GC24-5143-1
File No. S370/4300-30

Program Product

VSE/VSAM General Information

IBM

GC24-5143-1
File No. S370/4300-30

Program Product

VSE/VSAM General Information

Program Number 5746-AM2

Releases 1 and 2

IBM

Preface

This manual provides a general, introductory description of VSE/VSAM (Virtual Storage Extended/Virtual Storage Access Method), a licensed program that operates under DOS/VSE (Disk Operating System/Virtual Storage Extended). VSE/VSAM is a file management system that creates, maintains, and processes files on direct access storage devices. The intended audience of this manual is customer managers whose decisions will influence the use of VSE/VSAM, system and application programmers who will use VSE/VSAM in both new and existing programs, and others seeking an introduction to VSE/VSAM.

This book provides customer management and technical staff with the information they need to evaluate the applicability of VSE/VSAM to their installation, and to understand what is involved in installing, programming, and using the licensed program.

This publication contains:

- A general description of the concepts of VSE/VSAM and an overview of the product's capabilities.
- An explanation of the facilities provided.
- A general description of the installation considerations.
- A discussion of product performance.
- System configuration, the devices supported, and storage estimates and/or requirements.
- Conversion considerations.
- Compatability information.

Prerequisite Publications:

Introduction to DOS/VSE, GC33-6108

DOS/VSE Data Management Concepts, GC24-5138

Related Publications:

Using VSE/VSAM Commands and Macros, SC24-5144

VSE/VSAM Programmer's Reference, SC24-5145

Second Edition (July 1979)

This edition, GC24-5143-1, is a major revision of GC24-5143-0 with Technical Newsletter GN24-5634. This edition applies to Releases 1 and 2 of IBM Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM), Program Product 5746-AM2, and to subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information herein; any such changes will be reported in subsequent revisions or Technical Newsletters.

Information on VSE/VSAM Release 2, the VSE/VSAM Space Management for SAM Feature, and the Backup/Restore Feature is included in this book for planning purposes only until the availability of the products.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of Amentments

For a list of changes, see page 3.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Dept. G60, P.O. Box 6, Endicott, New York, U.S.A. 13760. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Summary of Amendments for VSE/VSAM General Information

Summary of Amendments for GC24-5143-1 Releases 1 and 2

This major revision supports Releases 1 and 2 of IBM Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM), Program Product 5746-AM2. It adds and updates pages with information related to:

- VSE/VSAM Space Management for SAM Feature
- Backup/Restore Feature

Miscellaneous additions, improvements, and corrections also appear throughout this manual.

Contents

Introduction .7

What is VSE/VSAM? .9
Relationship of VSE/VSAM to Virtual and Auxiliary Storage .9
VSE/VSAM's Data Organization .9
How VSE/VSAM Assigns Addresses to Data Records 10
The Physical Structure of VSAM Files 10
Control Intervals 10
Utilizing Space on a Storage Volume 11
Control Areas 12
Spanned Records 12
Clusters 12
VSE/VSAM's Disk Storage Structure 13
Entry-Sequenced Files 13
Relative Record Files 14
Key-Sequenced Files 14
The Role of a Prime Index in a Key-Sequenced File 14
Distribution of Free Space in a Key-Sequenced File 16
Using Free Space to Delete or Add Records 16
Control Interval/Control Area Splits 17
Alternate Indexes 18
Gaining Access to the Original File 19

The Processing of VSAM Files 21
How VSE/VSAM Locates Data Records 21

VSE/VSAM's Approach to Space and File Management 22
Master and User Catalogs 22

How to Communicate with VSE/VSAM 24
Job Control Statements for VSE/VSAM 24
VSE/VSAM's Access Method Services Commands 24
Processing Data with VSAM Macros 26

Preparing for VSE/VSAM 28
Machines With Which VSE/VSAM Can Be Used 28
What's New With VSE/VSAM 28
NOIMBED for Catalogs 28
Control Interval Split Integrity 28
Fixed Block Architecture Support 28
Data Space Classification Support 29
New Physical-Record Size 29
New SHOWCAT Operand 29
Improved EXPORT/EXPORTRA Performance 29
What's New for VSE/VSAM Release 2? 30
DASD Sharing 30
SHAREOPTIONS 4 Enhancement 30
Catalog Management Performance Improvement 30
Access Method Services CANCEL Command 30
Control Area Split Integrity 30
Dedicated VSAM Volume 31
Additional Classes of Space 31
Dynamic Files 31
Default Models 31
Default Volumes 31
Disposition Parameters for a File 31
Job Control Simplification 32
Partition/CPU Independence 32
Document Subset 32
VSE/VSAM Space Management for SAM Feature 32
Backup/Restore Feature 32
Storage Requirements 33
Programming Requirements 34
Programming Languages With Which VSE/VSAM Can Be Used 34
Use of VSAM by Existing Programs 34
Comparison of VSE/VSAM and ISAM 34
Differences Between VSE/VSAM and ISAM in Common Areas 34
VSE/VSAM Functions Not Available With ISAM 35

Summary of VSE/VSAM Advantages Over ISAM 36
How to Use the ISAM Interface Program 36
What the ISAM Interface Does 36
Restrictions in the Use of the ISAM Interface Program 36

VSE/VSAM Performance Options 38
Control Interval Size 38
I/O Buffer Space 38
Distributed Free Space 38
Preformatting Control Areas 38
Index Options 38
Index and Data on Separate Volumes 39
Number of Index Records in Virtual Storage 39
Sequence Set Records Adjacent to Control Areas 39
Replication of Index Records 39
Key Ranges 39
Performance Measurement 39

VSE/VSAM Data Protection 41
Options to Achieve Data Security 41
Passwords to Authorize Access 41
User Security Verification Routine 41
Options to Achieve Data Integrity 41
Protecting Shared Data 42
VSE/VSAM Data Backup Considerations 42
Data Recovery 42
Options For a Recoverable Catalog 44
Quick Recovery 45

Problem Determination 46
Exits to Error-Analysis Routines 46
VSE/VSAM Messages 46
VSAM Return Codes 46

Glossary 47

Index 49

Figures

Figure 1. Position of VSE/VSAM between a Processing Program and Auxiliary Storage 9
Figure 2. Adding a Fourth and Fifth Record to an Entry-Sequenced File 9
Figure 3. Relative Record File with Records 1, 3, 4, and 6 Inserted 9
Figure 4. Key-Sequenced File with Data Records Organized According to their Key-Sequence Value 10
Figure 5. Relative Byte Address 10
Figure 6. A Conceptual View of File XYZ's Divisions (or Control Intervals) 10
Figure 7. Control Interval Containing One or More Data Records 11
Figure 8. Conceptual View of a Control Interval That Occupies an Entire Track 11
Figure 9. Control Intervals, Physical/Logical Records, and Tracks 12
Figure 10. Distributing Free Space in a Key-Sequenced File 12
Figure 11. Control Intervals Containing Spanned Records 12
Figure 12. VSAM File Organization 13
Figure 13. Volumes, Data Spaces, VSAM Files, and nonVSAM Files 14
Figure 14. VSAM's Index Structure for a Key-Sequenced File 16
Figure 15. Vacated Space being Reclaimed by VSAM 17
Figure 16. Adding a Record to a Control Interval 17
Figure 17. Control Interval Split 18
Figure 18. Two Alternate Indexes Over a Single Key-Sequenced File 19
Figure 19. Differences Among Key-Sequenced, Entry-Sequenced, and Relative Record Files 20
Figure 20. Catalog Relationships 23
Figure 21. Communicating with VSE/VSAM 25
Figure 22. Use of ISAM Programs to Process VSAM Files 37
Figure 23. Sequence Set Record Imbedded 40

Introduction

A computer installation needs to provide facilities for a variety of different processing objectives. For example, it needs to be able to handle online, inquiry and batch transactions, as well as data base and data communications (DB/DC) applications--sometimes with multiple CPUs performing under control of different operating systems.

To accomplish this, an installation must have a file management system that is capable of supporting this demanding processing environment. The file management system (or access method) needs to provide programmers with the means to transfer data between main storage and auxiliary storage devices. In order to assist the programmer the file management system should meet the following requirements:

- Applicability to different types of processing that require different kinds of access and different levels of performance (such as online and batch processing).
- Central control over the creation, access, and deletion of files and over the management of space by keeping file and storage information in one place and making it independent of processing programs.
- High performance: fast retrieval and storage of data in direct or sequential fashion, uninterrupted by requirements to reorganize files and independent of previous insertions of records into files.
- Simplicity of use by means of a common set of instructions for different types of access, simplified job control statements, and optimization of the use of space in auxiliary storage.
- Independence from the type of storage device: no need to be concerned with physical record size, control information, and record deblocking.
- 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 data files in the event of failure or damage.
- Ability to move data from one operating system to another in a format that is common to them.
- Ease of conversion of data and programs from other access methods to this access method.

Over the years various techniques and access methods have been developed to manipulate data.

VSE/VSAM is a file management system that is designed specifically by IBM for the virtual systems environment; it consists of VSAM (virtual storage access method) and Access Method Services. VSAM is used to process files that reside on direct access storage devices and Access Method Services is a multifunction utility program that creates and maintains files on direct access storage devices.

VSE/VSAM provides an approach to meet the above requirements because of its:

Applicability to Different Types of Processing. VSE/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 direct requests.

VSE/VSAM permits both direct and sequential access and access can be by key or by 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.

Central Control. Extensive information about files and storage space is collected in a central information point called a VSAM catalog. Commands, specified by you, enable VSE/VSAM to control the definition and deletion of files and the alteration of information about them in the catalog. Access to this information may be protected by passwords assigned to the files or to the catalog itself.

These commands also help to manage the allocation of space on a direct access storage device. You simply set aside a total area of space to be used exclusively by VSE/VSAM. From this space, VSE/VSAM selects whatever space is needed for a file when it is created (defined). Space allocation is dynamic; if a file or catalog must be extended, more space is allocated to it.

Space for files can be allocated or deallocated without mounting volumes, because the information describing the contents of VSAM data spaces on those volumes is contained in the catalog. (The only exceptions are unique files and volumes that are recoverable. These terms are explained later.)

High performance. High performance is achieved by:

- An efficiently organized index that permits rapid access to individual records as well as rapid sequential processing

- Performance options for reducing disk-arm movement and rotational delay
- Space distributed throughout a file that is available for fast insertion of records (and consequently, minimal reorganization of the file).

The use of virtual and auxiliary storage for the index of a data file is self-optimizing. VSE/VSAM keeps in virtual storage as many index records as it can in the space you allow and reduces the size of the index by compressing key values to eliminate redundant information. (The type of index used for a data file is used also for VSAM catalogs to give fast catalog access.)

VSE/VSAM's method of inserting records into a data file provides access whose speed following a large number of insertions is equivalent to the speed of access without previous insertions. Available (free) space is used for efficient automatic reorganization of data files: inserted records are stored and addressed in the same way as original records, and space given up by deletions is reclaimed as free space.

Simplicity of Use. VSE/VSAM provides a common way of requesting the different types of access (sequential or direct, by key or by address), so you can use the same instructions to achieve different results.

The VSAM catalog holds the physical and logical attributes of all VSAM files in the system and also keeps track of the space allocated to these files. This simplifies the job control specifications needed to process a file.

VSE/VSAM automatically calculates an appropriate size for the unit (physical record) used to store data on a disk, and the total amount of auxiliary-storage space required for the number of data records you want to store. It optimizes the use of 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 VSE/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 do not need to be concerned with different formats for fixed-length or variable-length records.

Device Independence. The application programmer does not need to be concerned with storage devices and device addresses because VSE/VSAM accesses a record in a file by the record's displacement from the beginning of the file. VSE/VSAM calculates the corresponding storage-device address from that displacement, taking the physical characteristics of the file and the device into account.

The unit of data that VSE/VSAM transmits between main and auxiliary storage does not depend on the size of the physical records in which data is stored on a volume. This concept allows a VSAM file to be moved to different devices or even different device types without reprogramming.

Protection of Data. VSE/VSAM has been designed to provide full protection of data from inadvertant destruction or alteration (which is referred to as data integrity) and from unauthorized use or purposeful destruction or alteration (referred to as data security).

The design of VSE/VSAM allows access to data only via the catalog, that is, by identifying to the catalog the data you want to process. The catalog itself contains all the relevant information on your data.

You can optionally specify passwords for different levels of protection (read-only, update, full access) and include your own routine to check a requestor's authority to gain access to data. VSE/VSAM also provides various levels of protection for data that is to be shared across partitions.

You can select options for formatting files before data is stored in them and for verifying write operations to ensure data integrity.

Recovery of Data. VSAM catalogs that are defined with an optional recovery attribute allow data, which has become inaccessible because of a failure, to be recovered. Recovery is based on information recorded on the volumes controlled by the catalog as well as in the catalog itself.

Portability of Data Between Systems. VSE/VSAM's technique for storing records uses a format that is common to OS/VS, DOS/VS and DOS/VSE. Communication with VSAM is essentially the same for these operating systems, except for job control. The multifunction utility program (Access Method Services) includes functions that facilitate moving data files and volumes from one operating system to another operating system.

Ease of Conversion. If you are presently using ISAM (indexed sequential access method) files, VSE/VSAM provides for easy conversion of these files to VSE/VSAM format and for the continued use of your existing ISAM programs to process the converted files as well as the new VSAM files.

What Is VSE/VSAM?

The intent of this section is to provide you with some background information on VSE/VSAM's structure. This will help you to more easily understand the capabilities of VSE/VSAM as explained in the succeeding sections.

Relationship of VSE/VSAM to Virtual and Auxiliary Storage

Before beginning the discussion of VSE/VSAM, refer to Figure 1. This figure illustrates the role of VSE/VSAM in a DOS/VSE system; it shows how VSE/VSAM is situated between a processing program (logical data) and auxiliary storage (physical data). As you can see, VSE/VSAM is the connection or link between a user's logical data and the physical data in auxiliary storage; it can be viewed as a tool that frees programmers from the many tedious or time consuming tasks in the management of data.

Let us now begin our discussion of VSE/VSAM by first examining the different ways you can organize data to have it interact with auxiliary storage.

VSE/VSAM's Data Organization

VSE/VSAM is an access method that allows you to choose a data organization to meet your needs:

You can enter either fixed- or variable-length data records into a file one after the other. This is a sequential type of data organization referred to as an *entry-sequenced file.*

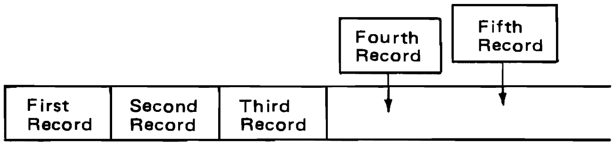


Figure 2. Adding a Fourth and Fifth Record to an Entry-Sequenced File

You can enter fixed-length data records into a file in assigned slots. This is a direct type of data organization referred to as a *relative record file.*

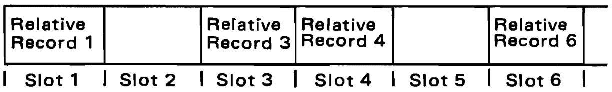


Figure 3. Relative Record File with Records 1, 3, 4, and 6 Inserted.

You can enter either fixed- or variable-length data records into a file organized in key sequence. This is an indexed type of data organization referred to as a *key-sequenced file.*

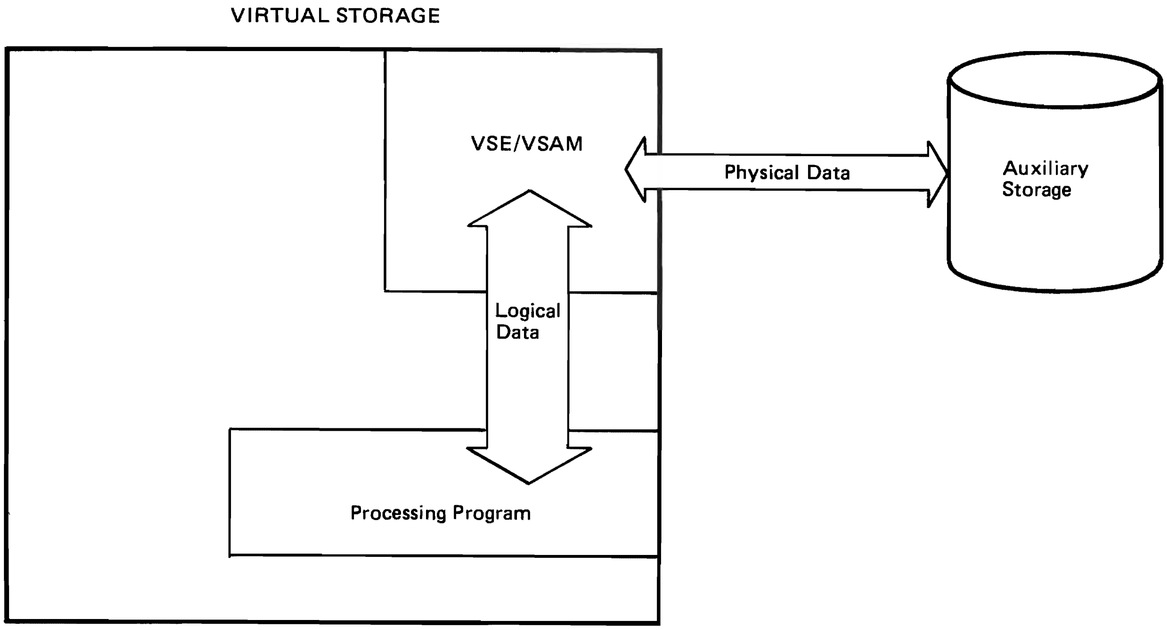


Figure 1. Position of VSE/VSAM between a Processing Program and Auxiliary Storage

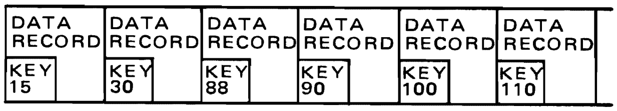


Figure 4. Key-Sequenced File with Data Records Organized According to their Key-Sequence Value

The primary difference among VSE/VSAM's three file organizations is the sequence in which data records are stored in them. These three file organizations will be discussed in greater detail in a following section.

How VSE/ VSAM Assigns Addresses To Data Records

VSE/VSAM uses a method of accessing data records that is independent of the device on which the data record is stored; it accesses a data record by its relative displacement, in bytes, from the beginning of the file and not by its physical-device address.

You do not have to be concerned with assigning addresses to data records, VSE/VSAM does this for you. In those instances where you need to know the address of a data record, VSE/VSAM will provide it to you.

VSE/VSAM assigns address 0 to the beginning of the first data record in each VSAM file (entry-sequenced, key-sequenced, and relative record). It then assigns each succeeding data record the byte value that equals the displacement of that data record from the beginning of the file. These displacement values are called *RBA's (relative byte addresses)*. (Do not confuse the terms relative byte address and relative record file; they are not the same.)

