data Sets

By eliminating or minimizing secondary allocation, the difficulty of using REPRO in catalog recovery stemming from secondary extents is overcome. An entry-sequenced data set is extended only by adding new control areas to the end of the data set. Thus, the effect of addition is predictable and the problem is eased. If it is impractical to allocate enough primary space to accommodate additions, the secondary allocation quantity should be large enough so that extension is infrequent. Whenever secondary allocation does occur, a new backup of the catalog or data set (or both) can be made. By monitoring the data set statistics in the catalog, either by way of a LISTCAT command or by way of a SHOWCB macro against an open ACB (to inspect the number of bytes of available space), you can predict when secondary allocation will occur. You can determine when a secondary allocation has taken place with a SHOWCB or TESTCB for the RPL feedback information after each PUT request.

For a key-sequenced data set the problem is much more complicated. If existing records are not lengthened and all additions are made to the logical end of the data set, then the situation is similar to that of an entry-sequenced data set except that the index must also be checked. The other patterns of insert and update activity are limitless. Some of them may be specific and dictate specific backup strategies, but discussion here assumes a random distribution of activity against the data set.

There are reasons other than recovery to design a key-sequenced data set to minimize extensions. A control-area split takes a relatively long time. For many online systems this can be a serious disruption. A characteristic of key-sequenced data sets is that, assuming a random insert pattern, all control areas tend to split at roughly the same time. Because each split results in two control areas being created from the original one, the data set's physical size doubles in a short period of time.

For these reasons it is advisable to design free-space percentages to minimize the probability of a split for a given insert level rather than to allow extra primary allocation for expansion. The data set should be reloaded (reorganized) when its insert level approaches the design point.

User Catalogs

Once a catalog defined without the RECOVERABLE attribute has been destroyed, the data it controls can no longer be accessed. Thus, if a system contains only one (master) catalog and that catalog is destroyed, the resources of the whole system are lost and must be restored using backup copies.

With several user catalogs, only the resources controlled by the destroyed user catalog are affected, and the catalog can be rebuilt while processing on other data continues. Since user catalogs, like the master catalog, are self-describing, even with the destruction of the master catalog, only the master catalog and the resources directly connected to it need be rebuilt. No data sets in a user catalog connected to that master catalog can be accessed until the user catalog is again connected to a master catalog.

A large number of requests for information from a VSAM catalog may result in some of the requests being answered more slowly than they would be if several catalogs had parts of the information. You might have the master catalog primarily contain pointers to user catalogs, which would contain entries for most data sets, indexes, and volumes. By decentralizing data set entries, you also reduce the time required to search a given catalog and minimize the effect of a catalog's being inoperative or unavailable.

Data Set Backup and Recovery

Like users of other methods of organizing and storing data, you should establish backup and recovery procedures for your data sets and catalogs.

Consider first all of the rules governing the relationships of catalogs and catalog entries to VSAM data sets:

- All VSAM data sets must be cataloged. 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.
- A volume containing VSAM data sets or data spaces can be owned by only one catalog. All VSAM data sets on a volume must be cataloged in the same VSAM catalog. With multivolume data sets, all current and candidate volumes must be owned by the same catalog.
- Logical and physical mapping information is contained in the catalog entries. For data sets defined in suballocated VSAM data spaces, the catalog contains the only record of the physical extents allocated to the data set. For unique data sets, entries in the VTOC also contain a record of physical extents. In both cases only the catalog contains the logical-to-physical mapping information (the relationship of the RBA ranges of the data set to the physical extents).

Any recovery procedure must match 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.

Access Method Services has two utility functions for creating backup copies:

- The EXPORT command is used to create an unloaded, portable copy of the data set. The operation is simple, there are options that offer protection, and most catalog information is exported along with the data, easing the problem of redefinition. You can prevent the exported data set from being updated until the IMPORT command reestablishes its accessibility.
- The REPRO command is used to create either a SAM or a duplicate VSAM data set for backup. An advantage over EXPORT is the accessibility of the backup copy. A DEFINE command is required before reloading, but this is a relatively minor inconvenience, particularly if the original DEFINE statements can be used.

You can also write your own backup scheme. For performance reasons, it is advisable to integrate these into regular processing procedures whenever possible. For example, many large data sets which are normally accessed randomly require sequential processing during the regular cycle. A SAM (or any other) backup data set may easily be created as a by-product of this procedure without materially affecting performance.

You must keep in mind that any backup procedure that does not involve an image copy of the data set (such as EXPORT or REPRO) will result in data reorganization and the re-creation of the index for a key-sequenced data set. Therefore, any absolute references by way of RBA may become invalid.

Catalog Backup and Recovery

Because of the importance of the VSAM catalog, you should consider backup for the catalog as well as for the individual data sets. In theory, if all of the data sets in the catalog are backed up individually, as they should be, it is possible to recover from destruction of the catalog by carrying out recovery procedures for each of the data sets. In practice, this may be reasonable. The probability of losing an entire catalog is very low.

However, to speed recovery or minimize exposure in the case of catalog damage or destruction, two tools are available: catalog unload and reload using the REPRO command, and recovery of data and its associated catalog information (from catalog recovery areas on volumes owned by recoverable catalogs) using the EXPORTRA/IMPORTRA commands.

Catalog Unload and Reload

You can schedule catalog backups to minimize the exposure for critical data. Some events that may have changed the catalog since the last backup are the execution of a DELETE, permanent EXPORT, DEFINE, or IMPORT command, or the extension of a data set owned by the catalog.

