SC24-5145-1
File No. S370-30

Program Product

VSE/VSAM
Programmer's Reference

Program Number 5746-AM2

Release 2

IBM

Second Edition (December 1979)

This edition, SC24-5145-1, is a major revision of SC24-5145-0. It applies to Release 2 of 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; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 and 4300 Processors Bibliography,* GC20-0001, for the editions that are applicable and current.

Summary of Amendments

For a list of changes, see page iii.

A change to the text or to an illustration is indicated by a vertical line to the left of the change.

It is possible that this material may contain references 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.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or the 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 Programmer's Reference

Summary of Amendments for SC24-5145-1 VSE/VSAM Release 2

SC24-5145-1 contains information about the following line items:

- Dedicated VSAM volume.
- Share option 4 performance improvement.
- DASD sharing.
- Additional space classes.
- Dynamic files.
- Default volumes.
- File disposition parameters.
- Default models: Chapter 6 is new; it explains the different kinds of models and how to use them, along with listing which parameters can be modeled under different circumstances.
- Job control simplification: Chapter 4 explains when you can omit DLBL, EXTENT, and ASSGN statements from your job control for the catalog and file when running application programs and Access Method Services commands.

  Where possible, DLBL, EXTENT, and ASSGN statements have been removed from coding examples. Although you may continue to specify full DLBL, EXTENT, and ASSGN statements, in most cases it is not necessary to do so.
- VSE/VSAM Space Management for SAM Feature: This manual contains storage estimates, job control, and file disposition information for managed-SAM files. For further information, refer to *Using the VSE/VSAM Space Management for SAM Feature,* SC24-5192.

Preface

This manual is for people who provide technical support for VSE/VSAM users.

This includes the system console operator and system programming and planning personnel who are responsible for system design and maintenance.

The chapters are in modular format to permit you to insert them into the associated VSE publications (when applicable) if you wish. The major topics contained in this manual are:

- System generation
- Operating procedures
- Storage estimates
- Job control
- Modeling
- Catalogs
- VSAM labels
- ISAM Interface Program
- Data interchange
- Performance optimization
- VSE/VSAM data protection

The operating procedures, system generation, storage estimates, job control, and VSAM labels chapters contain information about parameters, procedures, field descriptions, etc. that are unique to VSAM. They supplement the information in *VSE/Advanced Functions Operating Procedures,* SC33-6097, *VSE/Advanced Functions System Generation,* SC33-6096, *VSE/Advanced Functions System Control Statements,* SC33-6095, and *VSE/Advanced Functions DASD Labels,* SC24-5213, respectively. A general knowledge of these publications, along with *Using VSE/VSAM Commands and Macros,* SC24-5144, is a prerequisite to this volume; continue to refer to these manuals for general information about VSE.

The ISAM Interface Program (IIP) chapter presents information to help you determine whether your existing ISAM programs can use the IIP to process files that have been converted from ISAM format to VSAM format.

The performance guidelines chapter is in response to requests we've received for such information. Although great care has been taken to ensure accuracy, this chapter contains *guidelines,* not rules, for enhancing your system's efficiency. This kind of information is difficult to present because of the great number of variables that exist; not everything will be true for all installations under all conditions. We do hope you find this chapter generally useful, but you should be aware that the material it contains may require altering, depending on the needs of your installation.

The VSAM data protection information and some of the performance optimization material formerly appeared in *DOS/VS Access Method Services User's Guide.*

Contents

Chapter 1: System Generation

Supervisor Generation Macros

The SUPVR, FOPT, and IOTAB supervisor generation macros have special considerations for VSE/VSAM users. This section documents only those parameters that are unique to VSE/VSAM. All other information appears in *VSE/Advanced Functions System Generation*.

SUPVR

```
NPARTS=n
```

Specify a number from 2-12 if files with the TRKHOLD attribute are to be processed (FOPT TRKHOLD=n). The default is 3.

FOPT

```
DASDSHR= {NO|YES}
```

If you are using the VSE/Advanced Functions Release 2 DASD Sharing facility, specify DASDSHR=YES at supervisor generation.

```
TRKHOLD= {NO|n}
```

Specify a number from 1-255 representing the number of tracks that can be held at one time by all tasks in the system.

You must use this parameter for VSE/VSAM Space Management for SAM Feature files using managed-SAM access when DTF HOLD=YES is specified.

IOTAB

```
BUFSIZE=n
```

VSE/Advanced Functions System Generation gives information about how to calculate the number of buffers to specify for channel program translation. After you have determined a value, add 40 to it for VSAM blocks. VSAM uses these copy blocks only during actual I/O and frees them immediately after use.

```
NRES = (NPARTS*4) + 21|n
```

The NRES value indicates the number of lock table entries. It must be a value between 3 and 512; the default is (NPARTS*4) + 21. To compute the optimal upper limit for n, use the formula:

```
n = (2*UCAT + 1)*P + (5*C) + (3*OS4) + (2*OS3) + S + P + 5
```

UCAT is the number of user catalogs open concurrently.

P is the number of partitions.

C is the number of catalogs open concurrently.

OS4 is the number of share option 4 VSAM components (for example, key component or data component) concurrently open for *output*.

OS3 is the number of share option 3 VSAM components concurrently open for *output*.

S is the number of VSAM components that are open but not accounted for by OS3 and OS4.

All these values should reflect the situation that exists when NRES is at its maximum value.

The value for NRES (calculated in the above manner) will cause sufficient space to be reserved for the variable resources to be used. Depending on the application, however, the number of resources actually required most of the time might be much lower.

Note: If the value substituted for n is too small and the pool of named resources gets exhausted, the VSAM partition is canceled with an error message displayed.

VSE/VSAM Installation

For installation procedures for program products, refer to *VSE/Advanced Functions Maintain System History Program (MSHP) User's Guide.*

Chapter 2: Operating Procedures

This chapter describes only operating procedures that are unique to VSE/VSAM. Use this information in conjunction with *VSE/Advanced Functions Operating Procedures.*You may even wish to insert these pages into that manual. The commands are documented here in the order in which you should enter them.

IPL

Assigning SYSCAT

During IPL, you can enter the define catalog command if there is no SYSCAT assignment or to change the SYSCAT assignment. Specify the following:

```
DEF SYSCAT=cuu
```

cuu indicates the hexadecimal channel and unit number of the disk device containing the VSAM master catalog. For a 3340, the device must be ready before you IPL.

The new SYSCAT assignment is valid until the next IPL.

Note: If you issue the DEF SYSCAT command, it must follow the (optional) SET command and precede the DPD command.

Defining the Lock File

If you are using the VSE/Advanced Functions Release 2 DASD Sharing facility, you must define the system lock file by specifying the DLF command at IPL. Refer to *VSE/Advanced Functions Operating Procedures* for the format of the DLF command.

Specifying the Size of the SVA

The system builds the system directory list at IPL and allocates SVA space for VSAM and system modules and buffers. No user action is required.

Specifying the Size of the Virtual Address Area (System/370 - Mode Only)

```
VSIZE=nK
```

Specify VSIZE in response to message 0I03A to define the size of the virtual address space (including SVA).

n must be a multiple of 2; if you specify an odd number, VSE adds 1 to it. The maximum value you can specify is 16,384K bytes (16,777,216 bytes) minus the size of your processor storage. You must code the character K.

Determine n using the following formula:

```
n = A + B + U
```

A is the sum of the size of the partitions. Choose 512K or 100K * NPARTS, whichever is larger. Up to 497K bytes are required in the partition in which Access Method Services runs, depending on the function to be performed. (Refer to "Storage Requirements: Access Method Services Virtual Storage" in Chapter 3.)

B is the SVA requirement for the system. It is equal to a basic requirement of 180 + (6 * NPARTS). Depending on the supervisor options in your system, additional storage may be required. *VSE/Advanced Functions System Generation* lists the options and their storage requirements.

U is any special SVA requirements that exist for licensed programs. VSAM requires approximately 400K bytes of SVA.

Executing a Job Step

When you are about to execute a job step, you must enter the EXEC command, specifying the SIZE operand. (SIZE can be specified in conjunction with other operands, including the REAL operand, mentioned below. Refer to *VSE/Advanced Functions Operating Procedures* for information about them.)

```
EXEC [progname],SIZE={nK|AUTO|(AUTO,nK)}
```

nK indicates the size of the root phase. Do not specify a number larger than the size of the partition the program will run in. If you specify an odd number, the system will use n+1.

AUTO indicates that the size of the program, as calculated by the system from information in the core image directory, will be inserted into the SIZE value. For ease of use, specifying SIZE=AUTO is recommended for Access Method Services.

(AUTO,nK) indicates that nK will be added to the size of the program, and this value will be inserted into the SIZE value (rounded upward to a multiple of 2K).

You must specify the SIZE operand for all VSAM programs, any ISAM program that uses the ISAM Interface Program, and SAM files using the VSE/VSAM Space Management for SAM Feature.

For System/370 users, specifying SIZE (without the REAL parameter) means that the partition to be used for the job step is divided into two parts. The lower part (with a size derived from the AUTO and nK parameters) contains the program to be executed. The upper part serves as a storage pool for the partition.