For example, in Figure 5, the first record in the file begins with relative byte address 0. The second record in the file has a relative byte address of 100 (this is equal to the length of the first record), the third record has a relative byte address of 200 (200 equals the length of the first record added to the length of the second record), and so on to the end of the file.

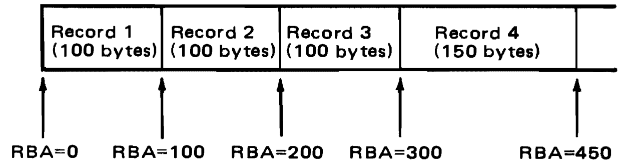


Figure 5. Relative Byte Address

The Physical Structure of VSAM Files

Transferring either a single data record or an entire file between virtual and auxiliary storage could be an inefficient and time-consuming way of processing records. What is needed is to choose a structure for data records that will make an access method as efficient as possible. VSE/VSAM's approach to this concern is through the way it formats auxiliary storage space.

Control Intervals

Picture your file (entry-sequenced, relative record, or key-sequenced) as fitting into or occupying a certain amount of auxiliary storage space. Additionally, picture the file's storage space as being divided (see Figure 6) into separate pieces. Each separate piece is called a *control interval.* The control interval and the data record(s) within it are what VSE/VSAM transfers between virtual and auxiliary storage.

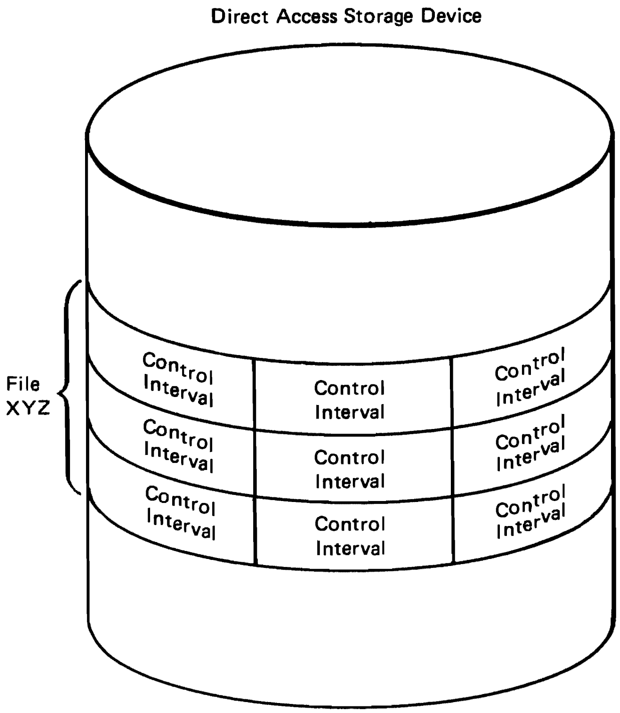


Figure 6. A Conceptual View of File XYZ's Divisions (or Control Intervals).

The size of each control interval can vary from one file to another but for a specific file, the size of each control interval is fixed (unchangeable length), either by VSE/VSAM or by you, within limits acceptable to VSE/VSAM. VSE/VSAM, in certain cases, will calculate an acceptable control interval size for you.

Figure 7 illustrates a typical control interval; it contains data records, unused (free) space, and control information. See "Distribution of Free Space in a Key-Sequenced File" for a discussion of free space. Control information provides VSE/VSAM the information needed about a control interval's data and space.

Just as a data record has a relative byte address, so does each control interval in a file have a relative byte address. The relative byte address of the first control interval in a file is 0. The relative byte address of the second control interval in the file is equal to the length of the first control interval, and so on up to the last control interval in the file.

Before proceeding to the next section let us review the highlights of the preceeding discussion:

- VSE/VSAM allows three different types of file organization; a sequential organization (entry-sequenced), a direct organization (relative record), and an indexed organization (key-sequenced).
- Addresses for the data records in a file are assigned by VSE/VSAM, according to the byte displacement of the records from the beginning (address 0) of the file.
- VSE/VSAM transfers data records between virtual and auxiliary storage as units of information called control intervals.
- Control intervals must be a certain size; VSE/VSAM will in certain cases, calculate a control interval size for you.

The next section explains how physical records are used by VSE/VSAM to help utilize storage space. Physical records apply to a CKD (count-key-data) device only and not to a Fixed Block Architecture device because VSE/VSAM can choose one of several different physical-record sizes for a CKD device whereas for a Fixed Block Architecture device no choice needs to be made (the size is always the same). See the "Glossary" for the definition of CKD and Fixed Block Architecture.

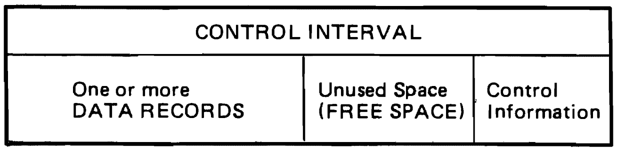


Figure 7. Control Interval Containing One or More Data Records

Utilizing Space On a Storage Volume

VSE/VSAM can store data records on different types of direct access storage devices. Because the different types of devices have different track (min-CA) capacities, the possibility exists that the control-interval size calculated for one type of device could be a poor choice for another type of device; much track space could be wasted if you created a VSAM file on one type of device and then subsequently moved it to another type of device. See the "Glossary" for the definition of min-CA.

VSE/VSAM frees you from this concern by the way it uses *physical records* to utilize track space. (Do not confuse logical records, control intervals, and physical records. The tracks on a storage device are divided into physical records. A VSAM file is divided into control intervals. Figure 8 shows how VSE/VSAM divides the logical records in a control interval into fixed-length physical records on a track.) VSE/VSAM chooses the physical-record size that best utilizes track space for a particular type of device; it selects the size to use based on the storage device and control interval size you specify or VSE/VSAM selects as a default. You do not have to be concerned with choosing physical-record sizes.

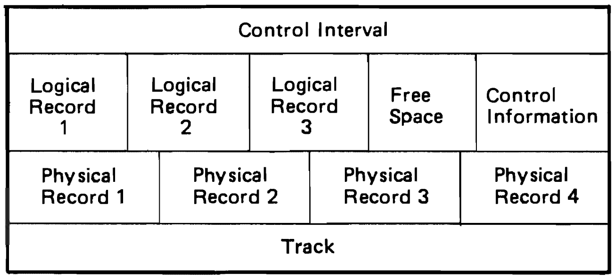


Figure 8. Conceptual View of a Control Interval That Occupies an Entire Track

Figure 9 illustrates how VSE/VSAM uses physical records to interchange the same size control interval between two direct access storage devices of differing track capacity. Device A (top part of Figure 9) accommodates one control interval (4 physical records) on track 1. Device B (bottom of Figure 9) accommodates a portion of one control interval (3 physical records) on track 1 and the remaining control interval portion on track 2 (1 physical record).

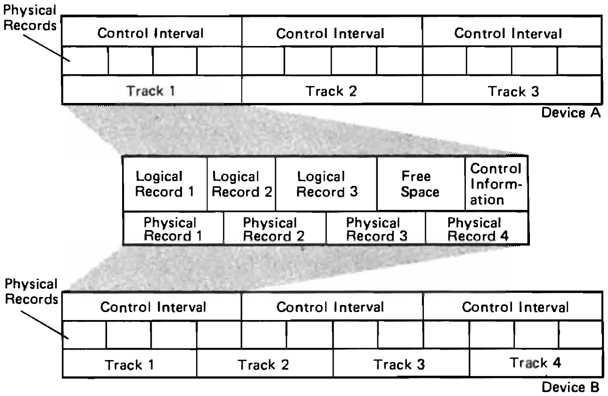


Figure 9. Control Intervals, Physical/Logical Records, and Tracks

The minimum physical-record size that VSE/VSAM uses is 512 bytes; it increases in multiples of 512 bytes to a maximum size of 8192 bytes. Note the similarity of physical-record size to control-interval size; they are both multiples of 512 bytes.

Let us now consider another basic structure of VSE/VSAM, the control area.

Control Areas

The control intervals in a VSAM file are themselves organized into larger structures called *control areas.* These are fixed-length areas of direct-access storage set aside by VSE/VSAM to accommodate the control intervals of a particular entry-sequenced, relative record, or key-sequenced file.

The number of control intervals per control area of a file is determined by VSE/VSAM and not by you. Whenever VSE/VSAM needs to extend the amount of space in a file, it extends it by one or more control areas.

Figure 10 illustrates a key-sequenced file that contains five control intervals in one control area. Notice the two empty control intervals. VSE/VSAM distributes empty control intervals throughout a *key-sequencedfile* as a percentage of control intervals per control area (you specify the percentage). The unused (free) space in this control area provides for the future growth of the file; as you add data records to the file, the free space is available to accommodate them without your having to take the time or trouble to reorganize the file. More information about adding data records to a key-sequenced file is presented in "Key-Sequenced Files."

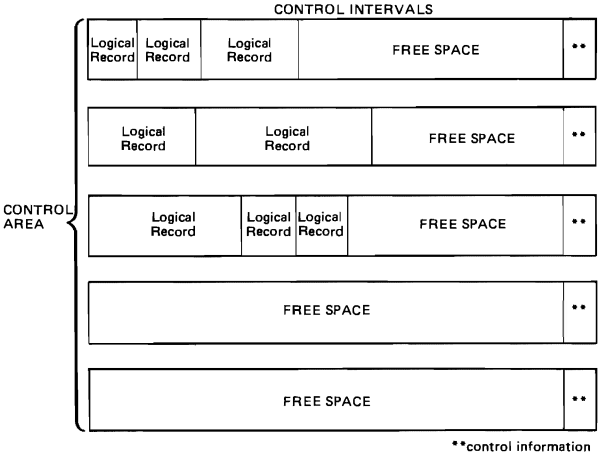


Figure 10. Distributing Free Space in a Key-Sequenced File

Spanned Records

If some of your data records happen to be larger than your control interval size you do not have to break them apart or reformat them in order to make them fit the control interval size. VSE/VSAM allows data records to extend across, or span, control interval boundaries. Such records are called *spanned records.* (Spanned records are applicable only to key-sequenced and entry-sequenced files.)

A spanned record must always begin on a control interval boundary. It occupies two or more control intervals within a given control area. As shown in Figure 11, the control interval that contains the last portion of a spanned record may contain unused space in those instances where it is not completely filled with data. This unused space can be used only to extend the spanned record; it cannot contain all or part of any other record.

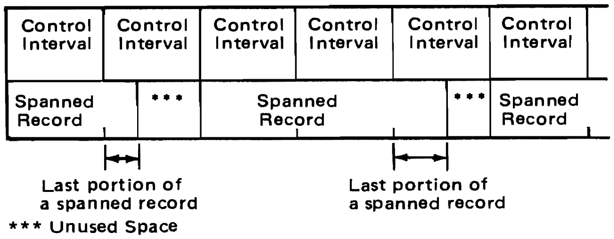


Figure 11. Control Intervals Containing Spanned Records

Clusters

As was stated previously, the primary difference among an entry-sequenced, a key-sequenced, or a relative record file is the sequence in which data records are stored in them. Another important aspect in the structure of VSAM files is illustrated in Figure 12.

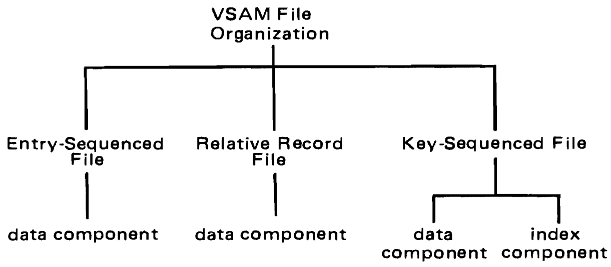


Figure 12. VSAM File Organization

Here you see that a key-sequenced file has a data component and an index component, while an entry-sequenced or relative record file has only a data component. The data component consists of the data records themselves, whereas the index component of a key-sequenced file is built and used by VSE/VSAM to locate the data records in the data component. More information on the index of a key-sequenced file appears in "Key-Sequenced Files".

VSE/VSAM's key-sequenced file organization allows you either to treat the data and index components separately or to combine them into a single unit called a *cluster.* For example, you could choose the name PAYROLL for the combined data and index components of a payroll file. The name PAYROLL would then be the *cluster name* used by VSE/VSAM to process the index and data components as a single functional unit. You could also choose to individually name the data and index components; this would allow you to refer to the components separately.

The concept of a cluster is also applicable to an entry-sequenced file or a relative record file. They are clusters also even though they have only a data component. The name that you give to these files is referred to as the cluster name.

VSE/VSAM's Disk Storage Structure

Up till now the discussion has dealt mainly with VSAM's file organization. Let us now examine how direct-access storage devices provide space for the files in a VSE/VSAM system.

VSE/VSAM reserves space on a direct-access storage volume for its exclusive use; this space is called a VSAM *data space.* A VSAM data space (for example Data Space 1 in Figure 13) can be as large as a volume.

When you create a file, you can specify either a space allocation that is large enough to have space left at the end of the file for future additions, or you can specify that VSE/VSAM extend the file for you when space is needed. (VSE/VSAM automatically extends the file by the amount of space you indicated in the creation of the file.) A file can be extended beyond its original size to a maximum size of approximately 4.3 billion bytes.

A VSAM data space can contain one or more VSAM files; conversely, a file can be stored on one or more direct-access volumes. For example, in Figure 13, Data Space 1 on Volume 1 contains File A and File B. Note that File A occupies two nonadjacent areas (A1 and A2) of storage. File C is stored in Volume 2 in two nonadjacent areas (C1 and C3) of storage as well as having a portion (C2) stored in Volume 3. Data space 4 on Volume 3 is available to VSE/VSAM but is empty.

Note that space can be set aside for nonVSAM files (middle of Volume 3) on a volume that also contains VSAM data spaces. (NonVSAM files are system files or files of other access methods.) Also note that space on a volume may be unallocated to any access method as yet (middle of Volume 2).

The next section examines VSE/VSAM's three file organizations in greater detail.

Entry-Sequenced Files

As stated previously, the records in an entry-sequenced file are stored in the physical sequence in which they are entered into the file. You cannot insert new records into the middle of the file; you can only add them to the end of the file. You cannot physically delete existing records, but you can replace a record with one of the same length. If you want to lengthen a record, you must code a new copy of it so that it can be written at the end of the file (the old copy remains in its original place in the file). Once you add a record to an entry-sequenced file it stays there and keeps its original relative byte address.

An entry-sequenced file is appropriate for applications that require no special ordering of data by the contents of a record. This file organization is well suited for a log or journal in which the order corresponds to a sequence of events.

Even though an entry-sequenced file is essentially a sequential file, you can also process it with direct access. To retrieve records directly from an entry-sequenced file, you must keep track of the records' relative byte addresses and associate these relative byte addresses with the contents of the records. (VSE/VSAM helps you to do this by providing

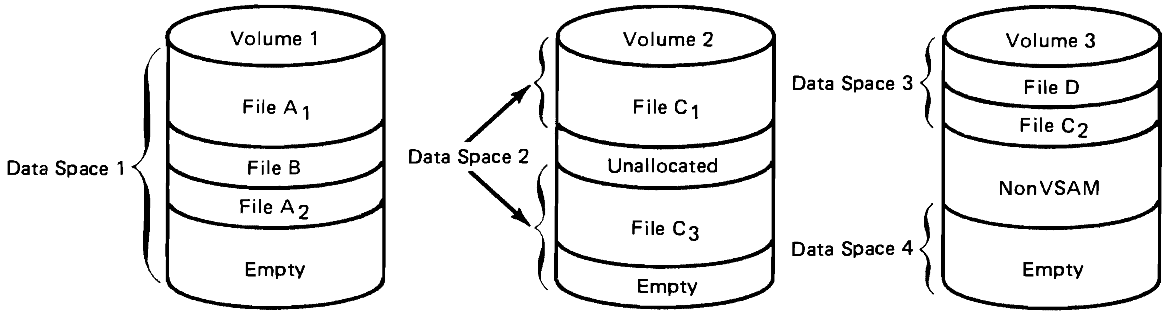


Figure 13. Volumes, Data Spaces, VSAM Files, and nonVSAM Files

you with a record's relative byte address when the record is loaded or added to the file.)

VSE/VSAM also provides you with another method of retrieving and updating individual data records. This method involves creating an index known as an alternate index (discussed in "Alternate Indexes") that allows you to directly access data records by a key. (You still retain the option of rapid sequential processing.)

Relative Record Files

A relative record file is essentially a structure of fixed-length slots. Each slot is assigned a relative record number that starts from one on up to the maximum number of records that can be stored in the file. One record occupies each slot; it is identified (stored and retrieved) by the relative record number of the slot.

Records in a relative record file are grouped together in control intervals, just as they are in a key-sequenced or an entry-sequenced file. Each control interval in a specific file contains the same number of fixed-length slots.

You can insert, delete or move a data record without affecting the position of other data records. A relative record file has no index because it is organized in a way that allows VSE/VSAM to first calculate the address of a control interval that contains the requested record and then the requested record's position within the control interval. VSE/VSAM retrieves records directly (based on a relative record number) or sequentially. VSE/VSAM does not permit the user to retrieve a relative record based on its relative byte address.

A relative record file is appropriate for many applications that use fixed-length records or where the user program assigns records to fixed-length slots. Each record could be processed to yield a unique relative record number (an employee's serial number, for example). In effect, each relative record number could be thought of as a key and each record could be located as though it were in a key-sequenced file, but without the time it takes to search the index of a key-sequenced file.

Key-Sequenced Files

Data records in a key-sequenced file are stored in a key sequence. Each record in the file has a key field that contains a unique (key) value, such as an employee or invoice number. VSE/VSAM uses this key value to store the data record into the file in its correct sequence.

A key-sequenced file has two distinctive features:

- A *prime index* that relates the key values in a file to the relative locations of the data records in a file.
- *Free space* that is distributed throughout the file for subsequent use when records are added and additional space is needed. (Free space also becomes available when you delete data records in a key-sequenced file.)

The Role of a Prime Index in a Key-Sequenced File

How do you keep track of all the records in a key-sequenced file, especially when your file is expanded by many new entries that can be located anywhere within the file? What do you have to do? As you will see in our discussion, VSE/VSAM will keep track of the records in a key-sequenced file, not you.

VSE/VSAM does this by building an index, called a prime index, that matches each data record's key-field value to its relative location in the file. (A prime index is different from the alternate index mentioned in the discussion of entry-sequenced files.) You can compare VSE/VSAM's index to the index in a book. Just as you refer to the index in a book to locate the page number of a particular topic, VSE/VSAM refers to its index to locate the pointer to a particular data record.

The size and position of the key field must be the same for every record in the file. Because VSE/VSAM keeps track of the records by their key-field value, you refer to a record by its key-field value and not by its physical-device address.