If VSAM data sets or data spaces have been deleted or permanently exported since the last catalog backup and the catalog is reloaded or restored, then the deleted data sets or data spaces will still be defined in the restored catalog. Any attempt to process these entries will yield unpredictable results because the space reflected in the catalog may no longer be owned by the catalog. The catalog may be corrected by reissuing the DELETE commands.

If VSAM data sets or data spaces have been defined or imported since the last catalog backup and the catalog is reloaded or restored, then the defined data sets or data spaces will not be defined in the reloaded or restored catalog. Processing these data sets or data spaces by means of the restored catalog is not possible since they cannot be accessed. The space formerly occupied by these VSAM data sets or data spaces will not be usable, but may be recovered by scratching the format-1 DSCBs in the VTOC for the data spaces. If any volumes were added to the catalog (between the backup and the recovery), they will also be unusable until you use the DELETE space command with the FORCE option.

If a VSAM data set has been extended since the last catalog backup, the new extents will not be defined in the restored or reloaded catalog. Any attempt to process records in the added extents will result in a logical error. If the data set has been extended within space already allocated to the data set before the backup but has acquired no new extents, then you can issue the VERIFY command to update the catalog pointers, and the data set may be accessed normally.

The data in any extents that have been acquired by the data set since the catalog was backed up is unrecoverable. For an entry-sequenced data set the data in any new extents should consist only of records that have been added to the end of the data set. Therefore, it is possible to recover all of the data in the old extents by accessing the data set sequentially up to the end of the old physical space allocation. For a key-sequenced data set, the data in the new extents may be any portion of the data set because of control-area splits. An attempt to read the data in logical sequence will fail with an invalid RBA indication when the data in the new extents is reached. You could access the key-sequenced data set by means of addressed sequence, but you then have the problem of identifying the missing records. Individual data set recovery for those data sets affected will be necessary.

Automatic Catalog Backup

All VSAM catalogs can be defined with the RECOVERABLE attribute that makes it possible to recover VSAM data sets and their catalog entries should the catalog be damaged or destroyed. A backup of the catalog is automatically achieved by recording catalog information about a given volume on that volume as well as in the catalog itself. Recovery information is recorded on each volume owned by a catalog. Space for this information is automatically set aside when you acquire volume ownership on a new volume and also when you define the catalog itself. There is no separate catalog entry for the recovery space: VSAM records its physical track address in the volume's format-4 label.

The recovery information in the volume's catalog recovery area (CRA) is updated immediately whenever parallel information in the catalog is changed. The affected volume(s) must be mounted, and the kind of operation to be performed on an object (data space, cluster, path, etc.) determines which volume(s) to mount.

To recover a VSAM data set and its catalog entries, you issue the EXPORTRA command. EXPORTRA uses the information in the CRA rather than the catalog to gain access to VSAM data sets and produce a copy of the VSAM data sets. The copy can be introduced back into the system by means of the IMPORTRA command.

You can use the LISTCRA command either to list the contents of the CRA before you do selective recovery or to list the entries in the recovery area that are different from those in its associated catalog.

Data Recovery

Using some of the corrective measures listed below, you can analyze and recover from the following conditions:

- Data set not properly closed
- Inaccessible data set
- Unusable catalog
- Inaccessible volume

You can use the LISTCRA command with the COMPARE option to identify the level of recovery that is required for the latter three conditions. (This command, as well as EXPORTRA and IMPORTRA, is usable only with recoverable catalogs.) The mismatches detected by LISTCRA vary in their degree of seriousness. The following list (order by severity) shows the type of mismatch, why it may have occurred, and how serious it is. Only the most serious mismatch is identified. Subsequent sections tell how to use corrective measures to recover.

Catalog Volume Records

Message	**Type**	**Cause**	**Seriousness**
DATA SPACE EXTENTS	Mismatched data space group	A difference in the number, size, and/or location of VSAM data spaces. A difference in the number and/or location of extents for one or more data sets. Space has been extended or deleted.	Requires recovery of the entire volume.
DATASET DIRECTORY	Mismatched data set directory	A difference in the names and/or number of data sets associated with this volume.	Requires recovery of the entire volume.
VSAM Object Records			
CATALOG ENTRY HAS DIFFERENT NAME	Mismatched name	The catalog was restored, or the volume containing the data set was restored. As a result the record in the catalog pointed to by the CRA record is no longer for the same object.	Requires data set recovery.
VOLUME OR KEYRANGE	Mismatched volume or key range.	The catalog was restored, or the volume containing the data set was restored. As a result the object's volume locations or keyranges in the CRA do not match those in the catalog.	Requires data set recovery.

Message	Type	Cause	Seriousness
EXTENTS	Mismatched extents	Data set was not properly closed, the catalog was restored, or the volume containing the data set was restored.	Requires data set recovery.
HIGH USED RBA	Mismatched high used Relative Byte Address	Same as for mismatched extents.	Requires use of the VERIFY command to correct the high RBA.
STATISTICS	Mismatched statistics	Same as for mismatched extents.	No recovery action is required; mismatched statistics do not affect the accessibility of data.
OTHER	Mismatch of something other than the above fields, e.g. passwords.	Same as for mismatched extents.	Same as for mismatched statistics.

In general, there are two types of data recovery in terms of the currency of the recovered data: *repair* and *reset.*

The *repair* type of data recovery operation restores addressibility and access to the most current version of the data. Repair operations are generally used to correct problems such as read and write errors associated with the data itself or with the data description, for example, by assigning alternate tracks.