The partition in which VSAM processing takes place must allow for a GETVIS area to accommodate VSAM buffers and control blocks, along with the user program. For VSAM, the size of the partition GETVIS area depends on the number of VSAM files being accessed, as well as their control interval sizes. The partition GETVIS area must be at least 40K for basic buffers and control blocks for each catalog that is open, plus 12K for each KSDS and 10K for each ESDS or RRDS (assuming a CI size of 2K or less). Additional space for modules, buffers, and control blocks is required if you are using any non-SVA-eligible VSAM phases (for example, ISAM Interface Program) or Access Method Services. Refer to Chapter 3 in this manual for further information about VSE/VSAM storage requirements.

ALLOC (System/370 mode only)

The ALLOC command allocates storage to the virtual foreground partitions (or modifies the amount of storage allocated to the virtual foreground partitions). All of the virtual address area not allocated to virtual foreground partitions is automatically part of the virtual background partition.

With VSAM, only the address space in the virtual address area not allocated to the shared virtual area (SVA) can be allocated to foreground partitions with this command.

```
ALLOC Fn=mK[,Fn=mK]...
```

n indicates the partition to which storage is to be allocated; m specifies the amount of storage.

The partition in which VSAM files are to be processed must allow for a GETVIS area to accommodate VSAM buffers and control blocks, along with the user program. The size of the partition GETVIS area depends on the number of VSAM files being accessed, as well as their control interval sizes. The partition GETVIS area must be at least 40K for basic buffers and control blocks for each catalog that is open, plus 12K for each KSDS and 10K for each ESDS or RRDS (assuming a CI size of 2K or less). Additional space for modules, buffers, and control blocks is required if you are using any non-SVA-eligible VSAM phases (for example, ISAM Interface Program) or Access Method Services. Refer to "Reserving Storage for VSE/VSAM" in Chapter 3 for further information.

Volume Mounting

Two approaches allow you to mount one or more volumes required to access VSAM files. If full job control describes the file (DLBL, EXTENT, and ASSGN statements), the required volume must be mounted on the device specified in the job control. VSAM will issue a message to inform you if the requested volume (except for a catalog volume) is not mounted on the requested device and allow you to correct the situation.

If you take advantage of the job control simplifications supported in VSE/VSAM Release 2 by not supplying a symbolic unit on an EXTENT statement, VSAM has greater flexibility in providing the required volume. In this case VSAM is free to use any physical unit — logical unit combination on which the required volume (as indicated by the VSAM catalog) is mounted or can be mounted. If the required volume is already mounted on some device, VSAM attempts to automatically assign that device and avoid the need for operator intervention. For the automatic assignment to be successful, the device must be up (DVCUP command, documented in the *Operating Procedures* manual) and not reserved (VOLUME cuu, RESERVE|FREE command, also documented in *Operating Procedures*). Therefore you should ensure that devices are up before mounting volumes, and refrain from reserving devices unneccessarily.

If the required volume is not mounted, VSAM prompts you to mount it. If possible, VSAM recommends a device and reserves it while the mount is pending. If you choose to use a device other than the recommended device (or if VSAM did not recommend one), you must ensure that the device you use is up and operational and that mounting the required volume does not interfere with other users in the system. Use the VOLUME cuu, RESERVE command to hold a device while a mount is pending. When the volume is mounted, the device becomes ready and the reserved status is reset to free. Your reply to the mount message allows VSAM to verify the volume mount and continue processing the file.

Note: VSAM never recommends a shared DASD device. If you want to use a shared device, you must override any device VSAM recommends.

Chapter 3: Reserving Storage for VSE/VSAM

For VSE/VSAM, there are two general areas for storage considerations. First, Access Method Services must be utilized for file definitions, catalog manipulation, and other VSAM file utility functions. Access Method Services modules cannot be loaded into the SVA; therefore their partition requirement depends on the functions required for the current job. Provide a partition GETVIS area by specifying SIZE=AUTO on the EXEC statement for Access Method Services. Further information about the EXEC statement appears in the Job Control chapter in this manual.

Secondly, most VSAM routines are automatically loaded into the SVA during IPL. Routines that are not reentrant and relocatable (for example, the ISAM Interface Program) are not eligible for SVA. Therefore they have a partition GETVIS requirement.

The partition in which VSAM files are to be processed must allow for a GETVIS area to accommodate VSAM buffers and control blocks; the user program resides in the same partition, below the GETVIS area. The size of the partition GETVIS area depends on the number of VSAM files being accessed, as well as their control interval sizes. When running routines that use the partition GETVIS area, the partition GETVIS area must be at least 40K for basic requirements for each catalog that is open, plus 12K for each KSDS and 10K for each ESDS or RRDS (assuming a CI size of 2K or less).

The sections that follow describe more fully the amount of storage required for VSE/VSAM processing.

General Considerations

This section discusses the four elements necessary to run VSAM: VSAM itself, VSAM/VTAM common macros, VSAM utilities (Access Method Services), and the ISAM Interface Program. VSAM, in combination with the VSAM/VTAM common macros, may run by itself or with either or both of the other two programs.

VSAM Data Management Modules (5745-SC-VSM)

Core Image Library

Transients	12
Blocks	13
Phases	33
Blocks	368
Load List Blocks	1

Transients:

$$BACLOS
$$BCLCRA
$$BCVSAM
$$BCVS02
$$BCVS03
$$BCVS04
$$BODADE
$$BODADS
$$BOSMIN
$$BOVSAM
$$BOVS01
$$BTCLOS

Phases loaded into SVA:

IKQFTIND
IKQVASMT
IKQVBRP
IKQVCAT
IKQVCLC
IKQVCLIF
IKQVCLOC
IKQVCLOS
IKQVCLOV
IKQVCLRD
IKQVDNT
IKQVDRP
IKQVDTPE
IKQVEDX
IKQVEOV
IKQVGEN
IKQVJIBS
IKQVLAB
IKQVMSG
IKQVNEX
IKQVOPEN
IKQVPBF
IKQVRBA
IKQVRM
IKQVSCAT
IKQVSHR
IKQVSTM
IKQVTMS

Non-SVA-eligible phases:

Phases	Bytes
IKQVDCN	156
IKQVDU	19,664
IKQVDUMP	9,216
IKQVEDA	5,808
IKQVRT	76

SVA Load List:

$SVAVSAM

Relocatable Library

Prefix	Modules	Blocks
$S	1	5
$$B	11	42
IGG0	70	898
IKQ	120	1,149
Totals	202	2,094

Modules:

$SVAVSAM
$$BACLOS
$$BCLCRA
$$BCVSAM
$$BCVS02
$$BCVS03
$$BCVS04
$$BODADE
$$BODADS
$$BOVSAM
$$BOVS01
$$BTCLOS
IGG0CLAB
IGG0CLAC
IGG0CLAD
IGG0CLAE
IGG0CLAF
IGG0CLAG
IGG0CLAH
IGG0CLAJ
IGG0CLAK
IGG0CLAL
IGG0CLAN
IGG0CLAP
IGG0CLAQ
IGG0CLAR
IGG0CLAS
IGG0CLAT
IGG0CLAU
IGG0CLAV
IGG0CLAW
IGG0CLAX
IGG0CLAY
IGG0CLAZ
IGG0CLA6
IGG0CLA7
IGG0CLA8
IGG0CLBA
IGG0CLBB
IGG0CLBC
IGG0CLBD
IGG0CLBE
IGG0CLBF
IGG0CLBG
IGG0CLBH
IGG0CLBL
IGG0CLBM
IGG0CLBN
IGG0CLBQ
IGG0CLBR
IGG0CLBS
IGG0CLBT
IGG0CLBU
IGG0CLBW
IGG0CLBX
IGG0CLBY
IGG0CLB8
IGG0CLCA
IGG0CLCB
IGG0CLCD
IGG0CLCG
IGG0CLCL
IGG0CLCO
IGG0CLCP
IGG0CLCR
IGG0CLCS
IGG0CLCX
IGG0CLCY
IGG0CLC9
IGG0CLEG
IGG0CLES
IGG0CLET
IGG0CLEX
IGG0CLEZ
IGG0CLFA
IGG0CLFB
IGG0CLFC
IGG0CLFD
IGG0CLFE
IGG0CLFF
IGG0CLFH
IGG0CLFQ
IKQAIX
IKQALL00
IKQASNMT
IKQBFA
IKQBFB
IKQBFC
IKQBFD
IKQBLD
IKQBRP
IKQCAS
IKQCIL
IKQCIR
IKQCIS
IKQCIU
IKQCLCAT
IKQCLEAN
IKQCLIF
IKQCLNLK
IKQCLO
IKQCLOCL
IKQCLOVY
IKQCLRDD
IKQDCN
IKQDDR
IKQDNT
IKQDRP
IKQDUMP
IKQDUMPC
IKQDUMPS
IKQEDX
IKQEOV
IKQERH
IKQERX
IKQFTIND
IKQFT1
IKQFT2
IKQFT3
IKQGCI
IKQGEN
IKQGNX
IKQGPT
IKQINT
IKQIOA
IKQIOB
IKQIOC
IKQIOD
IKQIXE
IKQIXF
IKQIXS
IKQJIBSM
IKQJRN
IKQKRD
IKQLAB
IKQLCD
IKQLCN
IKQLCP
IKQLNA
IKQMDY
IKQMTMSG
IKQNCA
IKQNEX
IKQOCMSG
IKQOCSHR
IKQOPN
IKQOPNAB
IKQOPNAI
IKQOPNCT
IKQOPNDO
IKQOPNDS
IKQOPNHC
IKQOPNNC
IKQOPNOV
IKQOPNPV
IKQOPNRD
IKQOPNRP
IKQOPNUC
IKQOPNUS
IKQOPNVC
IKQPBF
IKQPFO
IKQPOP00
IKQRBA
IKQRCL
IKQRDS00
IKQREN00
IKQRQA
IKQRQB
IKQRQC
IKQRRP
IKQRTV
IKQSCAT
IKQSCN
IKQSCR00
IKQSFT
IKQSGP
IKQSIN
IKQSLD
IKQSLN
IKQSLP
IKQSMDMB
IKQSMLNK
IKQSMTIN
IKQSPM
IKQSRG
IKQSRT
IKQSRU
IKQSTM
IKQSUP
IKQTMSD
IKQTMSF
IKQUPD
IKQUPG
IKQVDTPE
IKQVEDA
IKQVFY
IKQVRT
IKQVSM
IKQVSMLK
IKQVTC00
IKQWDS00