VSE/VSAM's high performance is due in part to its use of a prime index (sometimes just called an index); it allows you to access random records rapidly, while it provides you with the option of rapid sequential processing.

A VSE/VSAM index is itself a file with one or more levels. As Figure 14 illustrates, the top level of the index has a single record (index record) and each lower level has an increasing amount of records. Each level is a set of index control intervals, and each index control interval contains a single index record which can have one or more index entries.

Each level in the index contains entries giving the location of the index records in the next lower level. The index records in all the higher levels of the index are collectively called the *index set.*

The index records in the lowest level of the index are collectively called the *sequence set.* Their entries give the location of data control intervals that contain the data records.

Figure 14 illustrates how VSE/VSAM can quickly scan through an index to find a particular data record, either through direct access by key or sequential access by key. For direct access by key, VSE/VSAM follows vertical pointers from the highest index level down to the sequence set to find a pointer to the control interval containing the desired data record.

For sequential access by key, VSE/VSAM usually refers to the sequence set in the following manner. After retrieving the data records of one control area, VSE/VSAM will use a horizontal pointer in the sequence set to get to the next sequence-set record and, from there, to the next control area (and control interval) containing the data record which is to be processed next.

VSE/VSAM increases the number of entries that an index record of a given size can hold by compressing the key values in the index. That is, VSE/VSAM eliminates from the front and the back of a key value those characters that are not needed to distinguish it from the adjacent key values. For example, the key values in the key sequence 1110, 1230, 1450 will be compressed to 11, 23, and 45, respectively (the first and last digits of each number were dropped).

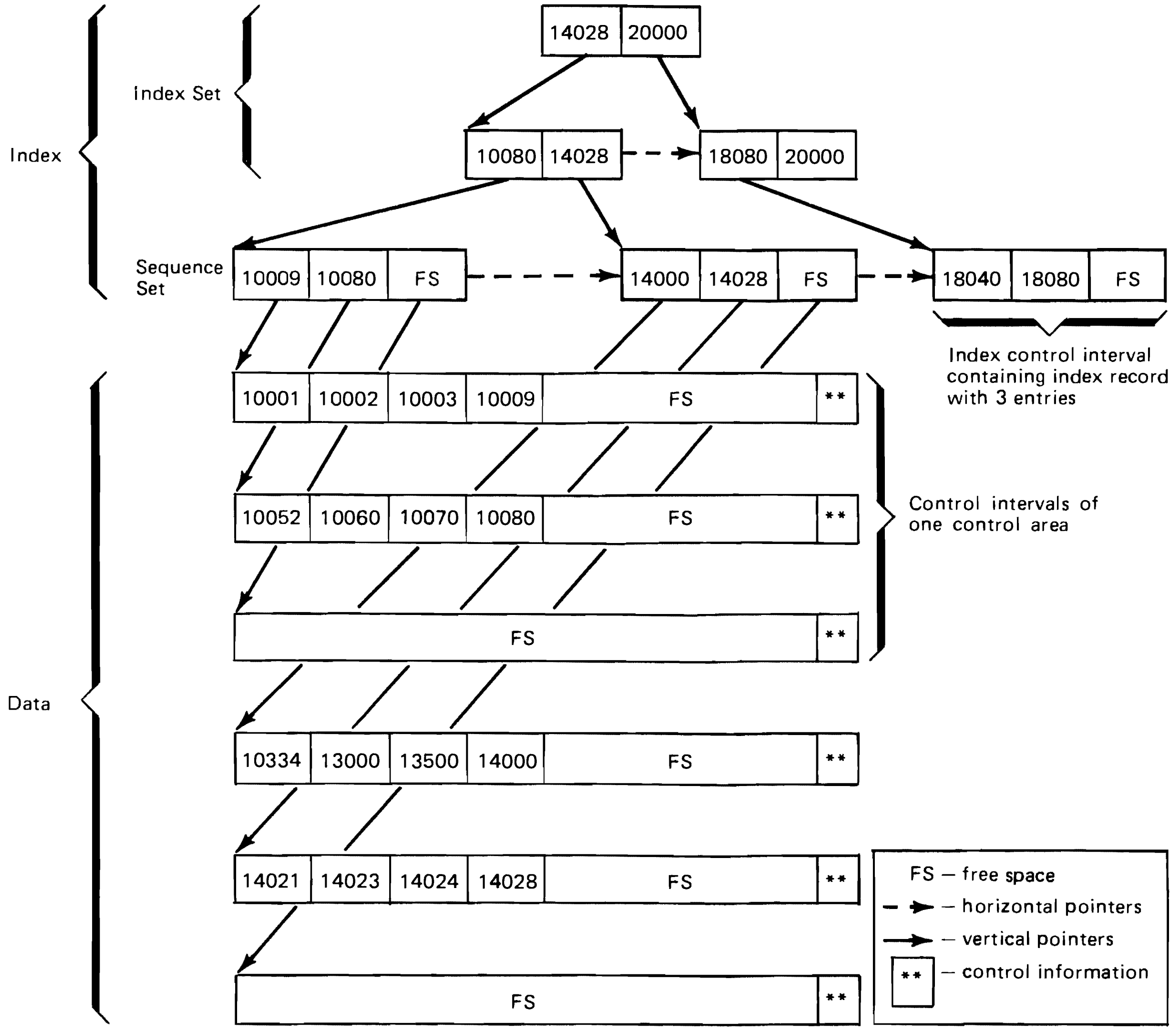


Figure 14. VSAM's Index Structure for a Key-Sequenced File

Distribution of Free Space in a Key-Sequenced File

The other distinctive feature of a key-sequenced file (besides its prime index) is its use of free space. When you create a key-sequenced file, you can specify that a certain percentage of free space is to be distributed throughout the file. This allows for the insertion or the lengthening of data records anywhere within the file. The amount of record processing that must be done to add a record is significantly reduced through the use of free space.

You can specify that free space is to be distributed throughout a key-sequenced file in two ways; by leaving some space free at the end of all the used control intervals, and/or by leaving some control intervals in a control area completely empty. The amount of free space in a used control interval and the number of free (empty) control intervals in a control area are independent of each other.

As Figure 14 illustrates, the sequence-set record for a control area has an entry for each control interval that contains data, as well as an entry for each control interval that is empty.

Using Free Space to Delete or Add Records

You may recall that when a record in a key-sequenced file is shortened or deleted (see Figure 15), the vacated space is automatically reclaimed by VSE/VSAM and added to the free space already in the control interval. This reclaimed space becomes available to you for future insertions or record extensions.

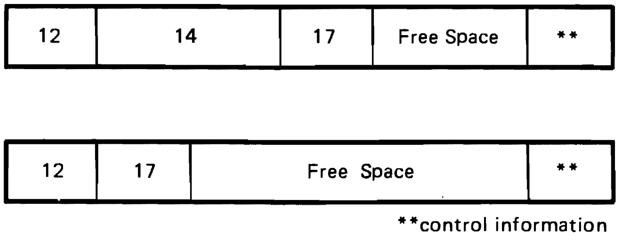


Figure 15. Vacated Space being Reclaimed by VSAM (Record with key value 14 was deleted.)

Conversely when a new record is added (see Figure 16) or when an existing record is lengthened, subsequent records in the control interval are moved over to the right into the following free space to make room for the new or lengthened record. This minimizes data movement because, as the example shows, you do not have to move records into a separate overflow area to accommodate the insertion of new records.

Control Interval/Control Area Splits

The preceding discussion assumed that there was enough free space in the control interval to accommodate a new record or a lengthened record.

If the record to be inserted does not fit into the control interval, a *control interval split* takes place. VSE/VSAM moves some of the data records (and their control information) from the full control interval to an empty control interval in the same control area, and inserts the new record in its proper key sequence.

Figure 17 illustrates a control interval split and shows the resulting free space available in the two affected control intervals. Since the number of records in the full control interval is reduced, any

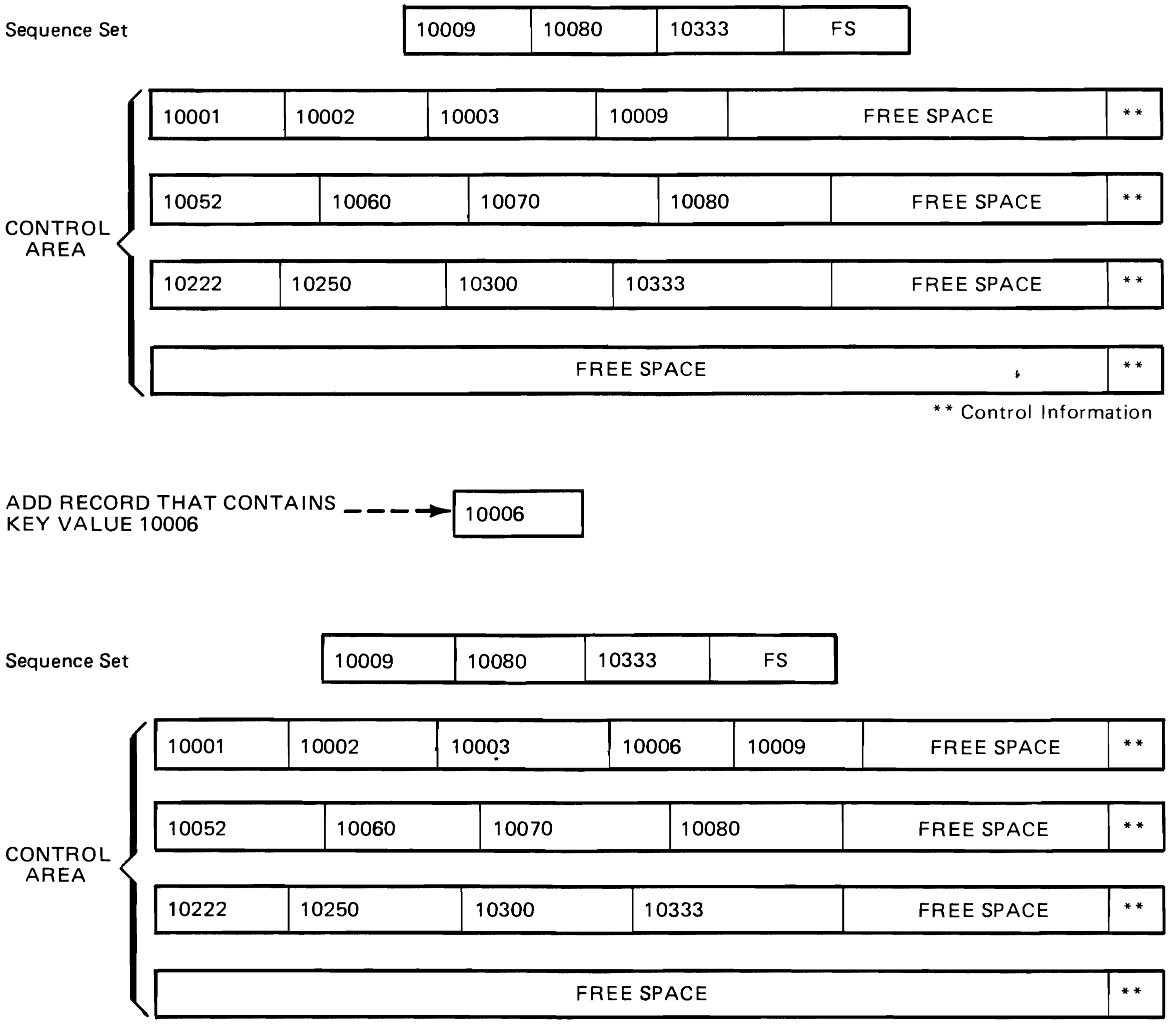


Figure 16. Adding a Record to a Control Interval.
(Note that (a) the relative byte address of "record 10009" will change after the new record is added
(b) the sequence set record in the index does not change.)

subsequent new or lengthened record can occupy the newly created free space in the control interval.

If the control intervals involved in a split are not adjacent,the relative byte address (RBA) sequence of the data records is no longer the same as their key sequence. In Figure 17, for example, the data records are not in ascending physical sequence after the control interval split. However the sequence-set index record for these control intervals always reflects the key sequence, so that, for keyed sequential requests, the data records will be retrieved in the correct order.

A *control area split* occurs if the available free space in the control intervals of a specific control area cannot accommodate a new record. VSE/VSAM sets up a new control area at the end of the file, either by using space already allocated or by extending the file (if you provided for extensions when you created the file). VSE/VSAM moves the contents of some of the control intervals of the full control area to the free control intervals in the new control area, and inserts the new record into the appropriate control area, as its key dictates. Generally speaking, direct inserts cause control intervals and control areas to be split at the mid-point. Sequential inserts, on the other hand, cause control intervals and control areas to split at the point of insertion.

Splitting should be an infrequent occurrence for files with sufficient distributed free space; splitting a control area does, however, allow you to insert records into a key-sequenced file without previously distributed free space.

Alternate Indexes

You have seen that VSE/VSAM always builds a prime index for a key-sequenced file. You can specify that VSE/VSAM build you an alternate index also.

An alternate index provides you with another

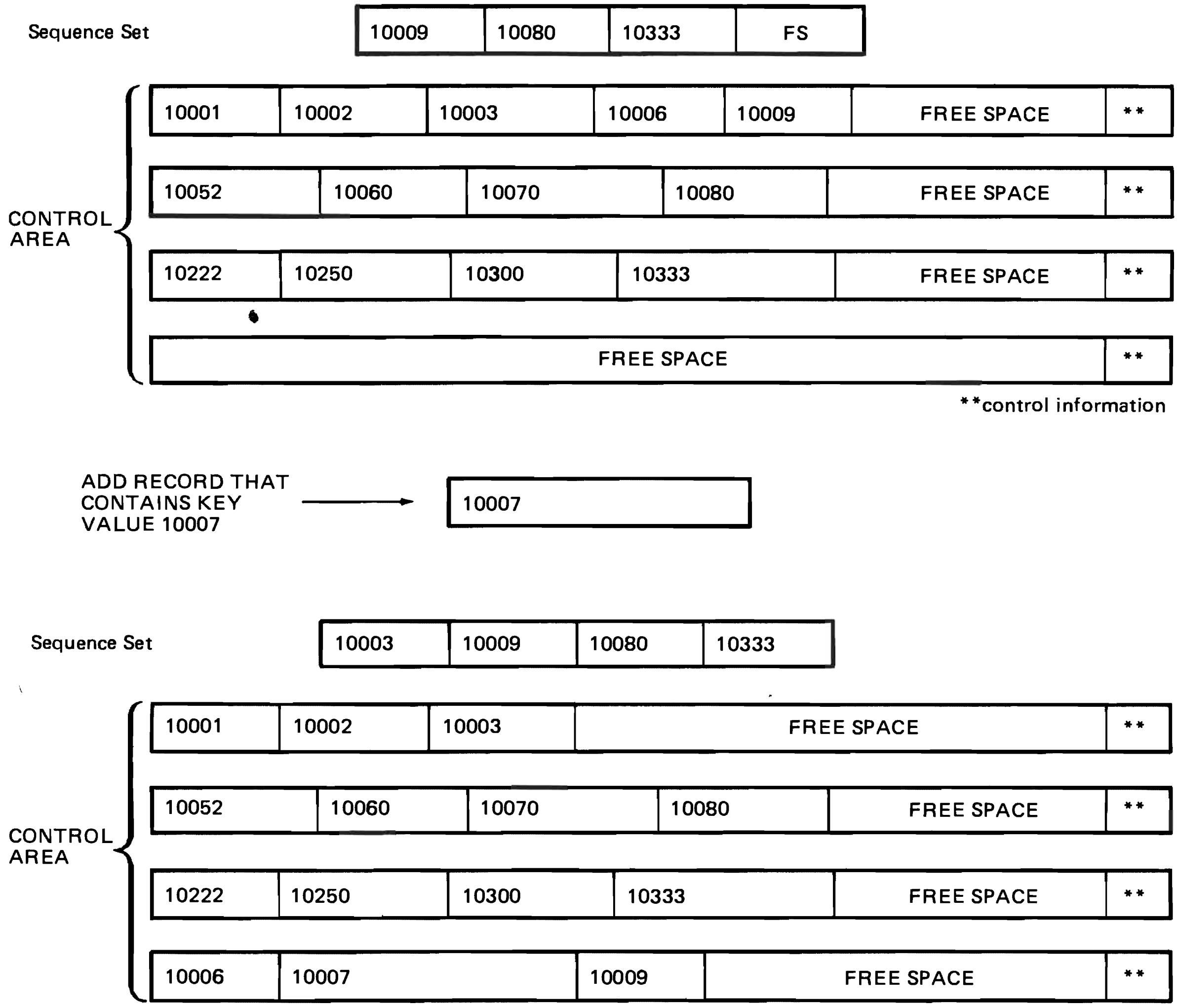


Figure 17. Control Interval Split

way of gaining access to the data records of a given key-sequenced or entry-sequenced file. It eliminates the need for you to keep multiple copies of the same information organized in different ways for different applications. You can, for example, take a key-sequenced payroll file that was originally indexed by employee name and using the same base data, index it according to employee number, department number, etc. (You can use any field in the data records of the original file as the alternate index key-field, as long as the field has a fixed length and fixed position in each record.)

You also can use an alternate index to gain the advantages of keyed direct access for entry-sequenced files which, as you recall, have no prime index.

The structure of an alternate index is very similar to that of a key-sequenced file; it consists of an index component that is identical in structure, format, and function to the prime index of a key-sequenced cluster. The data component in terms of control intervals and control information, also is identical to the data component of a key-sequenced file. However, a data record in the alternate index is unlike a data record in a key-sequenced file. It has a special standardized format. Each data record is of variable length. It contains an alternate key value, at least one pointer to a data record in the base cluster, and the following self-describing information:

- The number and length of pointers
- The length of the alternate key
- Whether the alternate index is built over a key-sequenced or entry-sequenced file.

Gaining Access to the Original File

In order to gain access to the original file (base cluster) with an alternate index you need to establish a *path* between them. The path relates or sets up an association between the alternate index and the base cluster.

Figure 18 illustrates this; it shows two alternate indexes associated with the same key-sequenced base cluster. Alternate index 1 is associated with the base cluster through path 1; alternate index 2 is associated with the base cluster through path 2. The two alternate indexes make the information in the base data record readily available to you. Path 1 allows you to gain access to a data record in the base cluster by 'Social Security Number,' while path 2 allows you to gain access to a data record in the base cluster by 'Department Number'.