The *reset* type of data recovery operation restores addressibility and access to a version of the data other than the most recent. Reset operations are generally used to correct logical problems such as a programming error or faulty transactions. Reset is generally the most common form of recovery, probably because of the types of problems encountered and the level of data available for recovery. An example of a reset operation is the dump (copy) and restore of a volume.

The following list of utility programs, whether Access Method Services, VSAM, or system, shows the type(s) of data recovery (or analysis) each program can undertake:

EXPORT/IMPORT	Reset
REPRO	Reset/repair
EXPORTRA/IMPORTRA	Repair (recoverable catalogs only)
COPY AND RESTORE	Reset
LISTCRA (COMPARE)	Analysis (recoverable catalogs only)
DELETE UCAT(FORCE)	Reset/repair
ALTER REMOVEVOLUMES	Reset/repair
DELETE SPACE(FORCE) NOSCRATCH	Reset/repair
DELETE CLUSTER NOSCRATCH	Reset/repair
VTOC UTILITY (SUPERZAP)	Reset/repair
DELETE SPACE (FORCE)	Reset/repair
VERIFY	Repair

REPRO can be used to create a backup copy of the catalog. The catalog can be reloaded (using REPRO) from this backup copy and can be used to minimize the amount of work required to accomplish recovery. The usefulness of a backup copy depends on what modifications have been made to the catalog since the backup was created. The following actions that may have occurred subsequent to taking a backup copy will affect the usefulness of the backup catalog:

Altering the amount of space controlled by the catalog. The volume entry in the backup catalog is no longer valid and will mismatch with the catalog recovery area.

Defining or deleting data sets. The volume entry and some of the data set entries in the backup copy are no longer valid.

Suballocating space to a VSAM data set. The volume space map is invalidated which, in itself, is not serious. However, the data set entry is also invalidated, a serious problem.

Several of the following recovery procedures use volume restore. If this is indicated, one or the other of the following must be true:

- The volume being restored doesn't contain multivolume data sets.
- If a volume does contain a portion of a multivolume data set, all volumes which contain portions of those multivolume data sets are treated as a single unit; that is, if a volume restore is required, the entire set is restored.

Data Set Not Properly Closed

VSAM data sets are not properly closed if they were opened for output and a system failure occurred, or if a program that is open for output terminated abnormally. This condition is reflected in the catalog and is communicated to the next program which does an OPEN of the data set. It acts as a warning in that while the data set may actually have been properly closed, an error condition may exist such as an incorrect high RBA in the catalog, an incomplete write to a direct-access device, or duplicate data.

If an error exists, it is probably an incorrect high RBA in the catalog. The VERIFY command, which is used to correct this condition, scans a given data set starting from the catalog-specified high RBA to the end of the data set. The resultant high RBA is then used to update the catalog.

You can avoid an incomplete write to a direct-access device and duplicate data either by doing synchronous direct inserts or by using abnormal termination exits in which you issue a CLOSE or TCLOSE to close the data set properly.

If you suspect that a write operation is incomplete, you can issue either an IMPORT or a REPRO command to get an old copy of the data; intermediate updates or inserts are lost. The use of IMPORT or REPRO requires that you have a previously exported version of the data set available.

Duplicate data in a key-sequenced data set, the least likely error condition to occur, can result from a failure during a control interval or control area split. If the failure occurred before the index was updated, the insert is lost, no duplicate data exists, and the data set is stable and usable.

If the failure occurred between updating the index and writing the updated control interval into secondary storage, some data is duplicated. However, both versions of the data are accessible by using addressed processing. The condition can be corrected by issuing a REPRO or an IMPORT command. If you want the current version of the data, you can use the REPRO command to copy the current version to a temporary data set and again to copy it back into a newly created key-sequenced data set. If you have a backup copy of a previous version of the data, you can use the IMPORT command to obtain a reorganized data set without duplicate data.

If the index is replicated and the error occurred between the writes of the index control intervals but the output was not affected, both versions of the data can be retrieved. The condition is similar to that described in the preceding paragraph and the same recovery measures can be taken.

The sequence of operations for a control area split are similar to those for a control interval split; the possible error conditions and corrective actions are the same.

Although the likelihood of having duplicate data is rather small, you can further reduce it by specifying free space for both control intervals and control areas to reduce the problem of splits. The only warning indication that VSAM sets for this condition is 'data set not properly closed.' If a more positive indication is desired, you can obtain it by using the journal exit (JRNAD) to determine control interval and control area splits and the RBA range affected.

To summarize, the warning 'data set not properly closed' may indicate an error in a VSAM data set. This condition can generally be corrected by using the VERIFY command. If other errors are encountered or suspected, they can generally be corrected by using either the IMPORT or the REPRO command.

Inaccessible Data Set

A VSAM data set may become inaccessible due to damage to the data set itself, to related information in the catalog, or to both. Depending on the extent of damage and prior actions, it may be possible to get access to either the current or a previous version of the data. A data set is inaccessible when it cannot be opened or is either partially or completely unreadable.