Link-Edit Statements:

```
// OPTION CATAL
   INCLUDE IKQVSMLK
   INCLUDE IKQSMLNK
// EXEC LNKEDT
```

Ignore notice "CONTROL SECTIONS OF ZERO LENGTH IN INPUT" given by the linkage editor at the end of the partition storage map.

Source Statement Library

Number of Macros	Number of Blocks
20	1,066

Macros (Edited):

BLDVRP
DLVRP
ENDREQ
ERASE
IKQACBG
IKQACB1
IKQCB1
IKQCB2
IKQEXLG
IKQEXL1
IKQGCB
IKQMCB
IKQRPLG
IKQRPL1
IKQSCB
IKQTCB
POINT
SHOWCAT
TCLOSE
WRTBFR

VSE/VSAM Space Management for SAM Feature

Core Image Library

Transients	2
Blocks	2
Phases	6
Blocks	38

Transients:

$$BOSMMW
$$BOSMXT

Phases:

IDCSM01
IKQSMACL
IKQSMMON
IKQSMOEI
IKQSMSSR
IKQVIMR

Relocatable Library

Number of Modules	Number of Blocks
42	295

Modules:

IDCSM01
IDCSM02
IDCSM03
IKQOCIMR
IKQSMACL
IKQSMGDC
IKQSMLNK
IKQSMMBF
IKQSMMLM
IDQSMMON
IKQSMMRT
IKQSMMS0
IKQSMMS1
IKQSMMXI
IKQSMOEI
IKQSMSBF
IKQSMSCD
IKQSMSCW
IKQSMSI1
IKQSMSMC
IKQSMSMO
IKQSMSO1
IKQSMSSF
IKQSMSSR
IKQSMSW1
IKQSMSXI
IKQSMTMW
IKQSMTXT
IKQSMVAL
IKQSMVBD
IKQSMVBJ
IKQSMVCB
IKQSMVFA
IKQSMVGA
IKQSMVHU
IKQSMVMC
IKQSMVMO
IKQSMVOA
IKQSMVRC
IKQSMVRJ
IKQSMVRX
IKQSMVVM

ISAM Interface Program

Core Image Library

Transient	1
Blocks	1
Phases	4
Blocks	8

Transient:

$$BOCISC

Phases	Bytes
IIPCLOSE	575
IIPOPEN	1300
IIPPROC	3750
IIPAMDTF	800

Relocatable Library

Number of Modules	Number of Blocks
7	46

Modules:

IIPAMT00
IIPBMR00
IIPCLS00
IIPIIP00
IIPOPN00
IIPPRCMR
IIPPRCPR

Link-Edit Statements:

```
// OPTION CATAL
   INCLUDE IIPIIP00
// EXEC LNKEDT
```

VSAM/VTAM Common Macros (5745-SC-VCM)

The following macros are shared by VSAM and VTAM.

Source Statement Library

Number of Macros	Number of Blocks
11	770

Macros (Edited):

ACB
EXLST
GENCB
IKQACB
IKQERMAC
IKQEXLST
IKQRPL
MODCB
RPL
SHOWCB
TESTCB

Access Method Services (5745-SC-AMS)

Core Image Library

Phases	82
Blocks	542

Phases:

IDCAL01
IDCAMS
IDCBI01
IDCCDAL
IDCCDBI
IDCCDCL
IDCCDDE
IDCCDDL
IDCCDLC
IDCCDLR
IDCCDMP
IDCCDPM
IDCCDPR
IDCCDRC
IDCCDRM
IDCCDRP
IDCCDRS
IDCCDVY
IDCCDXP
IDCCL01
IDCDB01
IDCDB02
IDCDE01
IDCDI01
IDCDI02
IDCDI03
IDCDI04
IDCDI05
IDCDI06
IDCDI07
IDCDI08
IDCDI09
IDCDI10
IDCDI11
IDCDI12
IDCDI13
IDCDI14
IDCDI15
IDCDL01
IDCEX02
IDCEX03
IDCIO02
IDCIO03
IDCLC01
IDCLR01
IDCMP01
IDCPM01
IDCPR01
IDCRC01
IDCRIKT
IDCRILT
IDCRI01
IDCRM01
IDCRP01
IDCRS01
IDCSA04
IDCSA05
IDCTP04
IDCTP05
IDCTP06
IDCTSAL0
IDCTSBI0
IDCTSDE0
IDCTSDL0
IDCTSEX0
IDCTSIO0
IDCTSLC0
IDCTSLC1
IDCTSLR0
IDCTSLR1
IDCTSMP0
IDCTSPR0
IDCTSRC0
IDCTSRI0
IDCTSRS0
IDCTSTP0
IDCTSTP1
IDCTSTP6
IDCTSUV0
IDCTSXP0
IDCVY01
IDCXP01

Relocatable Library

Number of Modules	Number of Blocks
106	2,391

Modules:

IDCAL01
IDCBI01
IDCCDAL
IDCCDBI
IDCCDCL
IDCCDDE
IDCCDDL
IDCCDLC
IDCCDLR
IDCCDMP
IDCCDPM
IDCCDPR
IDCCDRC
IDCCDRM
IDCCDRP
IDCCDRS
IDCCDVY
IDCCDXP
IDCCL01
IDCCMZ1
IDCCMZ2
IDCDB01
IDCDB02
IDCDE01
IDCDE02
IDCDE03
IDCDI01
IDCDI02
IDCDI03
IDCDI04
IDCDI05
IDCDI06
IDCDI07
IDCDI08
IDCDI09
IDCDI10
IDCDI11
IDCDI12
IDCDI13
IDCDI14
IDCDI15
IDCDL01
IDCEX01
IDCEX02
IDCEX03
IDCIO01
IDCIO02
IDCIO03
IDCLC01
IDCLC02
IDCLR01
IDCLR02
IDCMP01
IDCPM01
IDCPR01
IDCRC01
IDCRC02
IDCRC03
IDCRC04
IKQRIFF
IDCRIKT
IDCRILT
IDCRI01
IDCRI02
IDCRI03
IDCRM01
IDCRP01
IDCRS01
IDCRS02
IDCRS03
IDCRS04
IDCRS05
IDCRS06
IDCRS07
IDCSA01
IDCSA02
IDCSA03
IDCSA04
IDCSA05
IDCSA08
IDCTP01
IDCTP04
IDCTP05
IDCTP06
IDCTSAL0
IDCTSBI0
IDCTSDE0
IDCTSDL0
IDCTSEX0
IDCTSIO0
IDCTSLC0
IDCTSLC1
IDCTSLR0
IDCTSLR1
IDCTSMP0
IDCTSPR0
IDCTSRC0
IDCTSRI0
IDCTSRS0
IDCTSTP0
IDCTSTP1
IDCTSTP6
IDCTSUV0
IDCTSXP0
IDCVY01
IDCXP01

Link-Edit Statements:

```
// OPTION CATAL
   INCLUDE IDCCMZ1
// EXEC LNKEDT
// OPTION CATAL
   INCLUDE IDCCMZ2
// EXEC LNKEDT
```

Storage Requirements

VSAM

Working Set

VSAM Access Routines
VSAM runs in SVA. This means that one copy of the VSAM modules is shared by all partitions. When paging is considered, the working set for these modules is as follows:

• Basic	22K
• If any file is being loaded or extended, or is causing a lot of control interval splits, then add	+4K
• If both count-key-data and FBA devices are in use at the same time, then add	+4K
• If any program is processing a base cluster via an alternate index path (path entry), then add	+6K
• If any partition is using VSAM Shared Resources (LSR) to share control blocks among files, then add	+2K

Partition Requirement for Control Blocks and Buffers
Calculate each partition's storage requirements in the following manner:

- A basic requirement of 40K for the master catalog.
- If VSAM Shared Resources (LSR) is used to share control blocks among some files, then the requirement for the VSAM resource pool must be taken into account. See BLDVRP macro in *Using VSE/VSAM Commands and Macros.* (When paging is considered, this value applies whether or not there is a real storage requirement.)
- For each file, add the applicable numbers in *one* column of the chart below. The following symbols are used:

  u=Number of alternate indexes in the upgrade set.

  n=Bufferspace. Depends on the control interval size(s) and on buffer specifications that can be made in the Access Method Services DEFINE command, in the DLBL statement, and/or in the ACB macro. (See "Buffer Specification" in Chapter 9.) If upgrade is being done, one set of buffers serves for all alternate indexes in the upgrade set. This set of buffers includes two data buffers and one index buffer.