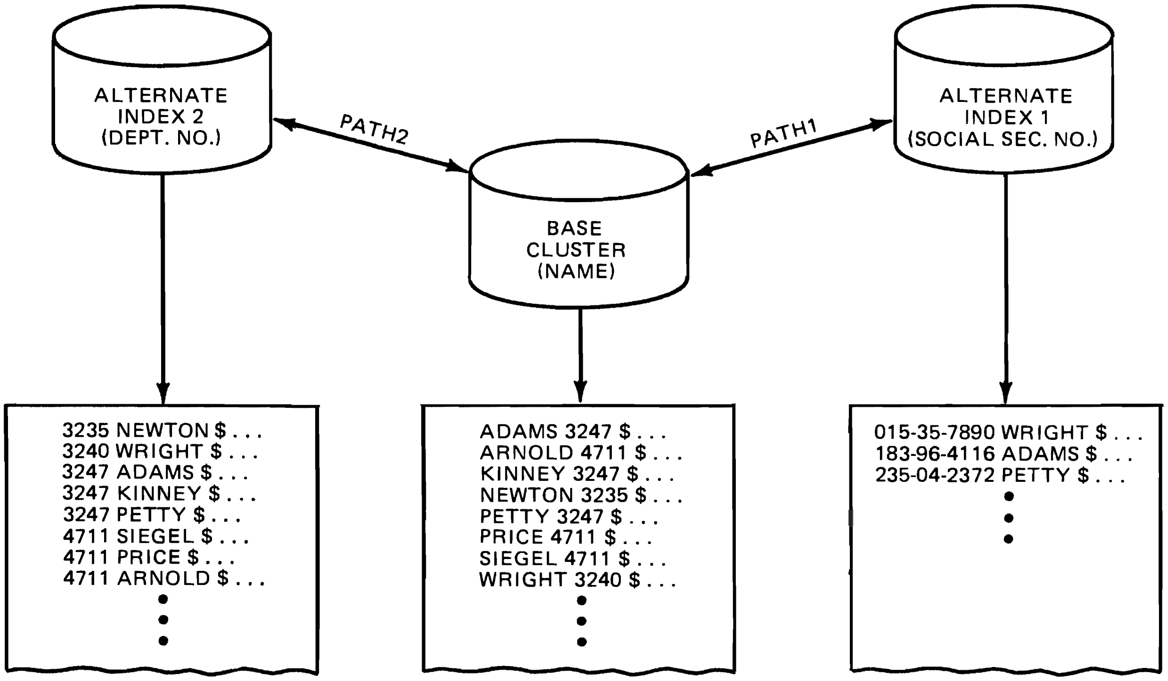


Figure 18. Two Alternate Indexes Over a Single Key-Sequenced File

Before continuing our discussion of VSE/VSAM, let us review a summary of the differences (see Figure 19) among these files.

Key-Sequenced File	Entry-Sequenced File	Relative Record File
Records are fixed- or variable length.	Records are fixed- or variable length.	Records are fixed-length only.
Records are in collating sequence by a key field.	Records are in the order in which they were entered.	Records are in order based on relative position.
Sequential or direct access. Direct access to a data record is by key or by relative byte address.	Sequential or direct access. Direct access to a data record is by relative byte address only (unless you build an alternate index over the file).	Sequential or direct access. Direct access to a data record is by relative record number (which is treated as a key).
A record's relative byte address can change.	A record's relative byte address cannot change.	A slot's relative record number cannot change.
Free space in the file is used to add records and to change their length in place.	Space at the end of the file is used for adding records; their length cannot be changed in place.	Empty slots in the file are used for adding records. Record length cannot be changed.
May have one or more alternate indexes.	May have one or more alternate indexes.	May not have alternate indexes.
Space given up by a deleted or shortened record is automatically reclaimed so it can be reused.	A record cannot be physically deleted but its space can be reused for a record of the same length.	Space given up by a deleted record can be reused by inserting a new record with the same relative record number.
Can have spanned records.	Can have spanned records.	Cannot have spanned records.

Figure 19. Differences Among Key-Sequenced, Entry-Sequenced, and Relative Record Files

The Processing of VSAM Files

Up to this point the discussion of VSE/VSAM has centered on identifying and explaining VSE/VSAM's structures. In this and succeeding chapters, the discussion will be concerned with how VSE/VSAM, you, and the system interact.

How VSE/VSAM Locates Data Records

VSE/VSAM always locates (accesses) a data record by using the data record's relative byte address. Three methods of accessing data records are available to you; they are keyed access, addressed access, and control-interval access. The file organization determines the manner in which a specific data record can be accessed.

- *Keyed access* is used with a key-sequenced and a relative record file organization. (An alternate index or path is treated like a key-sequenced file, except that addressed access is not allowed for an alternate-index path.) For a key-sequenced file, VSE/VSAM refers to a key in the index to access a data record. The index relates a key value to the relative byte address of a data record. (Recall that VSE/VSAM assigns relative byte addresses to the individual data records in a file.)

  For a relative record file, VSE/VSAM uses a record's relative-record number as the key to locate the record.

- *Addressed access* is used with a key-sequenced and entry-sequenced file organization. VSE/VSAM uses the relative byte address of a record to locate its position in a file.

- *Control-interval access* is used to access control intervals instead of data records. It is only used for special types of processing. The relative byte address of a control interval is used to access the contents of a control interval. If you use this type of access, certain tasks usually performed by VSE/VSAM become your responsibility; for example, you must keep the control information at the end of the control interval up to date.

VSE/VSAM can process records in each of its three file organizations in a sequential or direct mode. In either mode, you can retrieve, and update records of all three file organizations. Additionally, you can insert and delete the records of a key-sequenced and relative record file but not of an entry-sequenced file. (You can add records to the end of an entry-sequenced file.)

- *Sequential processing* of a record depends on the order of the previously processed record; direct processing does not. Records retrieved by key are in key sequence (key-sequenced file), records retrieved by relative-record number are in numerical order (relative-record file), and records retrieved by address are in entry sequence (entry-sequenced file). To retrieve or store records sequentially after initial positioning, you do not need to specify a key, relative-record number, or relative byte address; VSE/VSAM automatically retrieves or stores the next record in order. (VSE/VSAM also provides for sequential backward processing. Instead of retrieving the next record, VSE/VSAM retrieves the previous record.)

- *Direct processing* of a record does not depend on the key, relative-record number, or relative byte address of any previously retrieved record. You must identify the record to be retrieved or stored by its key, relative-record number, or relative byte address.

VSE/VSAM also allows you to switch back and forth between the modes of processing. For example, you can process records sequentially, then switch to direct processing, etc.

VSE/VSAM's Approach to Space and File Management

One of the requirements for an access method is that it should assist you, as much as possible in the task of managing space and creating files. VSE/VSAM meets this requirement by the way it uses a VSAM *catalog.* A VSAM catalog is a central file that collects information about storage spaces and files; it is created from the information you provide (via Access Method Services commands) to VSE/VSAM. A catalog contains information about:

- Data spaces - The catalog describes space allocated for files. It tells where and how much data space is available, the serial number and device characteristics of the volume, etc. Whenever data space is allocated to a file, VSE/VSAM automatically updates the applicable space information in the catalog. This is referred to as space management.
- VSAM files - The catalog describes a file's location and its attributes (for example, its record size and key location). Dynamic statistics about the file such as the number of records inserted since the file was created, the number of control intervals which have been split, etc., are also kept in the catalog. These statistics provide you with the information you need in making a decision to reorganize your files or alter the type of processing you are performing to gain performance improvements. This is referred to as file management.

A VSAM catalog also helps to simplify your coding requirements because much of the required information for processing data is in the catalog. You do not have to specify it in your processing program.

A catalog exists on a single volume. Space that is not reserved on the volume for the catalog's use can be made available for future expansion of the catalog, if that becomes necessary, or for other uses.

An entry for every one of your VSAM files and data spaces must exist in a VSAM catalog (as well as an entry that describes the catalog itself): as a matter of fact, a file or data space cannot exist in VSE/VSAM until an entry is made for it in a catalog. (Making an entry in a catalog for a file or data space iš known as defining them.) You cannot, for example, load records into a file until you have first defined the file in a catalog.

When you establish a VSAM data space on a previously unused (by VSE/VSAM) storage volume, you set up a relationship between that specific storage volume and the catalog on which it is defined; that storage volume now "belongs to" or is "owned by" the catalog in which it is defined. Any additional data spaces or files created on that storage volume must also be defined in that same catalog.

Master and User Catalogs

At this point you may be wondering about some of the possible negative results of having all space and file information stored on a single catalog volume. For example, what if an unexpected problem occurred to the catalog volume; how could you access your data? What if you had several entries in your catalog; would your total processing time increase as VSE/VSAM searched for a specific entry in the catalog? VSE/VSAM answers these (and other) concerns by allowing you to have multiple catalogs. Each catalog exists on a single volume; it is independent and controls exclusively its own data spaces and files.

This structure (multiple catalogs) provides for an increased measure of data integrity because it minimizes the effect of a catalog being inoperative or unavailable. If a failure occurs in a catalog, then only the information connected with the faulty catalog is lost and needs to be reconstructed. VSE/VSAM provides you with several options to recover from catalog failures; for example, you can define a catalog with the recoverable attribute. This attribute enables VSE/VSAM to record the information about a specific volume (and its files) in two places: on that volume, as well as in the catalog itself. Now if for any reason the catalog is destroyed, you can reconstruct it by using the duplicate information that was recorded on the volume(s) that it owns. Additional options that are available for catalog recovery are discussed in "VSE/VSAM Data Protection."

VSE/VSAM requires that one of the catalogs in a multi-catalog environment must be selected as the master catalog. All the remaining catalogs are called user catalogs; they are pointed to by the master catalog. (If you only have one catalog in your system, it is the master catalog.) By placing information about your files and storage volumes into user catalogs, you decentralize control and reduce the time required to search a given catalog. Figure 20 shows how everything in VSE/VSAM is linked back, either directly or indirectly, to the master catalog.It also shows that you can catalog a VSAM file or nonVSAM file in either a master or user catalog.

Catalogs also enable you to transfer volumes between DOS/VSE, DOS/VS and OS/VS systems. They are compatible with these operating systems.

You can also specify that one of the user catalogs is to be a job catalog; it will be used to reference all VSAM files in the current job. You have the option of overriding this specification at any time.

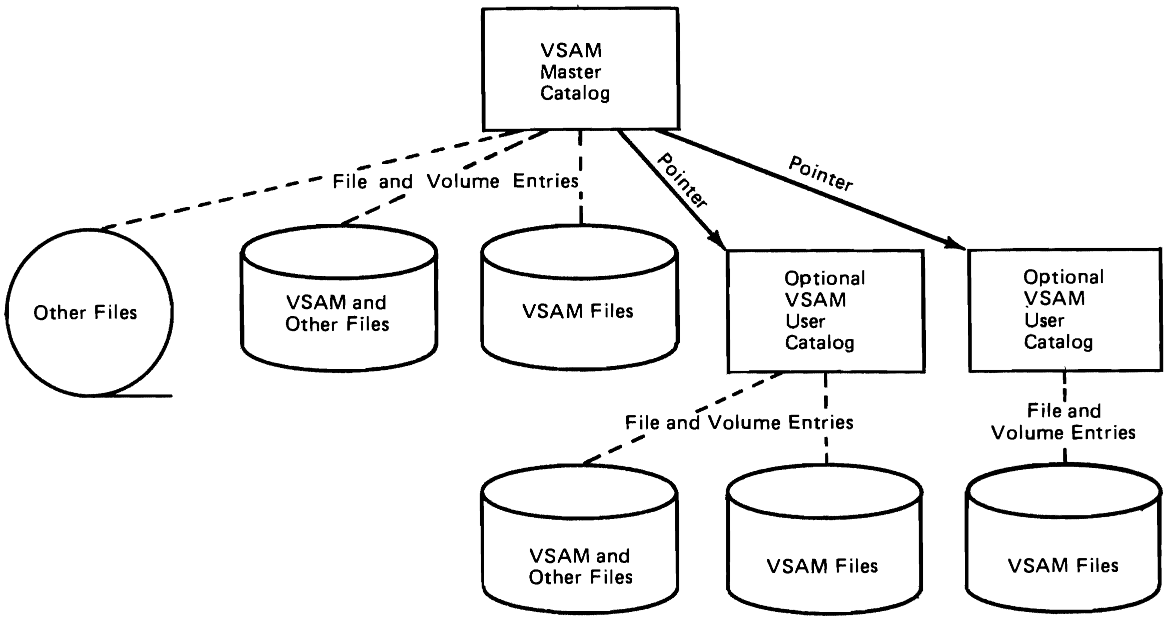


Figure 20. Catalog Relationships

How to Communicate with VSE/VSAM

You communicate with VSE/VSAM by using:

- Job control statements -- Because most of the information normally coded with job control statements is available to VSE/VSAM in the catalog, you need to specify only a minimum number of job control statements.
- Access Method Services commands -- The commands in this multifunction utility program simplify the coding requirements of your processing program by allowing you to create files separately from your processing program. Additionally, information about your files (and their storage volumes), is entered into a catalog to simplify VSE/VSAM's management of files and space. You can also use these commands to print files, delete files, backup and restore files and catalog entries, list the catalog entries, etc.
- Processing programs -- After you create your VSAM files using Access Method Services, you can process them with the following processing languages: assembler, COBOL/VS, PL/I Optimizer, and RPGII/VS. (Only assembler supports all VSE/VSAM functions.) If your current processing program uses ISAM to access your files you can continue to use it with the aid of VSE/VSAM's ISAM Interface Program (IIP). This is a fast way for ISAM users to begin to convert to using VSE/VSAM. See "Preparing for VSE/VSAM" for more information on IIP.

Figure 21 illustrates how you communicate with VSE/VSAM and how VSE/VSAM interacts with auxiliary storage. It shows that by using VSE/VSAM you divide your coding requirements between Access Method Services commands and a VSE/VSAM (or ISAM) processing program. VSE/VSAM will create your files and manage VSAM data space (via the Access Method Services commands) and also processes your data (via a processing program). Notice that the VSAM catalog is situated between VSE/VSAM and your data; VSE/VSAM uses the VSAM catalog to help control and manage space and files for you.

From the preceding discussion you can see that the coding requirements in your processing program have been simplified because:

1. Your file attributes are specified separately from your processing program, and
2. VSE/VSAM builds and uses catalog(s) to keep track of your storage volumes and data files.

Let us now examine job control statements, Access Method Services commands, and VSE/VSAM processing programs.

Job Control Statements For VSE/VSAM

With VSE/VSAM, you need to specify only a minimum number of job control statements (and parameters) because the information required by the system to check the location and characteristics of files is stored in the VSAM catalog. Also, Access Method Services commands, and not job control statements, allow you to define space for your data files.

See *VSE/VSAM Programmer's Reference* for an explanation of the job control statements used by VSE/VSAM.

VSE/VSAM's Access Method Services Commands

VSE/VSAM's Access Method Services is a multifunction utility program that provides you with commands to perform the following functions:

- Establish catalog(s)
- Create data spaces
- Create VSAM files and load records into them
- Build one or more alternate indexes for a file
- Convert nonVSAM files to VSAM files
- Create backup copies of VSAM files and their associated catalog entries
- Move catalogs and files from one system to another system
- Print catalog entries
- Print, copy or reorganize VSAM files
- Alter file attributes or definitions
- Delete VSAM files, data spaces, and catalogs
- Recover from damage to files or catalogs
- Cancel a job or the current job step. (Available for VSE/VSAM Release 2.)

Two types of Access Method Services commands are available to you: *functional commands* that are used to request the actual work (for example, defining a file or moving a catalog) and *modal commands* that allow the conditional execution of functional commands.

The functional commands are:

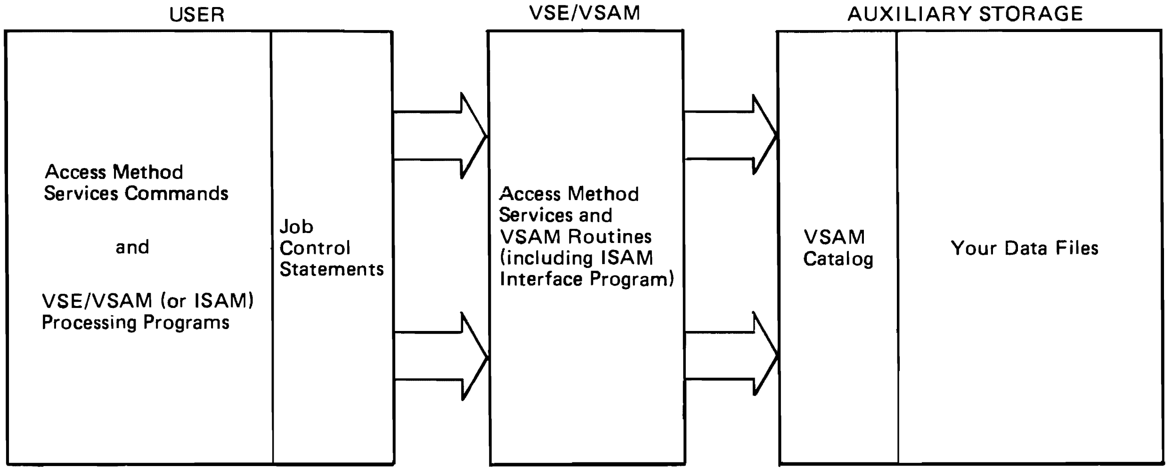

USER
Access Method Services Commands
and
VSE/VSAM (or ISAM) Processing Programs
Job Control Statements
VSE/VSAM
Access Method Services and VSAM Routines (including ISAM Interface Program)
AUXILIARY STORAGE
VSAM Catalog
Your Data Files

Figure 21. Communicating with VSE/VSAM

- DEFINE, which is used to define all objects (catalogs, files, alternate indexes, paths, VSAM data spaces, and non-VSAM files); it makes entries for them in a VSAM catalog and except for paths and non-VSAM files, allocates space for them on a direct access storage volume. All of the attributes of a file are specified in a DEFINE command.
- ALTER, which is used to change one or more of the previously-defined attributes of an object. VSE/VSAM alters attributes by replacing the old attributes in the catalog entry with the new ones that you specified. Most of the parameters of the ALTER command are the same as those of the DEFINE command.
- DELETE, which is used to delete catalogs, and remove the catalog entries for any previously-defined objects (this, in effect, causes the objects to cease to exist) or to free any associated data space. You can specify that VSE/VSAM overwrite the freed space with zeros to provide an additional measure of data security.
- BLDINDEX, which constructs one or more alternate indexes for existing key-sequenced and entry-sequenced files.
- REPRO, which is used to copy files, to convert sequential files on tape or disk and indexed-sequential files to VSE/VSAM format, to convert VSE/VSAM and indexed-sequential files to sequential format, to create backup copies of VSAM catalogs, to reload a VSAM catalog from a backup copy, or to reorganize files.
- EXPORT, which is used to create a copy of a VSAM file for backup or to make a file or user catalog portable so that it can be used on another system. The transporting volume can be magnetic tape or disk. You can transfer the file either permanently or temporarily.
- IMPORT, which is used to reestablish a backup copy of a VSAM file or to make a file or catalog that was previously exported from one system available for use in another system.
- LISTCAT, which is used to list entries from a catalog. The listing will show information about objects defined in the catalog, such as: attributes of the object, volume information, statistics, etc. The listing can be tailored to meet your needs by limiting the number of entries to be printed and by limiting the amount of information to be printed for each entry.
- PRINT, which is used to print all or the specified range of logical records of a VSAM or nonVSAM file. Three formats are available: each byte printed as a single character, each byte printed as two hexadecimal digits, and a combination of the previous two (side by side).
- EXPORTRA, which retrieves VSAM files and catalog entries that are not accessible from a VSAM catalog because of a processing error. (A VSAM catalog can be defined with a recoverable attribute. This option places critical catalog information into a catalog recovery area (CRA) for the subsequent recovery of inaccessible data.) EXPORTRA is also used for backing-up all the VSAM files and catalog entries on the entire volume, or set of volumes, even if the catalog is accessible.
- IMPORTRA, which is used to make the data recovered through the EXPORTRA function again available to you.