If the data set cannot be opened, there is probably damage to the catalog. To determine the extent of this damage, you can use either the LISTCAT or the LISTCRA (with the COMPARE option) command, the latter only if the catalog is recoverable. If as a result of processing one of these commands, you find only local damage to a small number of data sets and no serious damage to volume information (that is, either there is no mismatch or a general mismatch has occurred), you can use one of the following procedures:

- If an exported copy of the data set is available, you can then import it to gain access to the level of data at the time the backup copy was made. If the catalog is recoverable and you want to gain access to the current level of data, you can use the EXPORTRA command to extract the data, and then you can reestablish it by using the IMPORTRA command. It is not necessary to do any volume cleanup prior to reestablishing the data because the volume information was not seriously damaged.
- If the data set can be opened but none of the data can be retrieved, either the data set has been destroyed or the catalog and volume are not synchronized. To verify the condition of the catalog, you can use either the LISTCAT or LISTCRA (with the COMPARE option) command. If the results indicate catalog damage, you can use the above procedure to gain access to the data. If no catalog damage is indicated, you can import a previously exported version of the data. REPRO can also be used to effect a recovery if a copy of the data set is available.

If the data set can be opened and partially read, the problem is either confined to the data set itself or an entire physical extent of the data set is not readable. If the latter occurs (where the catalog indicates one or more extents than there are on the volume), it may have been caused by a restore of a volume independent of the catalog. You can use the LISTCRA (with the COMPARE option) to verify the mismatch in the number of extents. If this type of mismatch is not verified, you can then issue either an IMPORT or a REPRO command to correct the problem, using a backup copy of the data set. If the problem turns out to have been in the catalog, you can use either IMPORT or REPRO, or EXPORTRA followed by an IMPORTRA.

To summarize, the inaccessibility of a VSAM data set can either be a local problem restricted to a small number of data sets, or it can be a more serious problem. This can be determined by the use of the LISTCRA command. If the problem is local, the EXPORTRA, IMPORTRA, IMPORT, and REPRO commands can be used to correct the problem.

Unusable Catalog

A catalog may become unusable due to physical damage to the catalog volume. Depending on the extent of the damage and prior actions, it may be possible to either repair or reset the catalog and the data it controls. A catalog is unusable when many VSAM data sets cannot be opened, the catalog itself cannot be opened, or the catalog volume is not usable.

If the catalog can be opened, but many VSAM data sets controlled by this catalog cannot be accessed, a problem with the catalog probably exists. To determine whether it is a catalog problem, either a LISTCAT or a LISTCRA (with the COMPARE option) can be used. If I/O errors are encountered or mismatches are detected, some form of catalog recovery is required. If not, the problem is confined to the data sets themselves and the procedures given for unusable data sets can be used.

If the problem is with the catalog, recovery depends on the availability of backup copies of the catalogs, volumes, and data sets, and whether the catalog has been defined with the RECOVERABLE attribute. This section first discusses recovery of catalogs without associated catalog recovery areas (CRAs), then catalogs with associated CRAs.

Catalogs Without Associated CRAs. Recovery by way of an image copy of the data set must reestablish usable catalog entries. One way to reestablish catalog entries is to save a backup copy of the catalog along with the data. The major drawback to this approach is that other data sets defined in the catalog may become unusable as a result of the catalog reload. For example, assume that a backup copy of a data set and its corresponding catalog have been restored, but other data sets defined in the same catalog have not. To avoid mismatch problems, you can use the reloaded catalog to unload the reloaded data set, using the REPRO or EXPORT command. You can then use the current catalog to reload the data set, using REPRO or IMPORT, while other data remains unaffected.

This procedure is useful where nonVSAM data sets on the same volumes as VSAM data sets are routinely backed up. It requires only one backup operation for multiple VSAM data sets, without a recovery operation for undamaged data sets. Recovery should be needed infrequently, if at all, so the extra time required for this procedure would be more than offset by the saving in routine backup time.

Catalogs With Associated CRAs. If an unloaded copy of the catalog built by REPRO or a backup copy of the catalog volume is available, you can do the following:

- Either reestablish the backup copy of the catalog or restore the backup copy of the catalog volume.
- Use LISTCRA with the COMPARE option to identify mismatched volumes and data sets.
- If the catalog volume entry is included as a mismatched volume (that is, a data set directory or a data space group mismatch), the catalog and its own volume CRA are out of synchronization. (This condition should not occur if the catalog volume was restored.) You can do the following steps to reestablish the catalog and its data sets:
  - Recover all data sets on all owned volumes using EXPORTRA.
  - Use the ALTER REMOVEVOLUMES command to clean up all owned volumes, including the catalog volume.
  - If the unusable catalog is a user catalog, use EXPORT (DISCONNECT) to detach from the master catalog to avoid a conflict in catalog names.
  - Define a new catalog and space on all owned volumes, using the DEFINE command.
  - Use IMPORTRA to reload the catalog and its data sets.
- If a volume entry other than the catalog entry is included as a mismatched volume (that is, a data set directory or a data space group mismatch), you can recover all data sets on the mismatched volumes using the EXPORTRA command.
- If there are mismatched data sets which are not on volumes which were mismatched, you can use the VERIFY command for those data sets which have only mismatched RBAs and EXPORTRA for those with more serious mismatches.
- For mismatched *volumes* which require the use of EXPORTRA, use DELETE with the FORCE option to clean up the volumes and then use a DEFINE SPACE on the volumes.
- Use IMPORTRA to reestablish the data sets recovered by means of the EXPORTRA command.

To summarize, an unusable catalog can be reestablished provided that certain backup procedures made possible by the system copy utility and the REPRO command are followed. The amount of work required to recover is based on the currency of the backup data. Factors which affect the currency of the backup data are activities such as altering the amount of space controlled by the catalog, defining and deleting data sets, and suballocating space to a VSAM data set.