	No LSR No Path Input	**No LSR No Path Output***	**No LSR Path Input**	**No LSR Path Output***	**LSR No Path Input**	**LSR No Path Output***	**LSR Path Input**	**LSR Path Output***
Basic requirement (minimum)	4K	4K	10K	10K	2K	2K	6K	6K
Upgrade set (minimum)	0	(*u*+1)x2K	0	*u*x2K	0	*u*x2K	0	(*u*-1)x2K
Buffers for base cluster	n	n	n	n	0	0	0	0
Buffers for alternate index (path)	0	0	n	n	0	0	0	0
Upgrade buffers (if there is an upgrade set)	0	n	0	n	0	0	0	0

* The file may be opened for output only, or for output and input.

This requirement for each file must be taken into account whether or not paging is considered.

A file may exceed minimum requirements under any of the following conditions:

- If the file has key ranges associated with it.
- If the file has more than one extent for data or for index.
- If SHAREOPTIONS(4) is used.
- If the length of the key field is very long.
- If the ACB or RPL is not created by GENCB, with the space allocation left up to VSAM, or if the GENCB requests are not done in the following sequence:

  GENCB ACB
  GENCB RPL

- If the file is opened for more than one string (STRNO>1).

Virtual Storage

Consider the following when calculating virtual storage requirements for VSAM:

- The master catalog requires 40K for basic buffers and control blocks.
- During open processing, an additional 6K is required for open control blocks.
- For each file, the amount of virtual storage required is equal to the working set.

VSE/VSAM Space Management for SAM Feature

Working Set

SAM Access Routines

The working set is the same as for unmanaged SAM access. VSE/VSAM does not require additional working storage for managed-SAM access.

Partition Requirement for Control Blocks and Buffers

VSE/VSAM requires you to specify additional working storage for the following:

• Control blocks	2K
• Buffer	must equal CISIZE

DTFPH

The working set for DTFPH is determined by the user program.

Virtual Storage

The following virtual storage requirements are in addition to those for VSAM.

- During open processing, an additional 4K is required for open control blocks.
- For each file, the amount of virtual storage required is equal to the working set, except for DTFPH access, in which case the virtual storage requirement is determined by the user program.

ISAM Interface Program

Working Set

Add 6K (for interface translation modules) to the working set previously determined for VSAM record management modules. IIP modules are non-SVA-eligible and must be loaded into each partition where they are used.

Virtual Storage

IIP is non-SVA-eligible and runs in the requestor's partition. In addition to the storage required for buffers and control blocks the sizes shown below are required for IIP phases:

IIPOPEN	1,300 bytes
IIPPROC	3,750 bytes
IIPCLOSE	575 bytes
IIPAMDTF	800 bytes

Access Method Services

Working Set

To operate efficiently, the Access Method Services working set requires a working set of approximately 64K bytes.

Virtual Storage

In addition to the basic allocation for VSAM, Access Method Services requires up to 497K bytes of virtual storage in the partition in which it is to run. Unlike VSAM phases, the Access Method Services modules cannot be loaded into the SVA.

The root modules, comprising 28K bytes, are loaded into the virtual address area when the user wants to perform any of the following functions. 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) is 118K. In addition, you must provide virtual storage for the specific Access Method Services commands to be processed. If you know what your functional mix is in the job stream, you can calculate a more precise virtual size from the following table.

Function	Size (bytes)
ALTER	23,300
BLDINDEX	24,000 [1]
CANCEL	2,000
DEFINE	57,500
DELETE	10,000
EXPORT	23,700
EXPORTRA	65,300 [2]
IMPORT	35,500 [3]
IMPORTRA	36,300
LISTCAT	52,400
LISTCRA	39,000
PRINT	8,000
REPRO	17,000
RESETCAT	125,000
VERIFY	2,000
Total	497,000

[1] This does not include the storage required for sorting records. Refer to *Using VSE/VSAM Commands and Macros.*

[2] This includes the 20,000 bytes required for the EXPORTRA command table.

[3] This includes the 3,500 bytes required for the IMPORT command table.

Chapter 4: Job Control

This chapter describes only those job control statements or operands whose meaning is different for VSE/VSAM than for other access methods. The information here supplements that contained in *VSE/Advanced Functions System Control Statements.*

Use of DLBL, EXTENT, and ASSGN Statements

In many jobs you can omit DLBL, EXTENT, and ASSGN statements from your job control. The following tables show which statements are required in the job control for the catalog and the file when you are running VSAM application programs and Access Method Services commands.

Catalog Job Control Statements

VSAM Application Programs

All VSAM application programs (including ISAM programs accessing VSAM via the IIP and SAM programs accessing SAM ESDSs via DTFs) must specify a DLBL statement for the master catalog; no EXTENT statement is necessary. (The DLBL statement may be in the job stream or in the partition or standard label area.) If the program is accessing a file in a user catalog, you must supply either a job catalog DLBL for that user catalog or VSAM file DLBL(s) with the CAT=*filename* parameter pointing to a user catalog DLBL statement. In either case, you do not need to supply EXTENT and ASSGN statements.

Note that if an application program is accessing files in more than one catalog, you must specify a user catalog DLBL for all files not in the job's default catalog.

Access Method Services Commands

You can always omit EXTENT and ASSGN statements from the job control you specify to identify the catalog you are using. A DLBL statement may be required; in several cases, you can omit the DLBL statement only if you specify the name of the catalog via the *catname* subparameter to the CATALOG, WORKCAT, or MODEL parameter (depending on which command is issued). Figure 4-1 shows when you must specify a DLBL statement for a job catalog and when applicable, a user catalog (not IJSYSUC). *A DLBL statement (specifying IJSYSCT) for the master catalog is always required..*

<table>
<tr><td>ALTER</td><td colspan="7">No job catalog DLBL is required, but you must specify CATALOG <i>(catname)</i> in the command if the catalog to be referenced is not the master catalog, or if a password is required.</td></tr>
<tr><td rowspan="3">BLDINDEX</td><td>If the index is in the
If the workfile is in the</td><td>MCAT
MCAT</td><td>UCAT1
MCAT</td><td>MCAT
UCAT1</td><td>UCAT1
UCAT2</td><td>MCAT
none</td><td>UCAT
none</td></tr>
<tr><td>You must specify</td><td>no job cat DLBL; no BLDINDEX CATALOG parameter*</td><td>job cat DLBL; BLDINDEX CATALOG (MCAT)</td><td>no job cat DLBL; BLDINDEX CATALOG (UCAT1)</td><td>job cat DLBL (UCAT1); BLDINDEX CATALOG (UCAT2)</td><td>no job cat DLBL; no BLDINDEX CATALOG parameter*</td><td>job cat DLBL; no BLDINDEX CATALOG parameter*</td></tr>
<tr><td colspan="7">*Unless a password is required, in which case you must specify the CATALOG parameter.</td></tr>
<tr><td>CANCEL</td><td colspan="7">A job catalog DLBL is not applicable.</td></tr>
<tr><td>DEFINE AIX, CLUSTER, or PATH</td><td colspan="7">No job catalog DLBL is required, but you must specify CATALOG <i>(catname)</i> and MODEL <i>(catname)</i> (if applicable) in the command whenever the catalog to be referenced is not the master catalog, or if a password is required.</td></tr>
<tr><td>DEFINE NONVSAM, or SPACE</td><td colspan="7">No job catalog DLBL is required, but you must specify CATALOG <i>(catname)</i> in the command if the catalog to be referenced is not the master catalog, or if a password is required.</td></tr>
<tr><td>DELETE</td><td colspan="7">No job catalog DLBL is required, but you must specify CATALOG <i>(catname)</i> in the command if the catalog to be referenced is not the master catalog, or if a password is required.</td></tr>
<tr><td>EXPORT</td><td colspan="7">A job catalog DLBL (IJSYSUC) is required if the catalog to be referenced is not the master catalog.</td></tr>
<tr><td>EXPORTRA</td><td colspan="7">A job catalog DLBL is not applicable.</td></tr>
<tr><td>IMPORT</td><td colspan="7">No job catalog DLBL is required, but you must specify CATALOG <i>(catname)</i> in the command if the catalog to be referenced is not the master catalog, or if a password is required.</td></tr>
<tr><td>IMPORTRA</td><td colspan="7">No job catalog DLBL is required, but you must specify CATALOG <i>(catname)</i> in the command if the catalog to be referenced is not the master catalog, or if a password is required.</td></tr>
<tr><td>LISTCAT</td><td colspan="7">No job catalog DLBL is required, but you must specify CATALOG <i>(catname)</i> in the command if the catalog to be referenced is not the master catalog, or if a password is required.</td></tr>
<tr><td>LISTCRA</td><td colspan="7">You do not need to specify a user or job catalog DLBL, regardless of whether CATALOG <i>(catname)</i> is specified.</td></tr>
<tr><td>PRINT</td><td colspan="7">INFILE in master catalog — Do not specify a user catalog DLBL or a job catalog DLBL.
INFILE in user catalog — Specify either a user catalog DLBL (CAT= parameter) or a job catalog DLBL (IJSYSUC).
INFILE is nonVSAM — A user catalog DLBL or a job catalog DLBL statement is not applicable.</td></tr>
<tr><td>REPRO</td><td colspan="7">INFILE and OUTFILE in same catalog:
master catalog — Do not specify a user catalog DLBL or a job catalog DLBL.
user catalog — Specify either a user catalog DLBL (CAT= parameter) or a job catalog DLBL (IJSYSUC).
INFILE and OUTFILE in different catalogs:
Specify a user catalog DLBL for each catalog that is not the default catalog.</td></tr>
<tr><td>RESETCAT</td><td colspan="7">No job catalog DLBL is required, but you must specify CATALOG <i>(catname)</i> and WORKCAT <i>(catname)</i> in the command if CATALOG or WORKCAT is not the master catalog, or if a password is required.</td></tr>
<tr><td>VERIFY</td><td colspan="7">A job catalog DLBL (IJSYSUC) is required if the catalog to be referenced is not the master catalog, or if a password is required.</td></tr>
</table>