- LISTCRA, which is used to diagnose suspected problems in recoverable VSAM catalogs.
- RESETCAT, which is used to reset a recoverable catalog to the level of its owned volume(s). (Recall that a volume(s) is "owned by" the catalog in which it is defined.) The owned volume's CRA information is used to synchronize the catalog with the volume(s).
- VERIFY, which is used to restore access to data if a file was not closed successfully the last time it was processed.
- CANCEL, which is used to cancel either a job or the current job step. (Available for VSE/VSAM Release 2.)
- BACKUP, which is used to create a backup copy of a file(s).
- RESTORE, which is used to restore a file(s) that was backed up via the BACKUP command.

The modal commands, ones that control command execution and establish options, are:

- IF, which tests a condition code and executes according to the results of the test. IF is followed by THEN and ELSE clauses which specify alternative actions.
- DO and END which denote the beginning and ending of a functional command sequence (normally within a THEN or ELSE clause).
- SET, which changes condition codes.
- PARM, which specifies diagnostic aids and printed output options and changes input record margins.

Each Access Method Services command has a certain minimum number of parameters that are mandatory. For example, when you use the DEFINE command to create a master catalog you only need to specify four parameters; these four parameters specify all the information VSE/VSAM needs to establish the master catalog. You can of course specify any number of the other optional parameters that are available for data integrity, security, ease of use, etc.

Processing Data with VSAM Macros

Data is loaded into VSAM file(s) after catalog(s), VSAM data space(s), and catalog entries for file(s) and data space(s) have been created by means of job control statements and the DEFINE commands of Access Method Services. You load data into a VSAM file either with a VSE/VSAM or ISAM processing program or with the REPRO command of Access Method Services.

After the VSAM file is loaded, you can begin to process it with a VSE/VSAM or ISAM program. This section examines the role of VSAM macro instructions in the processing of data.

VSAM macros (and the appropriate job control statements) provide VSE/VSAM with the information it needs to:

- *Connect and disconnect your processing program to and from a file.* Before a processing program can issue requests for access to a file, it must open the file for processing (OPEN macro). Opening a file causes VSE/VSAM to: have the necessary volume(s) mounted; construct the control blocks needed to process your requests; determine the processing options to be used; verify that the requestor is authorized to process the file; and verify that the file is available for the kind of processing specified in the file's sharing option. When VSE/VSAM successfully completes these steps, the file is accessible or open to processing by your program.

  Disconnecting a program from a file (CLOSE macro) causes VSE/VSAM to: write out any buffers whose contents have changed and which have not already been written out; update the appropriate catalog(s); and free any resources obtained when the file was opened. If the file cannot be closed properly, VSE/VSAM issues a warning message to the operator.

  VSE/VSAM also provides you with a macro instruction (TCLOSE macro) that closes a file as before except that it leaves the program and file connected so that you can resume processing without reopening the file. In effect, this is a temporary close macro instruction. As one of the functions of both CLOSE and TCLOSE is to update the catalog to reflect the then-current status of the file, and if VSE/VSAM is subsequently unable to close a file properly, you can use the VERIFY command of Access Method Services to recover data that ordinarily would have been permanently lost.
- *Specify information that associates a program and its data.* These macros (ACB, EXLST, RPL, and GENCB) describe the kind of processing to be done. VSE/VSAM uses the information derived from your macro specifications (and the catalog entries) to create control blocks that contain the needed information to process data. The ACB, EXLST, and RPL macros create control blocks and lists at program assembly time; the

GENCB macro creates control blocks and lists at program execution time.

- *Manipulate the information that associates a program and its data.* These macros (MODCB, SHOWCB, TESTCB) are used to modify, display, and test the contents of the VSE/VSAM control blocks at program execution time.

  You can modify the contents of control blocks by specifying a new value for them in the same way you defined them originally. You can examine the contents of fields in control blocks by providing VSE/VSAM with an area to display the requested fields. You can test the contents of a field or combination of fields in control blocks for a particular value and alter the sequence of your processing steps as a result of the test. This operation is similar to a branch instruction. For example, you can test the error codes set in a control block or the attributes of a file, such as record length.
- *Request access to a file.* These macros (GET, PUT, POINT, ERASE, ENDREQ) initiate the transfer of data between auxiliary and virtual storage. You can cause: a record to be retrieved; a record to be stored; a record previously retrieved for update to be deleted from a key-sequenced file; VSE/VSAM to position at the desired record; or a specified request to be terminated.
- *Share resources among files.* Normally, buffers and control blocks are allocated to a file at the time the file is opened; they are freed when the file is closed. As long as the file is open, these buffers and control blocks cannot be used by any other file. The Shared Resources facility (BLDVRP, DLVRP, WRTBFR macros), however, allows you to share buffers, I/O control blocks, and channel programs among several VSAM files within a partition. These buffers and control blocks are allocated out of a common resource pool at the time you issue an I/O request for a file. When the request is satisfied, the same buffers and control blocks can be assigned to another file.

  Sharing these resources optimizes their use and also reduces the amount of virtual storage required per partition. The facility is especially useful when many VSAM files are open and, therefore, it becomes difficult to predict the amount of activity for a given period, or when a transaction refers to several files. In addition, facilities are provided for managing I/O buffers. VSE/VSAM buffer management can be used to increase the processing speed of VSAM files that have unpredictable activity.
- *Display catalog information about VSAM files.* You can retrieve selected data from the VSAM catalog with the SHOWCAT macro. The information, which is based on a specified object (cluster, alternate index, path, etc.), is returned in a work area that you must supply.

Preparing for VSE/VSAM

This chapter indicates the machines and programming languages that VSE/VSAM can be used with. Considerations on using ISAM programs in a VSE/VSAM environment are also discussed in this chapter.

Machines With Which VSE/VSAM Can Be Used

VSE/VSAM will operate on any system (minimum real storage size of 128K bytes) and any direct access storage device supported by DOS/VSE, (Program Number 5745-020). VSE/VSAM is designed to take full advantage of the benefits of virtual storage.

What's New With VSE/VSAM

VSE/VSAM Release 1 provides current users of VSAM several new enhancements that may impact portability to OS/VS VSAM and DOS/VS VSAM. Users of VSE/VSAM can achieve file and volume portability to the other environments that support VSAM by using only those functions, types of files, and devices that are supported by all the affected environments. Additionally, the VSE/VSAM user must in certain cases specify new VSE/VSAM parameters in order to produce output suitable for processing by all environments. These considerations are addressed in this section of this publication and in appropriate sections of other VSE/VSAM publications.

The new enhancements in VSE/VSAM are as follows:

NOIMBED For Catalogs

Discussion: Previously, the sequence set of a VSAM catalog was always imbedded in the catalog's data component. This provided fast access to catalog records but less than optimal disk space utilization. You now have the opportunity to decide between the time or space performance trade-off.

Compatibility with DOS/VS VSAM: Catalogs defined with the disk-space optimization enhancement (NOIMBED) cannot be processed on DOS/VS VSAM. A conversion process can be used, however, to allow processing on DOS/VS VSAM. See *VSE/VSAM Programmer's Reference* for information on the conversion process.

Compatibility with OS/VS VSAM: Catalogs defined with the disk-space optimization enhancement (NOIMBED) cannot be processed on OS/VS. A conversion process can be used, however, to make the catalogs suitable for processing by OS/VS. See *VSE/VSAM Programmer's Reference* for information on the conversion process.

Control Interval Split Integrity

Discussion: If a control interval split on a key-sequenced file (including alternate indexes and catalogs) is interrupted by a system failure, a duplicate record condition may occur. VSE/VSAM checks to determine whether duplicate records are present in the new and original control interval. If this condition exists, VSE/VSAM will erase the original versions of the copied records, except when the key-sequenced file is being processed by addressed or control interval (CNV) access.

Compatibility with DOS/VS VSAM: Duplicate records caused by a failure during a control interval split on VSE/VSAM may cause a processing failure if the applicable file is processed by DOS/VS VSAM prior to being re-processed by VSE/VSAM.

Compatibility with OS/VS VSAM: Duplicate records caused by a failure during a control interval split on VSE/VSAM may cause a processing failure if the applicable file is processed by OS/VS VSAM prior to being re-processed by VSE/VSAM (unless the appropriate PTF is installed on the OS/VS system).

Fixed Block Architecture Support

Discussion: A new Fixed Block Architecture is supported by VSE/VSAM. The new architecture (using native VSE/VSAM or the ISAM Interface Program) is transparent to your programs, with the exception that the specification of disk addresses on the direct access storage device is in terms of block numbers rather than CCHH or relative track number.

In the same way, the size of a data space or file can only be specified in terms of "number of records" or "number of blocks" and not in terms of cylinders or tracks. The specification of disk addresses is generally necessary in VSE/VSAM for the allocation of data spaces. Any user-displayed information concerning disk addresses or extent sizes is also in units of blocks.

The physical record of a Fixed Block Architecture device is a 512 byte block of data that is transparent to the VSE/VSAM user. The transfer unit between virtual storage and a Fixed Block Architecture device continues to be a control interval.

Compatibility with DOS/VS VSAM: Catalog entries for files on a Fixed Block Architecture device cannot be processed by DOS/VS VSAM. This does not affect the processing of catalog entries for a non-Fixed Block Architecture device.

Files on a Fixed Block Architecture device can be transferred from (via EXPORT and IMPORT) VSE/VSAM to a CKD device that is using DOS/VS VSAM. Conversely, files can be transferred from DOS/VS VSAM to a Fixed Block Architecture device.

Compatibility with OS/VS VSAM: Files on a Fixed Block Architecture device cannot be processed by OS/VS. This does not affect the processing of catalog entries or files for a non-Fixed Block Architecture device.

Files on a Fixed Block Architecture device can be transferred from (via EXPORT and IMPORT) VSE/VSAM to a CKD device on an OS/VS system. Conversely, files can be transferred from a CKD device on OS/VS to a Fixed Block Architecture device.

Data Space Classification Support

Discussion: VSE/VSAM data space can be classified to direct the suballocation of data space to VSE/VSAM objects and maximize performance. Space classes can be assigned by you when data space is defined; you may then request that a particular class of space be used when a file is defined. This support is particularly useful for controlling high-performance areas such as fixed-head areas on the applicable IBM direct access storage devices.

Compatibility with DOS/VS VSAM: Space class specifications are not supported by DOS/VS, but a file, data space, or volume established with space classes under VSE/VSAM can be processed by DOS/VS VSAM.

DOS/VS VSAM files can be transported to VSE/VSAM volumes defined with special classes.

Compatibility with OS/VS VSAM: Space class specifications are not supported by OS/VS, but a file, data space, or volume established with space classes under VSE/VSAM can be processed by OS/VS VSAM.

OS/VS VSAM files can be transported to VSE/VSAM volumes defined with special classes.

New Physical-Record Size

Discussion: The physical-record sizes previously used were 0.5K, 1K, 2K, and 4K bytes. This set is now extended to include all multiples of 0.5K from 0.5K to 8K bytes (applies to CKD devices only). With this larger set, VSE/VSAM is able to choose physical-record sizes that make better use of track capacity. Note that, as an index control interval is always one physical record, index control interval size is no longer constrained to the 0.5K, 1K, 2K, and 4K limits, resulting in better index space utilization.

Compatibility with DOS/VS VSAM: No change from compatibility as currently available to the user.

Compatibility with OS/VS VSAM: Files created by VSE/VSAM with a new physical-record sizes cannot be directly processed by OS/VS. File portability between VSE/VSAM and OS/VS via EXPORT, EXPORTRA, IMPORT, and IMPORTRA is not impacted by the new physical-record size enhancement.

New SHOWCAT Operand

Discussion: The EXTOPT operand has been added to the SHOWCAT macro. If you specify:

- EXTOPT=HARBADS, the file's high allocated relative byte address is returned to you.
- EXTOPT=VOLSER, the volume serial number of the file's primary allocation volume is returned to you.

Compatibility with DOS/VS VSAM: The operand is not supported by DOS/VS VSAM.

Compatibility with OS/VS VSAM: The operand is not supported by OS/VS VSAM.

Improved EXPORT/EXPORTRA Performance

Discussion: Enhanced SAM and VSAM I/O buffering is provided for EXPORT, IMPORT, EXPORTRA, and IMPORTRA. This can provide a significant reduction in I/O wait time (due to overlapped I/O) which in turn can significantly reduce job time.

For EXPORT/EXPORTRA, the CIMODE option allows data to be stored on the portable media in control interval format and to be retrieved in control interval mode. This significantly reduces CPU overhead and thus enhances job and overall system performance.

Virtual storage requirements for executing the EXPORTRA and IMPORT commands have been greatly reduced.

Compatibility with DOS/VS VSAM: Files from DOS/VS VSAM, when transferred (via IMPORT or IMPORTRA) into VSE/VSAM, automatically take advantage of the IMPORT or IMPORTRA command's I/O buffering performance enhancement. Files from VSE/VSAM, when transferred (via EXPORT or

EXPORTRA) into DOS/VS VSAM, automatically take advantage of the EXPORT or EXPORTRA command's I/O buffering performance enhancement.

The CIMODE (control interval mode) data-retrieval optional performance enhancement cannot be used on DOS/VS VSAM.

Compatibility with OS/VS VSAM: Files from OS/VS, when transferred (via IMPORT or IMPORTRA) into VSE/VSAM, automatically take advantage of the IMPORT or IMPORTRA command's I/O buffering performance enhancement. Files from VSE/VSAM, when transferred (via EXPORT or EXPORTRA) into OS/VS, automatically take advantage of the EXPORT or EXPORTRA command's I/O buffering performance enhancement.

The CIMODE (control interval mode) data-retrieval optional performance enhancement cannot be used in OS/VS.

What's New for VSE/VSAM Release 2?

VSE/VSAM Release 2 provides the following enhancements to previous releases (the previous general statements about VSE/VSAM Release 1 compatibility also apply to VSE/VSAM Release 2 compatibility):

DASD Sharing

Discussion: A user can share VSAM files across DOS/VSE systems through the use of the Access Method Services SHAREOPTIONS parameter. Any files to be shared across systems must reside on a 33xx or Fixed Block Architecture device.

Compatibility with Previous Releases: The sharing of VSAM files across DOS/VSE systems does not affect VSAM file or catalog format and therefore has no impact on previous release data compatibility. Note that the validity of VSAM files and catalogs is only ensured if *all* systems accessing the shared DASD devices are using VSE/VSAM Release 2.

Compatibility with OS/VS VSAM: Beginning with VSE/VSAM Release 2, two values can still be specified with the SHAREOPTION parameter; the first value specified is now used to indicate the way a file can be shared across both partitions and systems in DOS/VSE. The second SHAREOPTION value is the same as before; it is allowed for compatibility with OS/VS.

The sharing of VSAM files across DOS/VSE systems does not affect VSAM file or catalog format and therefore has no impact on previous release data compatibility. Note that the validity of VSAM files and catalogs is only ensured if *all* systems accessing the shared DASD devices are using VSE/VSAM Release 2.

SHAREOPTIONS 4 Enhancement

Discussion: Improved VSE/VSAM SHAREOPTION 4 performance.

Compatibility with Previous Releases: The new SHAREOPTION 4 support prevents a SHAREOPTION 4 key-sequenced file from being opened for output (MACRF=OUT) together with keyed (MACRF=KEY) and addressed (MACRF=ADR) or keyed (MACRF=KEY) and control interval (MACRF=CNV) access. Any previous programs to be processed using VSE/VSAM Release 2 must be changed to conform to this restriction.

Compatibility with OS/VS VSAM: The new SHAREOPTION 4 support does not impact file or volume portability to OS/VS.

Catalog Management Performance Improvement

Discussion: Improved catalog management performance as a result of changes to an internal algorithm.

Compatibility with Previous Releases: No change from compatibility as currently available to the user.

Compatibility with OS/VS: No change from compatibility as currently available to the user.

Access Method Services CANCEL Command

Discussion: The addition of the CANCEL command to Access Method Services. This command gives the user the option of canceling a job or the current job step.

Compatibility with Previous Releases: The command is not supported by previous releases.

Compatibility with OS/VS VSAM: The command is not supported by OS/VS VSAM.

Control Area Split Integrity

Discussion: The automatic correction of a duplicate record condition caused by a system failure during a control area split.

Compatibility with Previous Releases: Duplicate records caused by a failure during a control area split on Release 2 of VSE/VSAM may cause a processing failure if the applicable file is processed by DOS/VS VSAM prior to being re-processed by VSE/VSAM. (VSE/VSAM Release 1 is able to process the file.)

Compatibility with OS/VS VSAM: Duplicate records caused by a failure during a control area split on VSE/VSAM may cause a processing failure if the applicable file is processed by OS/VS VSAM prior to being re-processed by VSE/VSAM (unless the appropriate PTF is installed on the OS/VS system).

Dedicated VSAM Volume

Discussion: The ability to readily specify via the Access Method Services DEDICATE parameter that the entire space of a volume or the remaining free space of a volume is to be owned by VSAM.

Compatibility with Previous Releases: A volume allocated to VSAM with the new DEDICATE parameter can be processed with any prior release of VSE/VSAM and DOS/VS VSAM.

Compatibility with OS/VS VSAM: The DEDICATE parameter is not supported by OS/VS, however, a volume allocated to VSAM with the DEDICATE parameter can be processed by OS/VS.

Additional Classes of Space

Discussion: An additional 5 classes of space can now be defined by the user to allow further subdivision of data space. This makes a total of 8 classes that can be specified.

Compatibility with Previous Releases: Files specifying new space classes may be moved (along with their catalog records) to a previous release of DOS/VSE as long as a secondary allocation of the file (requiring a new class) is not needed. Files that call for a new USECLASS cannot be successfully transferred to a prior release via the IMPORT and IMPORTRA commands.