Inaccessible Volume

A given volume may become wholly or partially unusable because of physical damage to the volume or because the catalog which owns the volume was restored to a state which is not synchronized with the volume. If the problem is due to a catalog restore operation, the procedure outlined under "Unusable Catalog" can be used to correct the condition. If the problem is due to physical damage to the volume, recovery depends on the availability of backup copies of the catalogs, volumes, and data sets, and whether the catalog to which the volume belongs was defined with the RECOVERABLE attribute. If the catalog was recoverable, then a catalog recovery area (CRA) on the volume contains duplicate catalog information for each data set on it. Within this context, this section first discusses recovery of volumes without CRAs, then volumes with CRAs.

Volumes Without CRAs

- If a dump of the volume is available and a reset of the entire volume is desired, you can restore the damaged volume.
- If a dump of the volume is available, reset of nonVSAM data sets is desired, and the VSAM data sets are accessible (but no reset is desired), you can do the following:
  - Recover the accessible VSAM data sets on the volume by using an EXPORT command.
  - Restore the volume.
  - Use a DELETE command with the FORCE option to clean up the volume and then use a DEFINE SPACE on the volume.
  - Reestablish the recovered data sets using the IMPORT command.
- If no dump of the volume is available, the volume is damaged only in the nonVSAM area, and VSAM data sets are accessible, you can do the following:
  - Recover the VSAM data sets on the volume using an EXPORT command.
  - Initialize the volume and reestablish nonVSAM data sets.
  - Use DELETE with the FORCE option to remove the volume from the catalog and then use a DEFINE SPACE on the initialized volume.
  - Reestablish the recovered data sets using the IMPORT command.
  - If reestablished nonVSAM data sets were cataloged, delete and redefine the nonVSAM entries.
- If no dump of the volume is available, VSAM data sets are not accessible, and backup copies of the data sets on the volume exist, you can do the following:
  - Initialize the volume and restore nonVSAM data sets.
  - Use DELETE with the FORCE option to remove the volume from the catalog and then use a DEFINE SPACE on the volume.
  - If exported copies of VSAM data sets are available, use the IMPORT command to reestablish them.
  - If backup copies of the VSAM data sets are available (not, however, any data sets created by IMPORT), define the data sets using the DEFINE command, and then use the REPRO command to load the backup copies onto the volume.

Volumes With CRAs

- If a dump of the volume is available and a reset of the entire volume is desired, you can do the following:
  - Restore the damaged volume.
  - Use LISTCRA with the COMPARE option to see if the volume entry is mismatched, or if there are data set mismatches.
  - If there are data set mismatches only, use the VERIFY command for those data sets which have only mismatched RBAs and EXPORTRA for those with more serious mismatches.
  - If there is a volume mismatch other than a general mismatch, all data sets on the volume must be recovered using the EXPORTRA command.
  - If there was a volume mismatch which required the use of EXPORTRA, use DELETE with the FORCE option to clean up the volume and then use a DEFINE SPACE on the volume. Keep in mind that a DELETE FORCE results in the loss of all VSAM data on that volume.
  - Use IMPORTRA to reestablish the data sets recovered by means of the EXPORTRA command.
- If a tape dump of the volume is available and reset of damaged data sets is desired, you can do the following:
  - Recover the accessible VSAM data sets on the volume by using the EXPORTRA command.
  - Restore the volume from tape.
  - Use an EXPORTRA command to recover the previously inaccessible VSAM data sets that were restored.
  - Use a DELETE command with the FORCE option to clean up the volume and then use a DEFINE SPACE on the volume.
  - Reestablish the recovered data sets using the IMPORTRA command.

To summarize, a given volume which is wholly or partially unusable can be reestablished if backup copies of the data are available. In certain cases, the current version of the data can be extracted from the unusable volume and reestablished in the system.

How Is the Integrity of Shared Data Protected?

Data can be shared by different operating systems, by different jobs in a single system, and by different subtasks in an address space. There are provisions for controlling data sharing, and, therefore, protecting the integrity of data.

In determining the level of sharing you intend to allow, you must evaluate the consequences of a loss of *read integrity* (reading the correct information) to the processing program and a loss of *write integrity* (writing the correct information) to the data-set owner.

When a read request is issued, VSAM reads an entire control interval into storage. VSAM forces exclusive control over a control interval when the control interval is being modified.

Data-set sharing is controlled by the interaction of:

- The use of the share option (SHAREOPTIONS parameter) in the Access Method Services DEFINE command; the share options are specified for single systems and multiple systems.
- The use of the SHR and OLD parameters in the DD statement that identifies the data set.
- The type of processing (input or output) for which the data set was opened.
- The shareability of the DASD device.
- The use of the RESERVE and DEQ macros with shared direct-access storage devices.
- The use of the ENQ and DEQ macros.

If a data set cannot be shared and is not available when you request to open it, the request is denied.

When you issue the OPEN macro for an access-method control block, the Open routine enqueues the names of the components of a cluster. If DISP=OLD is specified in a DD statement, only the *dsname* associated with that DD statement is exclusively reserved by the operating system. That is, only the cluster name is reserved. To have the cluster component(s) reserved as well (to avoid having one of them unavailable), you may include DD statements with DISP=OLD for the component(s) of the cluster. This practice will ensure that all resources needed to open the data set will be exclusively reserved before your task is initiated. VSAM processing of an open data set is dependent upon actual share options in effect.