Figure 4-1. Job Catalog and User Catalog DLBL Statement Requirements
(A master catalog DLBL statement is always required.)

File Job Control Statements

In specifying job control statements for user files, DLBL, EXTENT, and ASSGN statements may or may not be required. Figure 4-2 indicates when you should specify these statements.

File Job Control			
Type of Processing	**DLBL Required**	**EXTENT Required**	**ASSGN Required**
Files to be implicitly opened. (For example, accessing a file via an AIX or path during which VSAM must open index files without user specification.)	No	No	No
Files to be explicitly opened. (The DLBL *filename* must match the ACB DDNAME parameter, or the ACB name if DDNAME is omitted, and the *file-ID* must be the name of the object being opened.)	Yes	No	No
ISAM programs accessing VSAM files via the ISAM Interface Program.	Yes	No	No
SAM programs accessing SAM ESDSs via DTFs. (The DLBL *filename* must match the DTFxx name field.)	Yes	No*	No
Access Method Services Commands			
ALTER	No	No	No
BLDINDEX	No	No	No
CANCEL	No	No	No
DEFINE AIX/CLUSTER UNIQUE	Yes	Yes	No
DEFINE AIX/CLUSTER not unique	No	No	No
DEFINE MASTERCATALOG	Yes	No	No
DEFINE NONVSAM/PATH/SPACE	No	No	No
DEFINE USERCATALOG	No	No	No
DELETE	No	No	No
EXPORT(RA) OUTFILE (SAM file on DASD)	Yes	Yes	Yes
EXPORT(RA) all other	No	No	No
IMPORT INFILE (SAM file on DASD)	Yes	Yes	Yes
IMPORT OBJECTS FILE UNIQUE unless predefined	Yes	Yes	No
IMPORT all other	No	No	No
IMPORTRA INFILE (SAM file on DASD)	Yes	Yes	Yes
IMPORTRA OBJECTS FILE UNIQUE	Yes	Yes	No
IMPORTRA all other	No	No	No
LISTCAT	No	No	No
LISTCRA	No	No	No
PRINT VSAM file	Yes	No	No
PRINT nonVSAM file (SAM or ISAM file on DASD)	Yes	Yes	Yes
REPRO VSAM file	Yes	No	No
REPRO nonVSAM file (SAM or ISAM file on DASD)	Yes	Yes	Yes
RESETCAT	No	No	No
VERIFY	No	No	No
*Exception: An EXTENT statement is required for the implicit define of an output or work file for which no implicit model exists.			

Figure 4-2. File Job Control Statement Requirements

DLBL

Refer to the beginning of this chapter to determine when you must specify a DLBL statement.

If you specify many DLBL parameters, you may need to use a continuation statement. If so, column 72 (on the first statement) must contain a continuation character. The columns between the last comma and the continuation character must be blank, and the continuation statement must start in column 16 (no // in columns 1 and 2).

```
// DLBL filename,['file-ID'],[date],[codes][,DSF]
         [,BUFSP=n][,CAT=filename][,DISP=disposition]
         [,RECORDS=n|(n1,n2)][,RECSIZE=n]
```

filename

For VSE/VSAM, filename (1-7 characters) is identical to the:

- dname of the FILE (dname) parameter in an Access Method Services command, and
- DDNAME=filename parameter of the Access Method Control Block (ACB) in the processing program that identifies the file. If DDNAME is omitted, the filename must be placed in the symbolic name field of the ACB.

'file-ID'

For VSE/VSAM, specify 'file-ID' when acccessing a file. The file-ID is identical to the name of the file that was specified in the DEFINE command of Access Method Services and listed in the VSAM catalog.

When a new VSAM data space or file is being created (defined), the file-ID is ignored if it is specified.

date

This parameter is ignored for VSE/VSAM; the date used is that specified in the Access Method Services DEFINE command. VSAM files (that have been explicitly defined) or data spaces can only be deleted through the DELETE command, even though the expiration date has been reached.

codes

Specify the word VSAM.

DSF

This operand is ignored because all VSAM files are data secured.

BUFSP=n

For VSE/VSAM, this operand specifies the number of bytes of storage (0 - 999999) to be allocated as buffer space for the file. It overrides both the BUFFERSPACE parameter of the Access Method Services DEFINE command and the BUFSP operand in the ACB macro, if its value is higher.

CAT=filename

This operand specifies the filename (1 through 7 alphameric characters) of the DLBL statement for the catalog owning this VSAM file. The system searches only this catalog for the file-ID when the VSAM file is to be opened. Specify this operand only if you want to override the system's assumption that the job catalog or, if there is no job catalog, that the master catalog owns the file.

The only Access Method Services commands that use the CAT operand to specify a non-default catalog are the PRINT and REPRO commands.

Job catalogs are discussed under the heading "VSAM Catalogs" in Chapter 5.

DISP=disposition

Specify one of the following values for disposition:

```
NEW
(NEW,KEEP)
(NEW,DELETE)
(NEW,DATE)
OLD
(OLD,KEEP)
(OLD,DELETE)
(OLD,DATE)
(,KEEP)
(,DELETE)
(,DATE)
```

The DISP parameter describes what is to be done with a reuseable file during open and close. Further information appears under "File Disposition" later in this chapter.

NEW indicates that the file is to be reset at open.

OLD indicates that the file is not to be reset at open.

,KEEP indicates that the file is to be kept at close.

,DELETE indicates that the data is to be made inaccessible at close.

DATE indicates that the disposition is the same as for KEEP (if the expiration date has not been reached) or DELETE (if the expiration date has been reached).

The RECORDS and RECSIZE parameters are used to determine allocation sizes for implicit file definition during managed-SAM open of a SAM ESDS.

RECORDS=n|(n1,n2) (VSE/VSAM Space Management for SAM Feature only)

If you specify RECORDS=n, n indicates the number of records for primary allocation of the file.

If you specify RECORDS=(n1,n2), n1 indicates the number of records for promary allocation, and n2 indicates the number of records for secondary allocation. If you do not specify a value for n2, 20% of n1 is used. Do not specify 0 as a value for n1.

If you specify the RECORDS parameter, you must also specify the RECSIZE parameter.

RECSIZE=n (VSE/VSAM Space Management for SAM Feature only)

n indicates the average record length for the file. Do not specify 0 as a value for n.

If you specify RECSIZE, you must also specify the RECORDS parameter.

EXEC

```
// EXEC  {,SIZE=nK|AUTO|(AUTO,nK)}
```

SIZE

Specifies how much storage is needed for loading the specified module.

n

Must be greater than zero and should be a multiple of 2. (If not, the system rounds the value up to the nearest 2K boundary.)

AUTO

Indicates that the program size, as calculated by the system from information in the core image directory, is to be taken as the value for SIZE. For ease of use, specifying SIZE=AUTO is recommended for Access Method Services.

(AUTO,nK)

Indicates that job control must take program size plus nK bytes as the value for SIZE. (If this value is not a multiple of 2, the value is rounded up.)

You must specify SIZE for VSE/VSAM programs (including IDCAMS), ISAM programs using the ISAM Interface Program (IIP), and SAM files using the VSE/VSAM Space Management for SAM Feature. Specifying SIZE (without the REAL parameter) means that the partition to be used for the job step is divided into two parts. The lower part (with a size derived from the AUTO and nK parameters) contains the program to be executed. The upper part serves as a storage pool for the partition.

The non-SVA-eligible VSAM phases and Access Method Services must be accommodated in the partition GETVIS area. The partition GETVIS area must contain at least 40K for VSAM buffers and control blocks for each catalog that is open, plus 12K for each KSDS and 10K for each ESDS or RRDS (assuming a CI size of 2K or less). Additional space for modules, buffers, and control blocks is required if you use any non-SVA-eligible VSAM phases (for example, ISAM Interface Program) or Access Method Services. Exact storage requirements can be calculated on the basis of the information contained in Chapter 3 of this manual.