Compatibility with OS/VS VSAM: Space class specifications are not supported by OS/VS, but a file, data space, or volume established with space classes under VSE/VSAM can be processed by OS/VS VSAM.

OS/VS VSAM files can be transported to VSE/VSAM volumes defined with special classes.

Dynamic Files

Discussion: The ability to define a file into the VSAM catalog without allocating any space to it. Space does not have to be reserved in advance by the user; space is automatically allocated to the file as it is needed. This function allows the user better use of DASD space in creating reuseable files.

Compatibility with Previous Releases: Files created by this function cannot be processed on prior releases.

Compatibility with OS/VS VSAM: Files created by this function cannot be processed by OS/VS.

Disposition Parameters for a File

Discussion: Allows users to specify (1) status options for a file at OPEN time (for example, resetting the file) via the DLBL job control statements (2) the disposition of the file at CLOSE time (for example, resetting the file) via the ACB macro and the DLBL job control statement.

Compatibility with Previous Releases: This function creates no data incompatibilities.

Compatibility with OS/VS VSAM: This function creates no data incompatibilities.

Default Models

Discussion: Allows users of Access Method Services to choose their own parameter defaults in place of the usual system defaults.

Compatibility with Previous Releases: Default models are not supported by prior releases; however, the resultant file and catalog data can be processed by prior releases.

Compatibility with OS/VS VSAM: Default models are not supported by OS/VS VSAM; however, the resultant file and catalog data can be processed by OS/VS.

Default Volumes

Discussion: Allows users to omit explicit volume lists in the DEFINE CLUSTER and DEFINE ALTERNATEINDEX commands. A new parameter (DEFAULTVOLUMES) is also provided in the IMPORT and IMPORTRA commands to allow users to override the exported volumes list in a new way. The required volumes are selected from the volumes list associated with the default model established for the object being defined.

Compatibility with Previous Releases: The command functions are not supported; however, the resulting file and catalog data can be processed.

Compatibility with OS/VS VSAM: The command functions are not supported; however, the resulting file and catalog data can be processed by OS/VS.

Job Control Simplification

Discussion: The simplification of job control specifications by eliminating in most cases the need to specify ASSGN and EXTENT job control statements. Additionally, for Access Method Services commands, the DLBL job control statement can also be eliminated in many cases.

Compatibility with Previous Releases: Unpredictable results.

Compatibility with OS/VS VSAM: Not applicable.

Partition/CPU Independence

Discussion: Provides the user with an easier way of avoiding the possibility of two or more jobs encountering the same workfile file-IDs in different partitions/central processors running at the same time.

Compatibility with Previous Releases: Not supported by previous releases; however the resultant file and catalog data can be processed by prior releases.

Compatibility with OS/VS VSAM: Not applicable to OS/VS VSAM; however, the resultant file and catalog data can be processed by OS/VS.

Document Subset

Discussion: The availability of a shortened (subset) version of the current *Using VSE/VSAM Commands and Macros* book. This book shows in an uncomplicated way, how to use VSAM effectively. It is especially useful to users who are unfamiliar with VSAM because it incorporates and is structured to take advantage of the following new support for Access Method Services:

- The simplified job control specifications and requirements.
- The assumption that certain complex DEFINE parameters need no explanation because they are previously-specified by the system programmer as defaults (via default models).
- Emphasis on the use of default volumes and dedicated VSAM volumes. Other miscellaneous functions that are considered to be used primarily by system programmers are not documented in this book.

VSE/VSAM Space Management for SAM Feature

Additional enhancements for space management are being provided in a separate feature package. These enhancements relate specifically to space management by VSAM for SAM files. They consist of:

- Definition and deletion of a SAM file in VSAM space (hereafter referred to as a SAM ESDS) via Access Method Services or implicitly at OPEN/CLOSE time.
- Access of a SAM ESDS via DTFSD and DTFCP with DISK=YES for files in CI format, and DTFPH for DASD with MOUNTED=SINGLE for files in either CI format or non-CI format. SAM access is provided for all VSAM supported CKD and Fixed Block Architecture devices.
- Dynamic secondary allocation of a SAM ESDS during creation or extension of the file according to the secondary allocation size specifed at the definition of the file.
- Access of a SAM ESDS via ACB (that is, native VSAM) for files in CI format. An existing VSAM program that processes a VSAM entry-sequenced file can access a SAM ESDS without change (except for extending the file).

The following support provided in VSE/VSAM Release 2 is available for a SAM ESDS:

- Dynamic files
- Disposition parameters for a file
- Default models
- Default volumes
- Job control simplification
- Partition/CPU independence

Compatibility with OS/VS VSAM: SAM ESDSs and their catalog entries cannot be processed by OS/VS.

Backup/Restore Feature

The following two new commands are being added to Access Method Services to provide a fast backup and restore service with high performance and low processor utilization:

- The BACKUP command is used to produce a backup copy of a file(s).
- The RESTORE command is used to restore a file(s) previously backed up with the BACKUP command.

Backup/Restore provides backup onto start/stop tape and restoration from start/stop tape with high performance. This feature generally offers improved performance in terms of run-time as well as processor utilization compared to the Access Method Services EXPORT and IMPORT commands. Backup/Restore is also designed to support the streaming mode of the IBM 8809 magnetic tape unit. Some of the highlights of this feature are:

- Backup onto tape and restoration from tape for the following VSAM files:
  - Key-sequenced files
  - Entry-sequenced files
  - Relative record files
  - Alternate indexes
  - SAM files in VSAM space in CI format.
- The BACKUP and RESTORE commands allow multiple VSAM files to be handled with a single command.
- The BACKUP and RESTORE commands allow the user to specify a name (generic) that identifies a group of VSAM files rather than the individual entry names for each of those files.
- VSAM files can be excluded from a generic backup or restore operation.
- The BACKUP command allows the user to backup all VSAM files owned by a catalog with a single command.
- On restoration, interspersed and unwanted VSAM files of a backup file can be skipped.
- Volumes of the backup file can be selectively and randomly mounted during restoration to avoid unnecessary mounting of unused volumes.
- VSAM files can be restored to different locations than they originally occupied on a volume.
- Restoration can be to the same volume as the one on which the file originally resided or to a different volume. However, the new volume must be of the same device type as that of the original volume.
- Because the size and number of buffers for reading and writing determine the performance of this feature, the user can specify the size of the buffers in the BACKUP command and the number of buffers in both the BACKUP and RESTORE commands.
- The number of parameters in the BACKUP and RESTORE commands is minimal.

Compatibility with EXPORT/EXPORTRA: The tape format produced by BACKUP and required by RESTORE is different from the format of the portable files produced by the EXPORT and EXPORTRA commands. Therefore, portable files created by the EXPORT command cannot be processed by the RESTORE command nor can backup files created by BACKUP be processed by IMPORT. That is, the RESTORE command is the only correct counterpart to the BACKUP command. VSAM files created by VSE/VSAM can be processed by BACKUP and RESTORE without restrictions.

Compatibility with DOS/VS VSAM: Due to the physical record size support introduced in VSE/VSAM Release 1, Backup/Restore first verifies that a DOS/VS VSAM file to be backed up can be successfully restored. In those cases where restoration is not possible, VSE/VSAM ignores the back up request and, instead, issues an informational message to the user.

Compatibility with OS/VS VSAM: Due to the physical record size support introduced in VSE/VSAM Release 1, Backup/Restore first verifies that an OS/VS VSAM file to be backed up can be successfully restored. In those cases where restoration is not possible, VSE/VSAM ignores the back up request and, instead, issues an informational message to the user.

Storage Requirements

VSE/VSAM requires 296K bytes in the SVA (shared virtual area) for the VSE/VSAM routines loaded automatically into the SVA during IPL (initial program load). Those routines that are not eligible to be loaded into the SVA (for example, ISAM Interface Program) are loaded into a partition that has a minimum size of 30K bytes for basic requirements. The basic real storage requirement for the VSAM modules in the SVA is 22K bytes.

The Access Method Services modules cannot be loaded into the SVA. They have a partition-size requirement that depends on the functions required for the current job. The following table lists the approximate size for each Access Method Services function.

Command	Size in Bytes
ALTER	23,300
BLDINDEX	*24,000
DEFINE	57,500
DELETE	10,000
EXPORT	23,700
EXPORTRA	65,300
IMPORT	35,500
IMPORTRA	36,300
LISTCAT	52,400
LISTCRA	39,000
PRINT	8,000
REPRO	17,000
RESETCAT	125,000
VERIFY	2,000

* This does not include the storage required for sorting records.

Root modules, comprising 28K bytes, are loaded into the virtual address area when the user wants to perform any of the functions in the previous table. In addition to the root modules, the remaining Access Method Services substructure modules plus their required dynamic work areas are required to perform any function. The total virtual storage requirement for the substructure (including the root modules), which needs to be added to the above sizes, is 118K bytes. To operate efficiently, the Access Method Services working set requires approximately 64K bytes of real storage.

Programming Requirements

VSE/VSAM Release 1 (5746-AM2) is a licensed program that runs in a virtual storage environment under DOS/VSE (5745-020) and requires VSE/Advanced Functions (5746-XE8). VSE/VSAM consists of the following components:

- Virtual Storage Access Method, 5745-SC-VSM
- Access Method Services, 5745-SC-AMS
- VSAM/VTAM Common Macros, 5745-SC-VCM

Programming Languages With Which VSE/VSAM Can Be Used

You can use the DOS/VS Assembler, COBOL/VS, PL/I Optimizer and RPGII/VS languages to gain direct access to VSAM files. (Not all VSE/VSAM functions are available for programs written in PL/I Optimizer, COBOL/VS and RPGII/VS; if there is a function which your application must use, you can code your own assembler language subroutine to get the needed support.)

You can also use DOS/VS Assembler, RPGII/VS, COBOL/VS, and PL/I Optimizer languages to process VSAM files by way of the ISAM Interface Program.

Use of VSAM by Existing ISAM Programs

This section is mainly intended for users of ISAM who are converting to VSE/VSAM. VSE/VSAM's ISAM Interface Program (IIP) minimizes your conversion costs and scheduling problems by permitting programs coded to use ISAM to process VSAM files. To use the interface program, you must convert ISAM files to VSAM files (for which you can use Access Method Services), convert ISAM job control statements to VSE/VSAM job control statements, and ensure that your existing ISAM programs meet the restrictions for using the interface program.

Comparison of VSE/VSAM and ISAM

In most cases, if you use the performance options described in "VSE/VSAM's Performance Options" you can get better performance with VSE/VSAM while achieving essentially the same results that can be achieved with ISAM. You can also achieve results that cannot be achieved with ISAM. The use of your existing ISAM processing programs to process key-sequenced files depends upon the extent to which VSE/VSAM and ISAM are similar in what they do, as well as upon the limitations of the ISAM Interface Program itself.

The following sections describe the similarities and differences between VSE/VSAM and ISAM and indicate the functions of VSE/VSAM that have no counterpart in ISAM.

Differences Between VSE/VSAM and ISAM in Common Areas

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

- Index structure
- Relationship of index to data and adding records
- Deleting records
- Defining and loading a file

These differences are described in the paragraphs that follow.

Index structure. Both a VSAM key-sequenced file and an indexed sequential file 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 during processing. VSE/VSAM keeps individual index records in virtual storage during processing, the number depending on the amount of buffer space you provide. It optimizes the use of space by keeping those records it judges to be most useful at a particular time.

Relationship of index to data and adding records. The relationship of a VSE/VSAM index to the direct-access storage space whose records it controls is quite different from the corresponding relationship for ISAM, in particular with regard to overflow areas for record insertion.

ISAM keeps a two-part index entry for each primary track that a file 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 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.

VSE/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, records cannot be deleted until the file is reorganized; you must mark the records that are to be deleted. VSE/VSAM automatically reclaims the space in a key-sequenced file 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, VSE/VSAM requires less file reorganization than ISAM does.

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

VSE/VSAM Functions Not Available With ISAM

VSE/VSAM has capabilities that are not available with ISAM. Some of the capabilities provided by VSE/VSAM for processing indexed files are:

Processing of spanned and variable-length records. You can process a fixed-length, variable-length, and spanned record with VSE/VSAM; with ISAM you can only process a fixed-length record.

Skip sequential access. You can process a key-sequenced file 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 VSE/VSAM keep track of a position in the file, by means of a single access-method control block and without closing and reopening a file.

Addressed sequential access. You can retrieve and store the records of a key-sequenced file by relative-byte address, 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 VSE/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 files. Each alternate index provides another way to gain access to the same base file.

Key-range allocation. With a key-sequenced file, you can assign data to one or more volumes according to the ranges of key values in the data records. For example, if a file resides on three volumes, you might assign records with key A-E to the first volume, key F-M to the second, and key N-Z to the third.

Secondary allocation of storage space. When you define a VSAM file, you can specify the amount of direct-access storage space that is to be allocated automatically, when required, and if available, 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 for count-key-data devices and in terms of a number of data records or in terms of a number of blocks for Fixed Block Architecture devices.

Automatic file reorganization. VSE/VSAM partially reorganizes a key-sequenced file by splitting control intervals when not enough free space is available and by splitting control areas when it has no more free control intervals and one is needed to insert a record.

No abnormal terminations by OPEN. The VSE/VSAM Open routine does not abnormally terminate the user program but returns an explanatory message in all cases where it cannot carry out a request to open a file.

Automatic close. DOS/VSE automatically attempts to close all open VSAM files within the partition at both normal and abnormal termination of a job step. If any control blocks for a file have been destroyed, VSE/VSAM informs the operator that the file cannot be closed.

Summary of VSE/VSAM Advantages Over ISAM

Efficiently organized index.

- Keys are stored in a compressed format
- Better method of updating the index
- Less updating required

Support of variable length and spanned records.

Catalog as the central information point for files and storage volumes.

Ability to dynamically delete records and reuse that space within the file.

- This results in less need to reorganize the file.

Access Method Services

- Backup and recovery tools
- Data interchange tools
- Reorganization tools
- Verify
- Information display facilities

Device and file organization independence.

OS/VS and DOS/VSE Compatibility.

Integrity and security

- Password protection
- Shareoptions

Secondary (alternate) indexes

Shared VSE/VSAM modules

Ability to get secondary space (when available) dynamically.

Simplified job control language.

Ability to dynamically change the length of a record within the file.

How To Use The ISAM Interface Program

To use the ISAM Interface Program, you must convert the ISAM files of your programs to VSAM key-sequenced files. You can use your ISAM load program by way of the ISAM interface to convert the file, or you can use the REPRO command of Access Method Services. You must also change the ISAM job control statements to meet the requirements of VSE/VSAM. In addition you must ensure that your existing ISAM programs comply with the restrictions as indicated under "Restrictions in the Use of the ISAM Interface Program" later in this chapter. (Most existing programs that use ISAM require little or no modification to use the ISAM Interface Program to process VSAM files.

Figure 22 summarizes converting indexed sequential files to key-sequenced files and processing them either with programs that have been converted from ISAM to VSE/VSAM, with programs that still use ISAM, or with new VSE/VSAM programs.

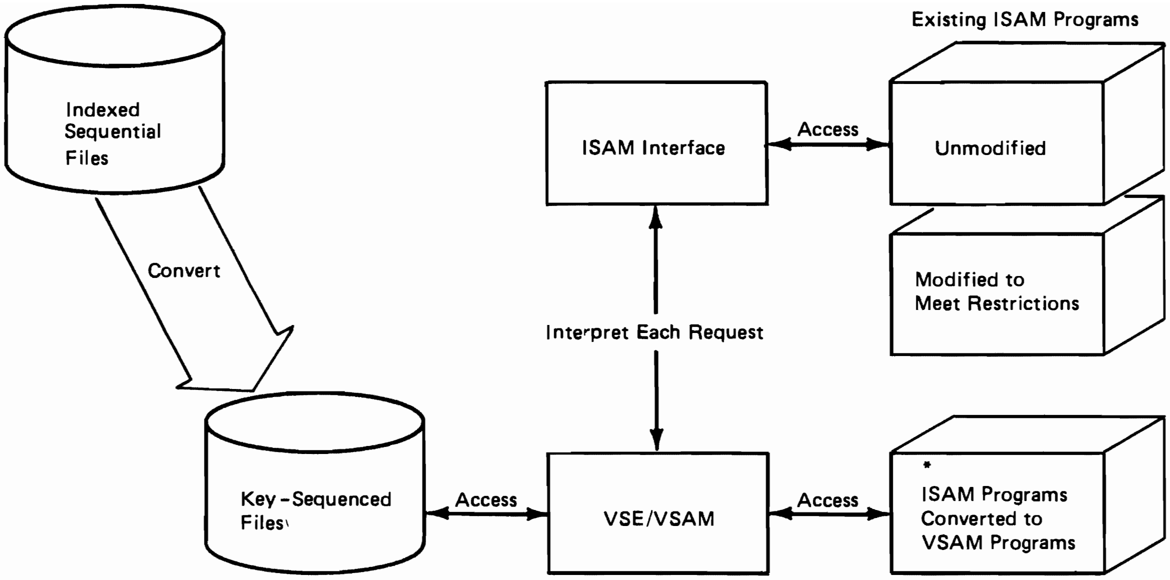
What the ISAM Interface Does

ISAM interface routines operate in conjunction with VSE/VSAM access method routines. The interface routines intercept ISAM requests and convert them to equivalent VSE/VSAM requests. For example, when a processing program (assembler, PL/I, RPGII, or COBOL) that uses ISAM issues an OPEN for a key-sequenced file, VSE/VSAM automatically invokes the ISAM Interface Program. The ISAM interface intercepts each subsequent ISAM request, analyzes it, and then initiates the equivalent VSE/VSAM request.

Restrictions in the Use of the ISAM Interface Program

The ISAM interface enables programs that use ISAM to issue only those requests that VSE/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.
- The program must use standard ISAM interfaces.
- Record ID processing of ISAM cannot be used because VSE/VSAM does not use the record ID functions.
- VSE/VSAM does not return (1) device-dependent information or (2) the virtual storage or DASD address of the record containing the error in the ERREXT parameter list.
- Files defined with SHAREOPTIONS(4) cannot be shared between IIP users in different systems because IIP always opens a file for output.
- The ISAM program cannot open a DTF while another ISAM DTF or VSAM ACB for output processing is already open for the same file unless SHAREOPTIONS (3) or SHAREOPTIONS (4) was specified for the file in the DEFINE command. Refer to "VSE/VSAM Data Protection" for information on file sharing options.



* Do this if you wish to take advantage of all the available functions of VSE/VSAM.

Figure 22. Use of ISAM Programs to Process VSAM Files

VSE/VSAM Performance Options

VSE/VSAM provides you with various options to optimize performance. You specify the options by using the Access Method Services commands. This section briefly describes some of the options available to you for optimizing performance and storage requirements. Details on the application (selection, use) of the various options are provided in *Using VSE/VSAM Commands and Macros.*

Control Interval Size

A file's control interval size can greatly affect the performance of your VSE/VSAM system. For example, if you desired optimum performance for the sequential processing of a key-sequenced file you generally would specify a large data control interval; a smaller data control interval usually is best for direct access processing.