Subtask Sharing

Subtasks within a region may share a data set through a single DD statement or through separate DD statements.

When separate DD statements are used and one or more subtasks are to perform output processing, the DD statements must specify DISP=SHR. With separate DD statements, several subtasks can share a data set under the same rules that apply to cross-region sharing: output processing is limited to update processing and/or add processing that will not change the high-used RBA.

With a single DD statement, or with multiple DD statements and the ACB DSN option specified, several subtasks can update a data set concurrently. This mode of subtask sharing is independent of the DISP specification. If, however, DISP=SHR is specified, subtask sharing and cross-region sharing can occur concurrently. To update a record, VSAM forces exclusive control over the control interval in which the record is stored. A GET macro, issued for update, gains exclusive control over the control interval in which the record to be updated is stored. When a subtask has exclusive control of a control interval, a GET macro, issued for update, against the same control interval by another subtask is refused, but may be reissued later. Exclusive control is relinquished when any request is made for data that is outside the control interval or when an ENDREQ macro is issued.

A read request may be satisfied for a resource that is being shared for both read and update processing. An update request may be satisfied for a resource that is being shared for read processing only.

Cross-Region Sharing

Independent job steps in a system may request the use of a data set at the same time. To share a data set, each job step must specify a disposition of SHR in its DD statement for the data set. The type of sharing allowed depends on the share option you specified when the data set was defined.

The following share options apply in a single-system environment:

- Cross-Region SHAREOPTION 1: The data set may be shared by any number of users for input processing *or* used by one user for output processing. With this option, the access method maintains full integrity.
- Cross-Region SHAREOPTION 2: The data set may be used by any number of users for input processing *and* by one user for output processing. With this option, the access method maintains write integrity, but the user must assume responsibility for read integrity.
- Cross-Region SHAREOPTION 3: The data set may be fully shared. With this option, the access method does nothing to assure integrity. The user 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. This option places very heavy responsibility upon the user, and it should not be treated lightly.
- Cross-Region SHAREOPTION 4: The data set may be fully shared, and buffers used for *direct processing* are refreshed for each request. This option *requires* you to use the ENQ and DEQ macros to maintain data integrity while sharing the data set. Output processing is limited to update and/or add processing that does not change the high-used RBA if DISP=SHR is specified. Improper use of ENQ will cause failures similar to those described under SHAREOPTION 3.

Cross-System Sharing

The following sharing options, which you may specify when you define a data set, apply in a multiple-system environment:

- Cross-System SHAREOPTION 3: The data set may be fully shared. With this option, the access method does nothing to assure integrity. The user 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. This option places very heavy responsibility upon the user, and it should not be treated lightly.
- Cross-System SHAREOPTION4: The data set may be fully shared, and buffers used for *direct processing* are refreshed for each request. The RESERVE and RELEASE macros *are required* with this option to maintain data set integrity. Output processing is limited to update and/or add processing that does not change the high-used RBA if DISP=SHR is specified. Data set integrity cannot be maintained unless all jobs having access to the data set in a cross-system environment specify DISP=SHR. Improper use of RESERVE will cause failures similar to those described under SHAREOPTION3.

Job steps of two or more OS/VS 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.

Note: In a *shared-DASD* environment, integrity cannot be guaranteed by the system when users share a data set for output processing. VSAM does, however, provide assistance in protecting the integrity of the catalog.

Sharing a Catalog Among Systems

A master or user catalog in a VS1 or VS2 operating system can be shared as a master or user catalog in a different VS1 or VS2 system. The only exception to the above is that a master catalog in a VS2 operating system can only be shared as a user catalog in another VS2 system; it cannot be shared as a master catalog.

The integrity of data in a catalog shared between systems is completely controlled by Catalog Management processing routines. The concept of share options does not apply to catalogs.

How Can Passwords Be Used to Authorize Access?

Passwords are optional: you do not have to have them to gain access to a data set. But for added security, you can define passwords for data sets, indexes, and VSAM catalogs. There are different passwords for various degrees of data integrity:

- Full access. This is the master password, which allows you to gain access to a data set and any index and catalog record associated with it for all operations (retrieving, updating, inserting, deleting). Using this password to gain access to a catalog record gives you the ability to delete an entire data set and to alter password information or any other information in the catalog about a data set, index, or catalog (the master password is required for the PRINT and REPRO commands).
- Control access. This password authorizes you to use control-interval access and to correct timestamp mismatches.
- Update access. This password authorizes you to retrieve, update, insert, or delete records in a data set. It gives you limited access to catalog records: you can define objects and alter their definitions, but you cannot delete entries.
- Read access. This is the read-only password, which allows you to examine data records and catalog records, but not to add, alter, or delete them.

The passwords associated with a data set, index, or catalog are specified through Access Method Services when you define it. This information is kept in the catalog, and when a processing program attempts to open a data set, the security-verification routine checks whether a password is required and whether the correct one is given. Computer operators and communications terminal users may also be given the opportunity to supply the correct password, and you can specify how many times they may try to do so.

Besides VSAM password protection, you may also have your own routine to check a requester's authority. You can define security-authorization records in the master catalog or in a user catalog to contain whatever special password information you wish, for use by your authorization routine. VSAM transfers control to your routine when a requester gives a correct password other than the master password.

How Are Programs Restarted Following a Failure?

In general, the checkpoint/restart program for VSAM data sets is similar to that provided by OS/VS for ISAM and BDAM.