To invoke Access Method Services via job control, specify:

```
// EXEC IDCAMS,SIZE=AUTO
```

If you do not specify the SIZE parameter, Access Method Services terminates your job immediately. SIZE provides information to the system to allow it to divide the processing partition into a static area and a GETVIS area. When you execute IDCAMS, only the first load, called the root segment, is loaded into the static area. The remainder of the partition must be left free for GETVIS area required by Access Method Services and for the subsequent modules it loads. The value in the SIZE parameter should be just large enough to contain the root segment. If you specify an amount greater than required, the excess is lost to the processing program. When you specify SIZE=AUTO, the system determines the amount of storage required for the IDCAMS root segment and leaves the rest of the partition free for the GETVIS area.

EXTENT

Refer to the beginning of this chapter to determine when you must specify an EXTENT statement.

```
// EXTENT [symbolic unit],serial number,[type],
          [sequence number],[relative track|block address],
          [number of tracks|number of blocks]
```

symbolic unit

Specify a six-character field indicating the symbolic unit (SYSxxx) of the volume for which this extent is effective. VSE/VSAM does not require this parameter; if you do not specify a symbolic unit, VSAM will assign one.

serial number

VSE/VSAM users are required to specify from one to six characters indicating the volume serial number of the volume this extent is on.

type

For VSE/VSAM users, a value of 1 is assumed.

sequence number

This operand is ignored for VSE/VSAM users, but if it is specified incorrectly it will be flagged by job control syntax checking.

relative track|block address

This operand indicates the number of the track (CKD) or block (FBA) on which the extent is to begin. You must specify it when a file with the UNIQUE option is being created (Access Method Services DEFINE, IMPORT, or IMPORTRA command). This operand is not required, and it is ignored if it is specified when a VSAM file is created within an existing data space. In this case, the space for the file is suballocated by VSAM from direct access extents it already owns. You are not required to specify this operand for a VSAM input file because the extents are obtained from the VSAM catalog.

number of tracks|number of blocks

This operand indicates the number of tracks (CKD) or blocks (FBA) to be allocated to the file or space. You must specify it when a file with the UNIQUE option is being created (Access Method Services DEFINE, IMPORT, or IMPORTRA command). This operand is ignored when a VSAM file is created within an existing data space, because the space for the file is suballocated by VSAM from direct access extents it already owns. This operand is also not required for VSAM input files because the extents are obtained from the VSAM catalog.

For an implicitly defined SAM ESDS that does not specify RECORDS and RECSIZE (on the DLBL statement), VSAM uses the number of tracks|number of blocks parameter to determine primary allocation size. A secondary allocation size equal to 20% of primary is used.

File Disposition

For *VSAM* access, the options available at open and the disposition of the file at close depend on the DISP parameter of the DLBL statement or the MACRF/CLOSDSP parameters of the ACB macro. Options specified for DISP override those specified for MACRF/CLOSDSP. The default for the DISP parameter depends on the file being opened or closed. For VSAM access, the default DISP=(OLD,KEEP).

For *managed-SAM* access, the options available at open and the disposition of the file at close depend on the DISP parameter of the DLBL statement and options specified in the DTF.

Each option of the DISP parameter has a corresponding option in the MACRF/CLOSDSP parameters that causes the same function to be performed. The NEW, OLD, RST, and NRS options apply when the file is being opened; KEEP, DELETE, and DATE apply when the file is being closed. VSE/VSAM *allocates* space, *resets* the file, or *implicitly defines* a file (for managed-SAM access of a SAM ESDS that is not already defined in the catalog) when the ACB/DTF for the file is opened. At close, VSE/VSAM *keeps, resets, deallocates,* or *deletes* the file, depending on which function has been specified.

Note: The definitions given below apply to the terminology used in Figures 4-3 through 4-6. For VSAM access, refer to Figure 4-3 for open disposition and Figure 4-5 for close disposition. For managed-SAM access, refer to Figure 4-4 for open disposition and Figure 4-6 for close disposition.

Keep means to retain a file's data and accessibility.

Reset means to set a file to empty and release its secondary extents.

Deallocate means to set a file to empty and release its primary and secondary extents.

Allocate means to provide primary DASD space, as specified by the user at DEFINE time.

Define means to place information describing the file into the VSAM catalog.

Delete means to remove all references to the file from the catalog, and release the file's space.

A file may appear in one of four states when it is opened for output:

Unallocated: A file is unallocated if its catalog records have no information of suballocated space. This happens for one of two reasons. Either the file was defined as a dynamic file and it has never been opened, or the file was defined as a dynamic file and it has been closed with DISP and/or MACRF/CLOSDSP options that caused deallocation.

All open options cause space to be suballocated for the file, provided enough space is available. If enough space is not available, the open fails. The ACB user is informed by an ACB return code; the DTF user's job is cancelled.

Allocated for Native VSAM Access: The options DISP=NEW and/or MACRF=RST cause the file to be reset to its primary allocation; its secondary extents are released. Although the file records are not erased, the file is considered empty. The options DISP=OLD and/or MACRF=NRS do not cause the file to be reset to empty and allow updating and extension of the file.

Allocated for Managed-SAM Access of a SAM ESDS: Same as for "Allocated for Native VSAM Access."

Undefined for Managed-SAM Access: All options of the DISP parameter cause the file to be implicitly defined. The native VSAM user cannot implicitly define a file.

When a file with suballocated space is opened for input, the options DISP=NEW and/or MACRF=RST are invalid, and the options DISP=OLD and/or MACRF=NRS cause the file to be opened without resetting the file to empty. The following figures show the action performed by VSE/VSAM when you try to open different kinds of files.

Files with REUSE Attribute			
OPEN (ACB)	**File Status**	**DISP on DLBL/MACRF on ACB**	
		NEW/RST	**OLD/NRS**
OUT	**Unallocated**	Allocate space for file.	Allocate space for file.
	Allocated	Reset file. (DISP=NEW prevents access to any data existing prior to open.)	File is not reset. Output operations allow updating and extension of the file.
	Undefined	Open fails.	Open fails.
IN	**Allocated**	Open fails.	File is not reset. If file is already empty, open fails.

Figure 4-3. VSAM-Access Open Disposition

Files with REUSE Attribute				
OPEN (DTF)	File Status	DISP Unspecified (Default Value in Parentheses)	DISP on DLBL[2]	
			NEW	OLD
OUTPUT	Unallocated SAM ESDS[1]	Allocate space for file (DISP= NEW).	Allocate space for file.	Allocate space for file.
	Allocated for SAM ESDS[1]	Reset file. Position to beginning of file (DISP= NEW).[2]	Reset file. Position to beginning of file.	File is not reset. Position to end of file for extension.
	Undefined	Implicitly define a SAM ESDS (DISP= NEW).	Implicitly define a SAM ESDS.	Implicitly define a SAM ESDS.
INPUT	Allocated for SAM ESDS	File is not reset. Position to beginning for input (DISP= OLD).	Invalid - file is not reset. Open fails.	File is not reset. Position to beginning for input.
WORK	Unallocated SAM ESDS[1]	Allocate space for file (DISP= NEW).	Allocate space for file.	Allocate space for file.
	Allocated for SAM ESDS[1], [3]	Reset file. Position to beginning of file (DISP= NEW).	Reset file. Position to beginning of file.	File is not reset. Position to beginning of file. (File may be read.)
	Undefined	Implicitly define a SAM ESDS (DISP= NEW).	Implicitly define a SAM ESDS.	Implicitly define a SAM ESDS.

[1] If the file's characteristics do not match those specified in the DTF, the open fails and the file cannot be implicitly deleted. In particular, the maximum logical block that may be written (DTF BLKSIZE) must not be greater than the maximum allowed in the file (DEFINE maximum RECORDSIZE). If DTFSD is used, the file must be in CI format.

[2] Do not specify the DISP parameter for IJSYSLN (SYSLNK file).

[3] DISP=NEW prevents access to any data existing prior to open.

Figure 4-4. Managed-SAM-Access Open Disposition

The following figures show the action performed by VSE/VSAM when you try to close different kinds of files.

Files with REUSE Attribute				
CLOSE (ACB)	DISP on DLBL/CLOSDSP on ACB			
	KEEP	DELETE	DATE	
			Expired	Unexpired
File was explicitly defined (NOALLOCATION)	Keep	Deallocate	Deallocate	Keep
Reusable (Suballocated)	Keep	Reset	Reset	Keep
File was implicitly defined	Keep	Reset	Reset	Keep

Figure 4-5. VSAM-Access Close Disposition

If you specify DELETE at close, VSAM deletes the data via deallocation, resetting the file, or implicit deletion. The contents of the file are lost. The next open for INPUT will fail because the file is empty. If any other DTF/ACB is open for the same file, however, the close is completed, but the file is not reset, deallocated, or deleted, and the operator and the invoking program are notified by a return code.