A particular control-interval size can be requested by you but remember that it must fall within the acceptable limits determined by VSE/VSAM, else VSE/VSAM will make adjustments to the size you requested.

I/O Buffer Space

Specifying an optimum I/O-buffer size is very important to VSE/VSAM's performance because VSE/VSAM transfers the contents of a control interval to an I/O buffer in virtual storage; therefore, buffers must have sufficient space to handle the control interval size selected. The amount of space for I/O buffers is one of the most flexible variables you have for influencing performance (along with control-interval size).

Distributed Free Space

The percentage of free space in a control interval and the percentage of free control intervals in a control area affects VSE/VSAM's processing performance for a key-sequenced file or an alternate index. If you specify an optimum amount of free space, you reduce the likelihood of control-interval and control-area splits. Avoiding control-interval and, in turn, control-area splits reduces the likelihood of moving records to a different cylinder, away from other records in key sequence.

The amount of free space to be provided depends on the number and location of records to be inserted, lengthened, or deleted. If a file is for reference only (no records lengthened or added to the file), it does not need any free space. If new records are added frequently to a file you might get the best performance by leaving half of the space free when you create the file.

If enough free space is not provided, VSE/VSAM will require extra processing time to: split control intervals or control areas; process the records sequentially when they are not physically in sequence (caused by the control-interval or the control-area split).

If too much free space is provided: more direct-access storage than necessary is used to contain the file; additional I/O operations are required to sequentially process the same number of records; additional index records are created which increases the time spent in direct processing.

In general, the percentage of growth of a file should be estimated and a proportionate amount of free space should be set aside. You can monitor the control-interval and control-area split activity of the file and change the free space specification, if necessary, for better performance.

Preformatting Control Areas

Two options are available to you when loading records into a file. You can specify that:

- For greater data integrity, VSE/VSAM is to preformat each control area before loading records into it.
- In the interest of performance, VSE/VSAM is not to format the last control area of a file until a CLOSE macro instruction is issued.

For more information on preformatting control areas, see the DEFINE CLUSTER command under "Data Recovery."

Index Options

The index of a key-sequenced file or alternate index can greatly influence VSE/VSAM's performance and storage requirements. Index options available to you are:

- Index and data on separate volumes
- The number of index records in virtual storage
- Sequence set records adjacent to control areas
- Replication of index records
- Placement of the index component in a fixed-head area of a volume

Index and Data on Separate Volumes

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

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

Number of Index Records in Virtual Storage

Direct processing performance improves if a large number of index records can be held in virtual storage where they are readily accessible to VSE/VSAM. This is controlled by means of the buffer size specification.

Sequence Set Records Adjacent to Control Areas

When a key-sequenced file, alternate index, or catalog is defined, you can specify that the sequence set index record for each control area is to be at the beginning of the control area. This reduces disk-arm movement because it minimizes the need to do separate seeks to locate both the sequence set index record and the data record. In most cases one arm movement enables VSE/VSAM to retrieve or store both the sequence set index record and the contents of the control interval in which the data record is stored. When this option is used, sequence set records are *replicated* to reduce rotational delay; that is, as many copies of the sequence set as will fit on the track (min-CA) are written. (See the "Glossary" for the definition of min-CA.)

Figure 23 illustrates replication of a sequence set record that has been placed adjacent to its control area. The advantages of this option are discussed further.

Replication of Index Records

You can specify that each index record be replicated; replication reduces the time lost waiting for the index record to come around to be read (rotational delay).

This option costs direct access storage space. It requires a full track (or the equivalent of a full track for a Fixed Block Architecture device) for each replicated index record. To decide whether to replicate, you have to weigh the relative values of direct access 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 and both sequence set records and index set records replicated.
- Sequence set and index set together and all index records replicated.

Key Ranges

The records of a key-sequenced file and an alternate index can be grouped on volumes according to key ranges. A payroll file, for example, could have employee records beginning with A, B, C, and D on one volume; E, F, G, H, and I on a second volume; etc. You can then process each volume's records separately or you can process as many volumes together as your program(s) requires.

Performance Measurement

VSE/VSAM keeps statistical information about a file in the file's catalog record. Some statistics, such as number of extents in a file, number of records retrieved, added, deleted, and updated, and number of control interval splits, can help you decide when to take action (such as reorganizing a file or altering the type of processing) to improve performance.

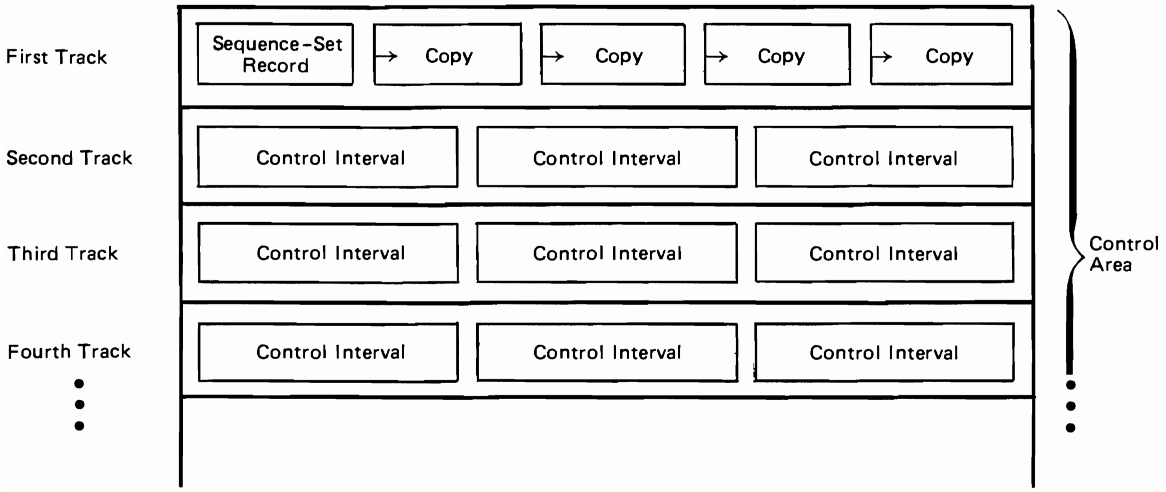


Figure 23. Sequence Set Record Imbedded

VSE/VSAM Data Protection

VSE/VSAM protects data by means of its design (data can be accessed only via the catalog) and its *security* and *integrity* options. Data security means the protection of data from unauthorized use or intentional destruction or alteration. Data integrity means the safety of data from accidental alteration or destruction.

All security options and some integrity options are specified for a file and its components when you define them. Some integrity provisions can be implemented through operator commands rather than through Access Method Services command options. The protection of data also includes commands to recover and restore data.

Options To Achieve Data Security

VSE/VSAM provides options to protect files against unauthorized use and destruction of data. These options include:

- Passwords
- A user-written security verification routine
- The deletion of confidential data by overwriting it with zeros.

Passwords To Authorize Access

VSE/VSAM provides a password option to enable you to protect your data from unauthorized use. You have the facility to prevent a program from processing a file, unless the program or the operator supplies an authorized password to VSE/VSAM.

You specify the password when you define the file with Access Method Services. Password protection is kept in the VSAM catalog. If you do not specify the password option, no password is required to access the file.

You can specify various levels of security with each level requiring a separate password. The various degrees of security (from high to low) are:

- Full access. This is the master password level which allows you to gain access to a file 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 file and to alter password information or any other information in the catalog about a file, index, or catalog.
- Control Access. This password level authorizes you to retrieve and store the contents of a control interval, rather than a single record.
- Update access. This password level authorizes you to retrieve, update, insert, or delete records in a file. Specifying a catalog's update password authorizes you to define files in it.
- Read access. This is the read-only password level which allows you to retrieve data records and catalog entries. You are not allowed to add, change, or delete them, nor to see password information in a catalog entry.

When you open a password-protected file for processing, VSE/VSAM compares the password supplied in the program or by the operator to the password information in the catalog. If an incorrect or an insufficiently high-level password is given, VSE/VSAM will not process the file.

VSE/VSAM provides two additional options that can be used when the operator is required to supply a password. You can request that the operator be given zero to seven chances to supply the correct password and you can also specify a code name in place of the file name. The code name helps keep data more secure by not allowing the operator to know both the name of the file and its password.

User Security Verification Routine

If you specify password protection for a file (or a catalog), you can also supply an additional measure of security to check the authority of a processing program to access the file. 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 an authorization routine. VSE/VSAM transfers control to the authorization routine only if a user gives a correct password. (The user verification routine is bypassed whenever a correct master password is specified.)

Options To Achieve Data Integrity

You must plan in advance how much and what kind of protection you need in the event that your data is destroyed or becomes inaccessible to you. You should consider such questions as: Does it take less time and effort, or expense, to recreate lost data than to maintain backup copies? Should I segregate VSAM and nonVSAM files and make maximum use of recoverability, or should I use another method to make the file current? The following section provides you with additional considerations.

Protecting Shared Data

Files can be shared among partitions or among tasks in a partition. (For VSE/VSAM Release 2, files can also be shared across DOS/VSE systems; this requires the VSE/Advanced Functions Package.) File sharing is controlled by specifications in Access Method Services commands and the type of processing (input or output) for which the file was opened.

In determining the level of sharing to be allowed, you must evaluate the consequences of a loss of *read integrity* to the processing program and a loss of *write integrity* to the file owner. You can specify one of the following file sharing options:

- Sharing option 1: The file can be opened by any number of programs for input processing (retrieve records) or it can be opened by one program for output processing (update or add records). This option ensures full (read and write) integrity.
- Sharing option 2: The file can be opened by more than one program for input processing and, at the same time, it can be opened by one program for output processing. This option ensures write integrity but, since the file might be modified while records are being retrieved from it, each user must ensure his own file's read integrity.
- Sharing option 3: The file can be opened by any number of programs for both input and output processing. VSE/VSAM does nothing to ensure either the integrity of information written in the file or the integrity of the data retrieved from the file. (VSE/VSAM does ensure that the file is not deleted or reset while another program has it open.)
- Sharing option 4: A key-sequenced file can be opened by any number of programs for both input and output processing. (For VSE/VSAM Release 2, the file can be opened by any number of users for input processing. At the same time the file can be opened by one or more users on a *single* system for output processing.) VSE/VSAM ensures write integrity by using the trackhold/blockhold facility of DOS/VSE. Read integrity is ensured by VSE/VSAM only when records are being retrieved for update.

VSE/VSAM Data Backup Considerations

You must consider several factors, other than the physical methods of completing the job, in choosing methods of backup and recovery. They are: the need for backup, operational characteristics, and security and integrity of the backup medium. VSE/VSAM provides you with a variety of options to choose from.

- *Necessity for backup.* If the file can be recreated from the original input or from records or journals kept, perhaps you have no need for backup. Considering the time required for backup procedures and the infrequency of recovery, many files may fit into this category.
- *Operational factors.* You should consider frequency of backup and possible frequency of recovery, time required for backup and recovery, and the ease or difficulty of the backup and recovery technique used.
- *Frequency factors.* You may find some methods of backup or recovery are considerably easier to use than others but may require more time to accomplish. Thus, a method that might be suitable for one file because of its relative infrequency might be unacceptable for another file because of its frequency.
- *Time required factor.* The time required for backup and recovery may be a deciding factor in the choice of method, particularly for real-time systems where recovery must be accomplished quickly.
- *Ease of use.* The alternatives for backup and recovery vary widely in relative ease of use. Complicated methods that are difficult to use may cause errors, which makes recovery much more time consuming than estimated.
- *Physical security and integrity.* Security and integrity of the backup medium are often neglected. Measures used while data is on the system are of no use for a backup copy that is stored elsewhere.

Data Recovery

Every installation regardless of the access method used, needs to guard against the inadvertant loss of data by having recovery and backup procedures available for its files and storage volumes. A recovery and/or backup procedure provides you with a way to restore addressability to data (it makes data accessible) after an error condition.

With VSE/VSAM, one additional concern (besides file and volume recovery) needs to be considered; that is, how to recover from errors that can occur to a catalog. Recall that (1) all VSAM files must be cataloged and (2) all physical and logical descriptions of a file are contained in its catalog entries. It is apparent that any recovery procedure for a catalog must synchronize (match) the file and its catalog entry information.

VSE/VSAM provides you with the ability (using Access Method Services commands and DOS/VSE utilities) to analyze and recover from the following error conditions:

- *File not properly closed.* VSAM files are not properly closed when after they are opened for output or update, a system failure occurs or abnormal termination was entered. (If abnormal termination is entered, an attempt to close all open VSAM files is made automatically.) This condition is reflected in the VSAM catalog. It results in the issuance of a warning message the next time a program tries to open that file.
- *Inaccessible file.* A file may become inaccessible due to damage to the file itself, to related information in the catalog, or to both.
- *Unusable catalog.* A catalog may become unusable because of physical damage to the catalog volume, or a power failure while the catalog was being updated.
- *Unusable volume.* A specific volume may become wholly or partially unusable because of physical damage to the volume or because the catalog that owns the volume was restored to a state that is not synchronized with the volume.

Most recovery situations require one or more of the following:

- A backup copy of the file, volume, and/or catalog
- A method of making the backup copy available for file or catalog recovery.
- A means of synchronizing the file and catalog information (matching catalog information to that of the volume or file).

This section briefly covers some of the many recovery techniques and actions that exist for the recovery of VSAM files. You can divide the types of data recovery into two general categories -- reset and repair.

You primarily use the *reset* type of operation to recover from physical damage to storage volumes or from logical problems that are caused by programming errors. This is probably the most common type of data recovery because of the types of problems encountered and the level of data available for recovery. A file that has been reset will usually require updating before it is usable. When you perform a reset type of operation, VSE/VSAM:

- Provides addressability and access to a *previous* version of the data.
- Requires that a backup copy of the data to be recovered is available (and in most cases a backup copy of the pertinent catalog information.)
- Requires that after you restore the backup copy, you make it current by rerunning all transactions made since you created the backup copy.

You use the *repair* type of recovery primarily if the file is not properly closed or if a catalog failure occurs to a catalog that you defined with the recoverable attribute. When you perform this type of data recovery VSE/VSAM restores addressability and access to the most *current* version of data (instead of a previous version of data).

VSE/VSAM provides you with the following preventive and corrective measures to achieve data backup and recovery for a variety of error situations:

- The *VERIFY command* enables VSE/VSAM to check and correct, if necessary, the end-of-file information that is stored in a VSAM catalog. If the VERIFY function discovers a discrepancy between the file and the catalog end-of-file information, it will update the catalog end-of-file information to correspond with the file. This type of error condition usually occurs when the data file was not closed properly. (VSE/VSAM informs you by means of a warning message that the program attempted to open a file that was not closed properly.)
- The *BACKUP command* enables VSE/VSAM to create a backup tape copy of one or more VSAM files. Two cross-reference listings are printed after each BACKUP command to show how individual objects that were backed up are distributed over the tape volumes of the backup file. Should the VSAM files ever become inaccessible, the RESTORE command can be used to restore them from the backup copy.
- The *EXPORT command* enables VSE/VSAM to create a portable (backup) tape or disk copy of a VSAM file. The backup copy includes relevant catalog entry information about itself, such as, control interval size or record size. Should the VSAM file ever become inaccessible, you would use the *IMPORT command* to introduce the backup copy of the file into the system and use it as a base in the reconstruction (if needed) of your data file.
- The *REPRO command* enables VSE/VSAM to recover from failures that make a nonrecovera-

ble catalog inaccessible by backing up (unloading and reloading) the catalog.

Catalog backup (unload) is when Access Method Services copies a VSAM catalog to a sequential file or to a VSAM key-sequenced or entry-sequenced file. Catalog recovery (reload) is when VSE/VSAM copies the unloaded version into an existing VSAM catalog. The existing catalog can be one that 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. The closer your backup copy comes to matching the active catalog, the more successful any recovery operation will be.

- The *LISTCAT command* lists the contents of a catalog after a recovery operation was performed. You compare this list with a copy of the most recent (before recovery) list to check for any catalog entry mismatches. You can then determine what action to take, depending upon the type of mismatch.
- The *Fast Copy Disk Volume system utility program* enables you to create a backup copy of an entire volume in a very short time. You can then restore that copy on a volume, as needed. The volume to be copied can contain any combination of files.
- The *DEFINE command* allows you to reserve an entire volume or group of volumes for VSAM's exclusive use. This allows recovery on a volume basis to be strictly VSAM or nonVSAM. (Two different approaches for data integrity are needed if VSAM and nonVSAM data spaces are on the same volume.)
- The *DEFINE USERCATALOG command* enables you to create many user catalogs (as many as one per volume) and thereby reduce the number of files per catalog. If a catalog becomes unusable and has, for example, only ten files cataloged in it, access to only these ten files has to be recovered.
- The *DEFINE CLUSTER command* enables you to specify that VSE/VSAM is to (a) preformat each control area as records are loaded into the cluster for greater data integrity (RECOVERY parameter) or (b) is not to preformat them in interest of performance (SPEED parameter). Preformatting clears all previous information from the direct-access storage area and writes an end-of-file indicator in the last control interval as records are loaded into the preformatted area. The end-of-file indicator helps to prevent data that has been added to a file from being lost. If an error occurs that prevents loading from continuing, you can readily identify the last successfully loaded record and resume loading from that point.

  Without preformatting, an end-of-file indicator is written only after the last record is loaded. If an error occurs that prevents further processing from continuing, you might not be able to identify the last successfully-loaded record. This could result in the loss of all of the data that has been loaded up to this point.

Options For a Recoverable Catalog

In addition to the data recovery options listed above, VSE/VSAM provides you with the following options to use for catalogs that were defined with the recoverable attribute. (A recoverable catalog is one that has critical catalog information about its files and volumes duplicated in the catalog recovery area of an applicable volume.) If you cannot access a file via the catalog because of an error condition, you use the catalog recovery area to access it.

- The *RESETCAT command* enables you to clear up any inconsistencies between the entries in a VSAM catalog and its associated catalog recovery areas. This is an efficient way of resetting a recoverable catalog to a level consistent with the catalog recovery area volume, particularly where extensive discrepancies exist. VSAM files are not affected by this operation; no data movement takes place.
- The *EXPORTRA/IMPORTRA commands* are similar to the EXPORT/IMPORT commands. They differ in that a catalog recovery area instead of a catalog is used to obtain the needed catalog information. If the file is not addressable through the catalog, you use the EXPORTRA command to gain access to both the catalog recovery area and the file to create a copy of the file. You can retrieve multiple files and place them on a single portable copy. You use the IMPORTRA command to introduce the recovered information back into the system.
- The *LISTCRA command* lists the contents of the catalog recovery area. It can also be used to compare the contents of the catalog recovery area with those of the catalog. You can use the compare option to detect potential catalog problems or to check the validity of a catalog that was established by means of the REPRO command or the FAST COPY utility program.