Recording Checkpoint Information

To restart after a failure that terminated processing, it is necessary to determine the status of processing programs when the failure occurred. A processing program defines a checkpoint by issuing a CHKPT macro instruction. The checkpoint program issues a VSAM temporary CLOSE macro to update the catalog. It then records information about VSAM data sets in a checkpoint data set. If a failure occurs, the latest checkpoint record can be used to reconstruct the situation that prevailed when the checkpoint was taken.

Restarting the Processing Program

Restart is the procedure of processing the checkpoint record and giving control back to the processing program interrupted by the failure. Different types of restart are distinguished for VSAM, for:

- Entry-sequenced output data sets. An entry-sequenced output data set is restored by the elimination of all records that have been added at the end since the checkpoint.
- Input data sets or key-sequenced data sets. A data set that was open for input at the checkpoint or a key-sequenced data set is prepared for restart by the restoration of any statistical information (such as number of records inserted) to its checkpoint status.

Restrictions and Options for Restarting a Program

The VSAM DD parameter, AMP, has a subparameter for specifying checkpoint/restart options that handle two special situations in restarting a processing program:

- Modifications other than records added sequentially to the end of an entry-sequenced data set. The restart program cannot restore a data set to its checkpoint status if there have been internal modifications to it since the checkpoint, and the restart program will normally not attempt restart processing.
- Addition of records to the end of a data set by way of a job step other than the job step that issued the checkpoint. Any records added to the end of an entry-sequenced data set will normally be erased in restoring the data set to its checkpoint status.

The AMP options for checkpoint/restart are: to let restart take its normal action for either situation, to override either one or the other of the two actions, or to override both. If you override the check for internal modification, your processing program is restarted, even though the data set it was processing cannot be restored; if you override the erasure of data at the end of a data set, your processing program is not restarted, if the catalog has been updated, unless you also override the check for modification.

For more information about checkpoint/restart with OS/VS, see *OS/VS Checkpoint/Restart*.

How Can the Causes of Problems Be Determined?

VSAM offers several diagnostic aids for you to determine what's wrong when things don't work.

Exits to Your Error-Analysis Routines

VSAM provides optional exits to routines you supply to handle error situations. If you provide the exit routines for analyzing errors, your processing program can investigate many errors and decide what to do in an orderly manner. Not only physical errors, but also logical errors that may arise out of unlikely combinations of events in a complex application can be handled by exits.

VSAM Messages

The messages put out by VSAM for the operator and programmer are designed to help them understand both the nature of the problem and the exact steps to take to correct it. Other messages that originate with VSAM are the diagnostic messages that are made available to your physical-error analysis routines and the open/close/end-of-volume messages that are printed in a special message area provided by your processing program.

Generalized Trace Facility (GTF)

GTF is an optional program of OS/VS that continually records, as they occur, events of selected classes that are necessary to trace a processing program. You must weigh the relative values of this diagnostic ability and the added processing time required. It is a debugging tool and a maintenance aid: it produces unformatted output. To format and print this output, use the Edit function of the HMDPRDMP or AMDPRDMP service aid. For information about GTF or the Edit function, see *OS/VS1 Service Aids* or *OS/VS2 System Program Library: Service Aids.*

VSAM Debug Switches

The CVT (Communications Vector Table) contains a field that, when set, allows you to run a VSAM program that contains an error and, when the error occurs, to issue a problem determination message and to save certain work areas that would otherwise be destroyed.

VSAM SNAP Dump Facility (VS2 Only)

The VSAM SNAP dump facility in VS2 provides a dump of VSAM-owned control blocks in CSA (common service area). Control blocks that are built for GSR (global shared resources) data sets reside in CSA subpools.

Cross-Reference Aids

Cross-reference aids, available on microfiche, provide two types of valuable information:

- Symbolic name usage table, which lists each symbolic name that appears in the VSAM code listings, lists each module that refers to the symbolic name, and specifies how each module refers to the symbolic name.
- Macro instruction usage table, which lists each macro instruction that is issued in VSAM listings, specifies the total number of times the macro

instruction is issued, lists each module that issues the macro instruction, and specifies the number of times the module issues the macro instruction.

The microfiche titles are: *OS/VS1 VSAM Cross Reference, OS/VS2 VSAM Cross Reference,* and *OS/VS2 Catalog Management Cross Reference.*

GLOSSARY

The following terms are defined as they are used in this book. If you do not find the term you are looking for, refer to the index or to the *IBM Data Processing Glossary,* GC20-1699.

Access Method Services: A multifunction service program that defines VSAM data sets and allocates space for them, builds alternate indexes, converts indexed sequential data sets to key-sequenced data sets with indexes, modifies data-set attributes in the catalog, reorganizes data sets, facilitates data portability between operating systems, creates backup copies of data sets and indexes, provides for catalog recovery, helps make inaccessible data sets accessible, and lists data set records and catalog records.

addressed direct access: The retrieval or storage of a data record identified by its relative byte address, 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 access: The retrieval or storage of a data record in RBA sequence relative to the previously retrieved or stored record. (*See also* keyed sequential access, addressed direct access, and keyed direct access.)

alternate index: A collection of index entries organized by the alternate keys of its associated base data records.

alternate index cluster: The data and index components of an alternate index.

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, key field, and prime key.)