Files with REUSE Attribute					
CLOSE (DTF)	DISP not specified	DISP on DLBL[2]			
		KEEP	DELETE	DATE	
				Expired	Unexpired
File was explicitly defined (NOALLOCATION)	Keep[1]	Keep	Deallocate	Deallocate	Keep
Reusable (suballocated)	Keep[1]	Keep	Reset	Reset	Keep
File was implicitly defined	Keep[1]	Keep	Delete	Delete	Keep

[1] DISP is DELETE if TYPEFLE=WORK and DELETFL is unspecified.

[2] Do not specify the DISP parameter for IJSYSLN (SYSLNK file).

Figure 4-6. Manged-SAM-Access Close Disposition

If you request DELETE at close, VSAM deletes the data by deallocation, resetting, or implicit deletion. In order to avoid sharing problems, however, if any other DTF (or ACB) for the same file is open at the same time, no deletion occurs, the operator is notified by a message with a warning return code, and close processing continues. With DELETE specified at close, the contents of the file are lost. The next open for OUTPUT WORK will write new data. If the file has been deallocated or reset, an OPEN for INPUT will be successful, but the first GET will cause control to be passed to the EOFADDR routine because the file is empty.

Additional Considerations

- Specifying DISP=NEW on the DLBL statement overrides MACRF=NRS in the ACB, such that the result is as if MACRF=RST were specified. Because MACRF=RST is mutually exclusive with MACRF=IN and MACRF=LSR, open fails if DISP=NEW is specified for a file opened via DTF TYPEFLE=INPUT or ACB MACRF=IN or MACRF=LSR.
- If the close disposition specified for the file results in the resetting or deallocation of the file, and if the file is password-protected, the ACB must specify (or the operator will be prompted for) the update- or higher-level password of the file at open. If the close disposition specified for the file results in the implicit deletion of the file, there is no prompting for the entry password because an implicitly defined file cannot be password-protected. If the catalog itself is password-protected, the operator is prompted for the master password of the catalog at close.

  Using DISP could eliminate data inadvertently if the wrong parameter is specified. You may want to use an entry password to protect against inadvertent destruction of data. A catalog password may also provide protection for files owned by the catalog. If the file is being accessed via DTF, the password must be supplied by the operator. If the file is being accessed via ACB, the password may be supplied in the ACB, by the operator, or through Access Method Services commands.

Access Method Services Input and Output Files

When a VSAM file is to be *openedfor access* by an Access Method Services PRINT or REPRO command, you must supply the following job control information:

```
// DLBL filename,'file-ID',,VSAM
```

The *filename* identifies the file and matches the *dname* of an INFILE or OUTFILE parameter in the command. The *file-ID* is identical to the name of the file (*entryname, catname, or newname* parameter) specified in the commands of Access Method Services and listed in the VSAM catalog.

Optionally, the BUFSP parameter of the DLBL statement can be specified to override:

- The cataloged BUFFERSPACE parameter (only if the BUFSP parameter is equal to or greater than the cataloged BUFFERSPACE parameter).
- The buffer space value (in the ACB macro) defaulted to by Access Method Services.

Note: Because EXPORT, EXPORTRA, IMPORT, and IMPORTRA always default to an optimum buffer space value (via the ACB macro parameter) VSAM ignores any BUFSP specification made by you in the DLBL statement associated with these commands. All other commands use the VSAM default values in their ACB macros.

For more information on buffer space options and defaults, see the ACB macro in *Using VSE/VSAM Commands and Macros.*

Use SYS004 for magnetic tape input and SYS005 for magnetic tape output. Access Method Services supports unlabeled and standard labeled tapes for magnetic tape input and output. You have the option of rewinding or not rewinding input and output files.

You must use SYSLST as the output file for listing. Print lines are 121 bytes in length. The first byte is the ANSI control character. The default parameters of this file are:

- Record format: Fixed, unblocked with ANSI carriage control
- Logical record length: 121
- Block size: 121
- Printed lines per page: 56. Use the job control SET LINECT command or the STDJC supervisor generation macro instruction if you want to specify the value (between 30 and 99) for SYSLST printed lines per page.

The chapter "VSAM Catalogs" describes the job control statements to be used for VSAM master and user catalogs. The chapter "Using Access Method Services" in *Using VSE/VSAM Commands and Macros* describes the job control statements to be used for VSAM data spaces and files.

Magnetic Tape Considerations for Access Method Services

The following commands support tape files as input or output via the subparameters of the ENVIRONMENT parameter:

- Tape output: EXPORT, EXPORTRA
- Tape input: IMPORT, IMPORTRA, PRINT
- Tape input or output: REPRO.

The NOLABEL and STDLABEL subparameters (of the above commands) allow you to process an existing tape file or create a new tape file with no tape labels (NOLABEL) or with EBCDIC standard labels (STDLABEL). VSE/VSAM does not support American National Standard (ASCII) labels (DTFMT macro, ASCII=YES) and nonstandard tape labels (DTFMT macro, FILABL=NSTD). User standard labels (DTFMT macro, LABADDR=xxxxx) are bypassed on input and not supported on output.

The NOREWIND, REWIND, and UNLOAD subparameters (of the above commands) allow you to control tape positioning for OPEN, CLOSE, and end-of-volume (EOV) processing.

- NOREWIND indicates that rewind is never performed. It becomes your responsibility to ensure that a tape is properly positioned before processing it with Access Method Services. You must specify NOREWIND if you wish to create or process multifile tape volumes.
- REWIND indicates that tapes are rewound to load point but not unloaded on OPEN, CLOSE, and end-of-volume (EOV) conditions. You should specify REWIND (or UNLOAD) whenever you wish to create or process single file tape volumes.
- UNLOAD indicates that tapes are rewound on an OPEN, and rewound and unloaded on a CLOSE and end-of-volume (EOV) condition. You should specify UNLOAD (or REWIND) whenever you wish to create or process single file tape volumes. Specify UNLOAD if you desire demounting when file creation or processing is complete.

You must specify REWIND or UNLOAD to process unlabeled tapes if VSE/Access Control is installed.

Tape Assignments for Access Method Services

You must assign magnetic tape input files to SYS004 and magnetic tape output files to SYS005.

Multivolume File Considerations for Access Method Services

If you are creating or processing a file which may require more than one tape volume, you must specify (or default to) the STDLABEL option. This is necessary because Access Method Services cannot distinguish between end-of-file and end-of-volume on a tape with no label; it therefore always assumes end-of-file.

Multifile Volume Considerations for Access Method Services

Standard Labels and Input: To process the first file on the tape, ensure that the tape is at load point (use the // MTC REW,SYS004 command) before processing and specify STDLABEL and NOREWIND (so that the succeeding file can be processed next).

To process any other file on the tape: If the previous command (or program) has processed the previous file and has not rewound the tape, specify STDLABEL and NOREWIND (this maintains positioning for OPEN and positions in front of the following file, if any, at CLOSE). If the tape is not positioned at the beginning of the file, rewind the tape (use the // MTC REW,SYS004 command) and then use the TLBL statement file-sequence-number parameter to identify the required file or use the // MTC FSF,SYS004,nn command to position to the beginning of the desired file. nn must equal three times the number of files to be bypassed (for example, to skip past 2 files, specify 6). In the processing command, specify STDLABEL and NOREWIND (for the reasons previously given).

Standard Labels and Output: To create the first file on the tape, a standard volume label (VOL1) should have been previously written as the first record on the selected tape volume (see *VSE/Advanced Functions Tape Labels*). Ensure that the tape is at load point (use the // MTC REW,SYS005 command) before processing and specify STDLABEL and NOREWIND (positions the tape for succeeding commands). On the file's TLBL statement, either omit the file-sequence-number parameter, or specify decimal 1.

To create any other file on the tape: If the previous command (or program) has processed the previous file and has not rewound the tape, specify

STDLABEL and NOREWIND (maintains positioning for OPEN and positions for a following file, if any, at CLOSE). If the tape is not correctly positioned, you must position the tape just beyond the tapemark following the EOF1 trailer record of the previous file. First, specify the // MTC REW,SYS005 command (this rewinds the tape to load point). From load point you can then properly position the tape by specifying the // MTC FSF,SYS005,nn command. nn is equal to three times the number of files on the tape preceding the file to be created. Specify STDLABEL and NOREWIND in the command (maintains positioning for OPEN and positions for a following file, if any, at CLOSE). You can, in all cases, omit the file-sequence-number parameter of the TLBL statement.

Unlabeled Tapes and Input or Output: To create or process the first file on the tape, ensure that the tape is at load point (use the // MTC REW,SYS004 command for input, and the // MTC REW,SYS005 command for output) before processing and specify NOLABEL and NOREWIND (allows the succeeding file to be processed or created next).

To create or process any other file on the tape: If the previous command (or program) has created or processed the previous file and has not rewound the tape, specify NOLABEL and NOREWIND (maintains positioning for OPEN and positions in front of the following file, if any, at CLOSE). If the tape is not positioned at the beginning of the file, rewind the tape (use the // MTC REW,SYSxxx command — SYSxxx is SYS004 for input and SYS005 for output). Next position to the beginning of the desired file with the // MTC FSF,SYSxxx,nn command (SYSxxx is SYS004 for input and SYS005 for output). For files created with Access Method Services, nn must equal the number of files to be bypassed (Access Method Services uses the DTFMT macro TPMARK=NO option). For each file to be bypassed, but created with the DTFMT macro TPMARK=YES specification, add two to nn.