Quick Recovery

Some applications, such as on-line teleprocessing systems, require that file recovery be done as quickly as possible. In this type of situation, normal VSE/VSAM recovery procedures may be too time consuming to be of much use. There are, however, some restrictions which can be placed on VSAM files which allow for much quicker recovery. These restrictions are ones which will not allow the backup catalog to become out of synchronization with the files that it controls. See "Quick Recovery" in *VSE/VSAM Programmer's Reference* for additional information on these restrictions.

Problem Determination

VSE/VSAM offers several problem determination aids for you to determine the cause of errors.

Exits to Error-Analysis Routines

VSE/VSAM provides optional exits to routines you supply to handle error situations. Errors can be analyzed and decisions made from the information provided by the error-analysis routines. 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.

VSE/VSAM Messages

The messages issued by VSE/VSAM for the programmer and operator are designed to help them understand both the nature of the problem and the necessary steps to take to correct it. For example, VSE/VSAM issues messages to the operator if an incorrect volume is mounted, or a subsequent volume of a multivolume file must be mounted. VSE/VSAM also issues messages to the operator if an error occurs during catalog processing, a file was not closed during previous processing, or when the operator must supply a password so that a file can be opened. The ISAM Interface Program issues messages to the operator when errors occur while using an ISAM program to access a VSAM file.

VSE/VSAM Return Codes

Most errors detected by a VSE/VSAM processing program are indicated by return codes which are set in a register following execution of a macro. The appropriate user-supplied routine examines the error code to determine what error occurred and how to handle it.

Glossary

CKD (count-key-data). A direct access storage device that contains a fixed-length *count* area, a variable-length *key* area (optional), and a variable-length *data* area in each of its physical data blocks.

Fixed Block Architecture. A direct access storage device that is linear in structure; it contains fixed-length (512 byte) blocks of data.

Max-CA. (See Min-CA.)

Min-CA. The terms "tracks" and "cylinders" as used for present CKD (count-key-data) devices are not necessarily meaningful for a Fixed Block Architecture device because this type of device stores data on "blocks". Fixed Block Architecture uses a linear addressing scheme whereby blocks are not necessarily associated with physical characteristics, such as cylinders or tracks. The following new terms that apply to both CKD and Fixed Block Architecture devices are being used in the documentation to more generally describe the physical characteristics of a device.

Min-CA replaces the former term, track; it represents minimum control area size. Max-CA replaces the former term, cylinder; it represents maximum control area size. Min-CA and max-CA are units of allocation. The size of these units of allocation depends upon the device being used. CKD devices can have more than one min-CA and max-CA value. This is because the min-CA (formerly track) and max-CA (formerly cylinder) sizes for CKD devices are dependent upon the number of physical records contained in them (a function of the control interval size selected). A Fixed Block Architecture device can have only one min-CA and max-CA value.

Index

- ACB macro 26
- accessing data records
  - addressed 21
  - keyed 21
  - control-interval 21
- access method requirements 7, 8
- Access Method Services
  - ALTER command 25
  - BACKUP command 26
  - BLDINDEX command 25
  - CANCEL command 26
  - commands 24-26
  - conditional statements 24-26
  - DEFINE command 25
  - DELETE command 25
  - DO command 26
  - ELSE command 26
  - END command 26
  - EXPORT command 25
  - EXPORTRA command 25
  - functional commands 24-26
  - how used 24
  - IF command 26
  - IMPORT command 25
  - IMPORTRA command 25
  - LISTCAT command 25
  - LISTCRA command 25
  - modal commands 25, 26
  - parameters 26
  - PARM command 26
  - PRINT command 25
  - REPRO command 25
  - RESETCAT command 26
  - RESTORE command 26
  - SET command 26
  - summary of functions 24
  - THEN command 26
  - VERIFY command 26
- access method control block, defining with ACB macro 26
- adding a record to a control interval, illustration 17, 18
- addressed access 21
- addressing data records 10
- addressing control intervals 11
- allocating space 25
  - comparison with ISAM 34, 35
- allocation of track 11
- ALTER command 25
- alternate index
  - advantages of 18, 19
  - data record 19
  - definition 19
  - how it is built 25
  - how it is defined 25
  - illustration 20
  - path to 19
  - records 19
  - structure 19
  - using 18, 19
- alternate index path
  - definition 19
  - illustration 20
- arm movement, minimizing 39
- assembler language 24, 34
- attributes of a file, changing 25
- authorization to process a file 41
- automatic catalog backup 22, 44
- auxiliary storage devices
  - count-key-data 28
  - Fixed Block Architecture 28
  - minimizing rotational delay 39
  - space required for index replication 39
  - VSE/VSAM can be used with 28

- backing up a catalog 42, 43
- backing up a file 42, 43
- backup and recovery procedures 42-45
- BACKUP command 26
- Backup/Restore Feature 32
- base cluster 19
- BLDINDEX command 25
- BLDVRP macro 27
- buffer, I/O
  - effect on performance 38
  - index-set records in virtual storage 39
- building an alternate index 25

- catalog
  - backup and recovery 42, 43
  - job 23
  - master 22-24
  - protection 22
  - recovery area 43, 44
    - listing 44
    - gaining access to 44
  - relationships, illustration 23
  - unusable 43
  - user 22-24
- CKD (see count-key-data)
- creating backup copies 42, 43
- creating files 22
- cause of problems, determining 46
- CLOSE macro
  - disconnecting program from data 26
- clusters 12
- COBOL/VS 24, 34
- coding, VSE/VSAM 24
- commands of Access Method Services (see Access Method Services)
- comparison of VSE/VSAM and ISAM 31
- compatibility with DOS/VS and OS/VS VSAM 28-32
- compression, key 15
- conditional statements, Access Method Services 24, 26
- connecting program to data 26
- control area
  - definition 12
  - preformatting 38, 44
  - relation to control interval 12
  - illustration 12
  - split 17
- control blocks 26, 27
- control interval
  - definition 10
  - effect of size on performance 38
  - relation to control area 12
  - size 10
  - spanning 12
  - split 17, 18
- control-interval access, definition 21
- converting files to VSE/VSAM format
  - indexed-sequential files 34
  - REPRO command of Access Method Services 25
- copying catalogs 25
- copying files 25
- count-key-data storage devices 28
  - physical records in 11

- data component of a file 13
- data component of an alternate index 19
- data format 10
- data integrity
  - options 41, 42
  - passwords 41
- data movement, minimizing 16
- data portability
  - file 24, 25
  - volume 25
- data protection 41-44
  (see also data integrity and data security)
- data record
  - accessing 21
  - illustrations 10
  - locating 21
  - method of addressing 11
    (see also relative byte address)
  - method of storing 10
  - processing 21
  - spanned 12
- data recovery 42-45
- data repair 43
- data reset 43
- data security
  - authorization routine 41
  - definition 41
  - passwords 41
- data space
  - allocation 11, 22
  - definition 23
  - extents 13
  - illustration 14
- DEFINE command 24
- DELETE command 25
- determining causes of problems 46
- devices, auxiliary storage
  - minimizing rotational delay 39
  - space required for index replication 39
  - VSE/VSAM can be used with 28
- diagnostic aids 46
- differences
  - among file organizations 20
  - between VSAM release 34 and VSE/VSAM 28
  - between VSE/VSAM and ISAM in common areas 34
- direct access
  - addressed 21
  - control-interval 21
  - keyed 21
- direct processing 21
- direct access storage devices
  - minimizing rotational delay 39
  - space required for index replication 39
  - VSE/VSAM can be used with 28
- disconnecting a program from data 26
  - temporarily 26
- disk storage structure for VSE/VSAM 13
- displaying fields of the VSAM catalog 27
- distributed free space 11, 14, 16
  - distribution 16
  - for deleting records 16, 17
  - for inserting records 16, 17
  - illustrations 16-18
  - reclamation 16
- DLVRP macro 27
- DOS/VSE and OS/VS
  - compatibility 28-32
  - file portability 25
  - volume portability 25
- DOS/VS Assembler 24, 34

- ENDREQ macro 27
- entry in VSAM catalog 22
- entry-sequenced file 9, 13
  - comparison with other types 19, 20
  - definition 9, 13
  - keeping track of relative byte addresses 13
- ERASE macro 27
- erasing a file, DELETE command 25
- error analysis 46
- error messages 46
- examining control blocks and lists 27
- EXLST macro 27
- EXPORT command 25
- EXPORTRA command 25

- failures, determining cause of 46
- file
  - copying 25
  - creating 22, 25
  - data component 13
  - defining 24, 25
  - deleting 25
  - differences in organization 20
  - direct organization 9, 14
  - entry in catalog 22, 25
  - entry-sequenced 9, 13
  - illustrations 9, 10
  - indexed organization 10, 14
  - key-sequenced 10, 14-18
  - management 22
  - organization 9, 10
  - portability 25
  - preformatting 38, 44
  - protection 41-44
  - recovery 42-45
  - relative record 9, 14
  - reorganizing 24, 25
  - sequential organization 9, 13
  - sharing 42
  - space for 13
  - transporting 25
- Fixed Block Architecture support 28
- fixed-length records 9
- format of stored data 11
- free space (see distributed free space)
- functions of VSE/VSAM 24
- functional commands 24

- GENCB macro 26, 27
- generating control blocks and lists 26, 27
- GET macro 27

- high-level languages 24, 34
- how existing programs that use ISAM
  can be used with VSE/VSAM 36
- how to communicate with VSE/VSAM 24
- how to use the ISAM interface program 36
- how VSE/VSAM assigns addresses to data records 10
- how VSE/VSAM locates data records 21

- IIP (ISAM Interface Program)
  (see ISAM interface)
- I/O buffer space 38
- I/O errors 46
- IMPORT command 25
- IMPORTRA command 25
- index
  - alternate 19
  - comparison with ISAM index 35
  - illustration for a key-sequenced file 16
  - options 38, 39
  - performance options 38-40
  - prime 14
  - purpose 14
  - structure 16
- index component of an alternate index 19

index component of a file 13
index entry
  description 15
  key compression 15
index record
  entries 15
  key compression 15
  levels 15
  replication 39
  sequence-set record adjacent to control area 39
index set
  definition 15
  description 15
  illustration 16
  physical placement in relation to sequence set 15
indexed sequential access method (ISAM)
(see also indexed sequential file and
ISAM interface)
  comparison with VSE/VSAM 34-36
indexed sequential file 34-37
  converting to VSE/VSAM format 36
integrity of data
  definition 41
  options 41, 42
  passwords 41
  shared data 42
interpreting ISAM requests 36, 37
ISAM file (see indexed sequential file)
ISAM interface
  converting files 36
  illustration 37
  operation 36
  purpose 34
  restrictions 36

job control statement 24
job catalog 23

key
  allocating space on volumes by range 39
  alternate 19
  compression 15
  ranges 39
  use in index 14, 15
key field, description 14, 15
key-sequenced file 10, 14-18
(see also distributed free space, file, and index)
  comparison with others types of files 10, 20
  illustration 10
keyed access 21

languages, programming 24, 34
lengthening a record
  changing relative byte addresses 18
  control-area split 17, 18
  control-interval split 17
  without split 17
levels of index
  illustration 16
  index set 15
  sequence set 15
LISTCAT command 25
LISTCRA command 25
listing catalog records 25
listing files 25
loading records into a file
  comparison with ISAM 35
  preformatting options 38, 43
  REPRO command 25
  using a processing program 26
locating data records 14, 21
logical record (see data record)

machines VSE/VSAM can be used with 28
macros
(see also Access Method Services for commands)
  ACB 26
  BLDVRP 27
  CLOSE 26
  DLVRP 27
  ENDREQ 27
  ERASE 27
  EXLST 26
  GENCB 26, 27
  GET 27
  MODCB 26
  OPEN 26
  POINT 27
  PUT 27
  RPL 26
  SHOWCAT 27
  SHOWCB 26
  summary of VSAM macros 26, 27
  TCLOSE 26
  TESTCB 26
  WRTBFR 27
making a file portable 25
master and user catalogs 22
master password 41
messages 46
minimizing data movement 16
min-CA allocation 11
min-CA space on storage volumes 11
modal commands 26
MODCB macro 26
modifying control blocks and lists 26
moving files from one system to
another system 25

nonVSAM files 13, 24, 25

OPEN macro 26
options to achieve data integrity 41
options to achieve data security 41
options for a recoverable catalog 44
organization of a file 9, 10
OS/VS and DOS/VSE
  compatibility 28-32
  file portability 25
  volume portability 25
overflow area
(see also distributed free space)
  comparison with ISAM 35

passwords 41
path (see alternate index path)
performance
  improved by control interval size 38
  improved by distributed free space 38
  improved by key ranges 39
  index options 38, 39
    illustrations 40
  measurement 39
  options with VSE/VSAM 38-40
physical record
  illustration 12
  use of 11, 12
  size 11, 12
PL/I Optimizer language 24, 34
POINT macro 27
portability
  file 25
  volume 25
preformatting control areas 37
preformatting end of file 38, 44
preparing for VSE/VSAM 28

- prime index 14, 15
- PRINT command 25
- problem analysis 46
- problem determination 46
- processing data with VSAM macros 26
- processing of VSAM files 21
- processing programs 24
- processing records 21
- programs, processing 24
- programming languages 24, 34
- programming requirements 34
- protecting data 41-45 (see also data security and data integrity)
- PUT macro 27

- quick recovery 45

- random access (see direct access)
- RBA (see relative byte address)
- reclaiming space
  - illustration 16
  - key-sequenced file 16
- recoverable catalogs 44
- recovering data 42-45
- recovering from catalog failure 44
- recovery tools 42-45
- reestablishing data integrity 41-44
- relationship, catalog and volume 22
- relationship of VSE/VSAM to virtual and auxiliary storage 8
- relative byte address (RBA) 10, 11
  - illustration 10
- relative record file 10, 14
  - comparison with other types 10, 20
  - illustration 10
- relative record number
  - definition 14
  - use as a key 14
- reorganizing files
  - automatically through control-area split 17, 18
  - by copying 25
  - comparison with ISAM 34-36
- replication of index records 39
- REPRO command 25
- requesting access to a file 27
- requirements
  - coding 24
  - for an access method 7, 8
  - programming 33
  - storage 32
- resetting a recovering catalog 44
- RESTORE command 26
- restrictions in the use of the ISAM Interface Program 36
- retrieving a record
  - addressed 21
  - control-interval 21
  - keyed 21
- role of a prime index in a key-sequenced file 14
- rotational delay, minimizing 39
- RPG-II/VS 24, 34
- RPL macro 26

- security of data
  - authorization routine 41
  - definition 41
  - passwords 41
- sequence set
  - definition 15
  - description 15, 16
  - determining key sequence 18
  - location 15, 16
  - physical placement in relation to index set 15
  - relation to control areas 15, 16
- sequence-set records adjacent to the file 38, 39
- sequential access 21
- sequential data organization 9, 13
- sequential processing 21
- sharing data 42
- shortening a record 16
- SHOWCAT macro 27
- SHOWCB macro 26
- size of a control interval 10
- size of physical records 11, 12
- space management 22
- spanned records 12
  - illustration 12
- special uses of user catalogs 22
- speed (see performance)
- split
  - control-area 17
  - control-interval 17, 18
- storage devices VSE/VSAM can be used with 28
- storage requirements 33
- storing a record 9, 10
- structure of VSAM files 9, 10
- summary of VSE/VSAM advantages over ISAM 36
- systems sharing data 42

- TESTCB macro 26
- testing control blocks and lists 26
- track space on storage volumes 11, 13
- track allocation 11, 13
- translating ISAM requests 36, 37
- transporting data between systems
  - file portability 22
  - volume portability 22

- unusable catalogs
  - with catalog recovery areas 44
  - without catalog recovery areas 42-44
- use of free space for processing a key-sequenced file 17, 18
- use of VSAM by existing programs 34
- user catalog
  - connecting to master catalog 22, 25
  - disconnecting from master catalog 25
  - protecting 22, 41-44
- user security verification routine 41
- using free space to delete or add records 16
- utilizing space on a storage volume 11

- variable-length records 9-10
- verification routine, security 40
- Virtual Storage Access Method
  - catalog 22, 24
  - comparison with indexed sequential access method 34-36
  - files 22
  - illustration of file organization 13
  - macros 26, 27
  - performance options 38-40
  - requirements for processing data 8, 9
- volume inaccessible
  - with catalog recovery areas 44
  - without catalog recovery areas 42-44
- volume ownership 22
- volume portability 22, 25
- VSAM (see Virtual Storage Access Method)
- VSAM catalog 22, 24
- VSAM return codes 46
- VSE/VSAM's approach to space and file management 22
- VSE/VSAM data backup considerations 42
- VSE/VSAM's Data Organization 9

VSE/VSAM's disk storage structure 13
VSE/VSAM enhancements 28-32
VSE/VSAM functions not available with ISAM 35, 36
VSE/VSAM messages 46
VSE/VSAM performance options 38
VSE/VSAM Space Management for SAM Feature 32
VSE/VSAM's structure 10

ways you can organize data records
for VSE/VSAM 9, 10
what is VSE/VSAM 9
what's new for VSE/VSAM Release 2 30
what's new with VSE/VSAM 28
what the ISAM interface does 36
WRTBFR macro 27

GC24-5143-1

VSE/VSAM General Information (File No. S370/4300-30) Printed in U.S.A. GC24-5143-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601

VSE/VSAM
General Information
GC24-5143-1

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

	Yes	*No*
• Does the publication meet your needs?	□	□
• Did you find the material:		
Easy to read and understand?	□	□
Organized for convenient use?	□	□
Complete?	□	□
Well illustrated?	□	□
Written for your technical level?	□	□

- What is your occupation? ____________________

- How do you use this publication:

As an introduction to the subject?	□	As an instructor in class?	□
For advanced knowledge of the subject?	□	As a student in class?	□
To learn about operating procedures?	□	As a reference manual?	□

Your comments:

If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GC24-5143-1

Reader's Comment Form

Cut or Fold Along Line

Fold and Tape Please Do Not Staple Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department G60
P. O. Box 6
Endicott, New York 13760

VSE/VSAM General Information (File No. S370/4300-30) Printed in U.S.A. GC24-5143-1

Fold Fold

If you would like a reply, *please print:*

Your Name ______________________________

Company Name ____________________ *Department* __________

Street Address ______________________________

City ______________________________

State ____________________ *Zip Code* __________

IBM Branch Office serving you ______________________________

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601

GC24-5143-1

VSE/VSAM General Information (File No. S370/4300-30) Printed in U.S.A. GC24-5143-1

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N. Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N. Y., U. S. A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N. Y., U. S. A. 10601