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.

backup data set: A copy that can be used to reconstruct a damaged data set

base cluster: A key-sequenced or entry-sequenced cluster over which one or more alternate indexes are built.

catalog: (*See* master catalog and user catalog.)

catalog recovery area: (*See* CRA.)

cluster: A named, logical entity comprising one or more components and defining relationships between them.

collating sequence: An ordering assigned to a set of items, such that any two sets in that assigned order can be collated. As used in this publication, the order defined by the System/370 8-bit code for alphabetic, numeric, and special characters.

component: A named, cataloged collection of stored records. The lowest member in the data structure hierarchy. A data set contains at least one component, and the component can contain no named subsets.

compression: (*See* key compression.)

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 and distributes free space. It is the unit of information transmitted to or from auxiliary storage by VSAM, independent of physical record size.

control-interval access: The retrieval or storage of the contents of a control interval.

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.

CRA: Catalog recovery area. An entry-sequenced data set that exists on each volume owned by a recoverable catalog, including the catalog volume itself. The CRA contains self-describing records as well as duplicates of catalog records that describe the volume.

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 and not from the standpoint of the manner in which it is stored. (*See also* stored record.)

data security: Prevention of access to or use of data or programs without authorization. As used in this publication, the safety of data from unauthorized use, theft, or purposeful destruction.

data set: 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. (*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.

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.)

distributed 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.

entry sequence: The order in which data records are physically arranged (according to ascending RBA) in auxiliary storage, without respect to their contents. (Contrast to key sequence.)

entry-sequenced data set: A data set whose records are loaded without respect to their contents, and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

extent: A continuous space allocated on a direct-access storage volume, reserved for a particular data space or data set.

field: In a record or a control block, a specified area used for a particular category of data or control information.

free space: (*See* distributed free space.)

generic key: A leading 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.

horizontal pointer: A pointer in an index record that gives the location of another index record in the same level that contains the next key in collating sequence; used for keyed sequential access.

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; organized in levels of index records. (*See also* alternate index, index level, index set, and sequence set.)

index build: The automatic process of creating an alternate index through the use of Access Method Services.

index entry: A key and a pointer paired together, where the key is the highest key (in compressed form) entered in an index record or contained in a data record in a control interval, and the pointer gives the location of that index record or control interval.

index level: A set of index records that order and give the location of records in the next lower level or of control intervals in the data set that it controls.

index record: A collection of index entries that are retrieved and stored as a group.

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.

index upgrade: The process of reflecting changes made to a base cluster in its associated alternate indexes.

integrity: (*See* data integrity.)

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 with an index.

key: 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, alternate key, 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 an 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 content is a 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 data set whose records are loaded in key sequence and controlled by an index.

keyed direct access: The retrieval or storage of a data record by use of an index that relates the record's key to its relative location in the data set, 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 sequence relative to the previously retrieved or stored record. (*See also* addressed sequential access, keyed direct access, and addressed direct access.)

mass sequential insertion: A technique VSAM uses for keyed sequential insertion of two or more records in sequence into a collating position in a data set: more efficient than inserting each record directly.

master catalog: A key-sequenced data set with an index containing 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.

password: A unique string of characters stored in a catalog that a program, a computer operator, or a TSO 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, which defines a means of access to a cluster (an alternate index and its base cluster, for example).

physical record: A physical unit of 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. (*See also* horizontal pointer and vertical pointer.)

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.

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: The key of reference for a base cluster, key-sequenced data set when it was loaded. (*See also* key.)

random access: (*See* direct access.)

RBA: Relative byte address. The displacement of a data record or a control interval from the beginning of the data set to which it belongs; independent of the manner in which the data set is stored.

record: (*See* index record, data record, stored record.)

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.

replication: (*See* index replication.)

reusable data set: A VSAM data set that can be reused as a work file, regardless of its old contents.

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.

security: (*See* data security.)

segment: The portion of a stored record contained within a control interval. A stored record may consist of one or more segments. (*See* spanned record.)

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 or its key sequence, relative to the previously retrieved or stored record. (*See also* addressed sequential access and keyed sequential access.)

shared resources: A set of functions that permits 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. May not be used to retrieve records in descending key sequence.

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.

stored record: A data record, together with its control information, as stored in auxiliary storage.

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 catalog: An optional catalog used in the same way as the master catalog and pointed to by the master catalog. It lessens contention for the master catalog and facilitates volume portability.

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.

INDEX

For additional information about any subject listed in this index, refer to the publications that are listed under the same subject in either *OS/VS1 Master Index,* GC24-5104, or *OS/VS2 Master Index,* GC28-0693.

This index makes no page references to the glossary.

Planning for Enhanced VSAM under OS/VS
GC26-3842-1

Reader's Comment Form

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name, job title, and business address (including ZIP code).

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

GC26-3842-1

Fold and Staple

First Class Permit
Number 439
Palo Alto, California

Business Reply Mail

No postage necessary if mailed in the U.S.A.

Postage will be paid by:

IBM Corporation
General Products Division
Programming Publishing—Department J57
1501 California Avenue
Palo Alto, California 94304

Fold and Staple

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

Reader's Comment Form

Planning for Enhanced VSAM under OS/VS
GC26-3842-1

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name, job title, and business address (including ZIP code).

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

GC26-3842-1

Fold and Staple

First Class Permit
Number 439
Palo Alto, California

Business Reply Mail

No postage necessary if mailed in the U.S.A.

Postage will be paid by:

IBM Corporation
General Products Division
Programming Publishing—Department J57
1501 California Avenue
Palo Alto, California 94304

Fold and Staple

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

Planning for Enhanced VSAM under OS/VS (File No. S370-30) Printed in U.S.A. GC26-3842-1