Chapter 5: VSAM Catalogs

- A *master catalog* must be defined in your system; this is the first job that you run, after you have installed VSE/VSAM in your system. You can have more than one master catalog at your installation; however, only one can be connected to the system at a time.

- An optional number of *user catalogs* can be defined in your system. They are pointed to by the master catalog and have the same structure and function as the master catalog. User catalogs are used to increase data integrity and security, improve performance, and provide volume portability. Figure 5-1 shows the relationship between master and user catalogs, as well as their relationships with VSAM and nonVSAM files.

- VSAM catalog(s) are central information points for files and volumes; they contain the information VSAM needs to allocate space for files, verify authorization to gain access to them, compile usage statistics on them, and relate relative byte addresses (RBAS) to physical locations. Each VSAM catalog also contains entries that describe itself.

- The master catalog volume must always be mounted whenever a VSAM file or catalog is to be processed. If the VSAM file to be processed is defined in a user catalog, the user catalog volume must be mounted also.

- The master catalog volume is connected to the system at IPL (initial program load) by the DEF SYSCAT=cuu command. It is always on a logical unit named SYSCAT. A user catalog volume(s) can be on any programmer logical unit. Each catalog exists on a single volume which is *owned by* the catalog.

- The filename of a master catalog must be IJSYSCT.

- All VSAM files (except implicitly defined SAM ESDSs) must be defined (have an entry) in a catalog; use the Access Method Services DEFINE command. A file does not exist in VSAM until it has been defined in a catalog. Most uses of Access Method Services involve doing something to a VSAM catalog. For example, establishing a file involves creating an entry in the catalog; deleting a file involves removing an entry from the catalog; and moving a file from one system to another involves moving an entry from one system's catalog to another system's catalog.

- A volume is *owned by* or can be controlled by one catalog only; therefore all VSAM files on a specific volume must also be cataloged (have an entry) in that same catalog. If a VSAM file is located on several volumes, each of the volumes must be *owned by* that same catalog.

- A large number of requests for information from a catalog may result in some of the requests being answered more slowly than they would be if several catalogs contained parts of the information. If the master catalog primarily contains pointers to user catalogs, which in turn contain entries for most files and volumes, catalog search time can be reduced, and the effect of an inoperative or unavailable catalog is minimized.

Specifying a VSAM Catalog's Job Control Statements

One important point about VSAM catalogs needs to be understood; VSAM either uses a catalog to *access a file* (as in the PRINT command, where VSAM locates the file to be printed via the catalog), or it *accesses the catalog information only and does not access a file* (as in the ALTER command, where VSAM changes an entry in the catalog). The following discussion (except where noted) pertains to the accessing of a file.

Specifying the Master Catalog's Job Control Information

To define a master catalog you must supply a master catalog DLBL statement and specify extent information, either in the form of an EXTENT statement or by DEFINE command parameters.

```
// DLBL     IJSYSCT,'MASTCAT',,VSAM
```

If you place the master catalog's DLBL statement in the system or partition standard label area by preceding it with one of the following job control statements, you can omit the master catalog's DLBL statement from the job stream.

```
// OPTION   STDLABEL or
// OPTION   PARSTD
```

The DLBL statement in the above example identifies:

- The filename: must be IJSYSCT.
- The file-ID: MASTCAT. This can be any name you choose (it must match the NAME parameter in the DEFINE MASTERCATALOG command).
- The access method: VSAM.

One other way of referring to the master catalog (after its initial specification) is by coding the CAT=filename parameter in a VSAM file's DLBL statement. A further explanation of the CAT=filename parameter follows.

Specifying a User Catalog's Job Control Information

To *define* a user catalog you supply a DLBL statement for the *master* catalog only. But to *access* files in a user catalog, specify a user catalog DLBL statement. (Refer to Figure 4-1 for user catalog DLBL requirements for Access Method Services commands.) No EXTENT statement is required.

VSAM additionally provides you with the capability to designate one, and only one, of your user catalogs as a *job catalog*. When you specify a job catalog, VSAM will always use that one catalog for all catalog and file access in the current job, unless it is specifically overridden by:

- The CAT=filename parameter of a VSAM file's DLBL statement.
- The CATALOG or WORKCAT parameter of an Access Method Services command.

You specify a job catalog by coding the filename, IJSYSUC, in the DLBL statement that specifies the user catalog (see the following example).

```
// DLBL     IJSYSUC,'JOBCAT',,VSAM
```

Using a Job Catalog

The following example makes use of the REPRO command (data is to be copied from one file to another) to show how you use a job catalog. It is assumed that the input file, PAY, and the output file, PAYROLL, were already defined (cataloged) in the job catalog. It is also assumed that the DLBL statement for the master catalog has been placed in the system or partition standard label area and so need not be included (in the example).

```
     // JOB       Specify a job catalog
(a)  // DLBL      IJSYSUC,'USER1',,VSAM
(b)  // DLBL      VSAMIN,'PAY',,VSAM
(c)  // DLBL      VSAMOUT,'PAYROLL',,VSAM
     // EXEC      IDCAMS,SIZE=AUTO
        REPRO     INFILE(VSAMIN) OUTFILE(VSAMOUT)
     /*
     /&
```

In this example, VSAM finds a DLBL statement with a filename of IJSYSUC (a). VSAM interprets this to mean that files PAY (b) and PAYROLL (c) have their respective entries in the job catalog. It therefore searches the job catalog to locate the entry for input file PAY and output file PAYROLL.

What happens when you want to process a file that is not cataloged in the job catalog? VSAM provides two different ways of overriding the job catalog; you can use the DLBL CAT=filename parameter or the CATALOG parameter of an Access Method Services command.

Explicit Catalog Specification (With a VSAM File's DLBL CAT Parameter)

The following example directs VSAM to search a catalog other than the job catalog (specified in the previous example). Assume that the input file PAY was defined in job catalog USER1 as before, but output file PAYROLL was defined in user catalog USER2. Also assume, as before, that the DLBL statement for the master catalog has been placed in the system or partition standard label area.

```
     // JOB       Using the DLBL CAT parameter
(a)  // DLBL      IJSYSUC,'USER1',,VSAM
(b)  // DLBL      VSAMIN,'PAY',,VSAM
(c)  // DLBL      VSAMOUT,'PAYROLL',,VSAM,CAT=PRIVCAT
(d)  // DLBL      PRIVCAT,'USER2',,VSAM
     // EXEC      IDCAMS,SIZE=AUTO
        REPRO     INFILE(VSAMIN) OUTFILE(VSAMOUT)
     /*
     /&
```

VSAM will once again encounter the filename IJSYSUC in a DLBL statement (a) but it also finds the CAT=PRIVCAT parameter in a file's DLBL statement. The CAT=PRIVCAT parameter directs VSAM to search catalog USER2 (d) to locate PAYROLL's file entry instead of searching the job catalog. (Filename PRIVCAT links the CAT parameter to the appropriate DLBL user catalog statement.)

VSAM will locate the entry of file PAY (b) in the job catalog, as before, because in this case you have not overriden the job catalog specification.

The DLBL CAT=filename parameter is used with the PRINT and REPRO commands. (Each of these commands is used to access data.) The DLBL CAT=filename parameter can also be used for VSAM application program access.

Explicit Catalog Specification (With the Access Method Services CATALOG Parameter)

If you are using Access Method Services and specify the CATALOG parameter in a command, no DLBL statement is needed for a user catalog. Only a master catalog DLBL is required. (The master catalog DLBL statement may be supplied in the job stream or in the partition or standard label area.) The format of the CATALOG parameter is:

```
CATALOG   (catname[/password])
```

VSAM Hierarchy of Catalogs

From the preceding, you can see that VSAM follows a certain order in searching for a file's catalog. The established hierarchy that determines the specific catalog to be searched is as follows:

1. Explicit user or master catalog. This is the catalog that is specified by the Access Method Services CATALOG parameter or by the CAT=*filename* parameter of a VSAM file's DLBL statement.
2. Job catalog. If the above catalog is not specified, the job catalog (IJSYSUC) specified as the *filename* of a DLBL statement is searched.
3. Master catalog. If the above catalogs are not specified, the master catalog (IJSYSCT) is searched.

The following table shows which catalog is searched, depending on your DLBL specification.

// DLBL IJSYSUC Specified?	**DLBL CAT=filename**	**CATALOG Parameter Specified?**	**Catalog to be Searched**
Yes	None	No	Job Catalog
Yes	Filename of User Catalog DLBL	No	User Catalog
Yes	'IJSYSCT'	No	Master Catalog
Yes	'IJSYSUC'	No	Job Catalog
Yes	None	Yes	CATALOG*(catname)*
No	None	No	Master Catalog
No	Filename of User Catalog DLBL	No	User Catalog
No	'IJSYSCT'	No	Master Catalog
No	'IJSYSUC'	No	Master Catalog*
No	None	Yes	CATALOG*(catname)*

* VSAM defaults to the master catalog in this case because you specified the filename for the job catalog (in the DLBL CAT= parameter) but you did not specify a job catalog.

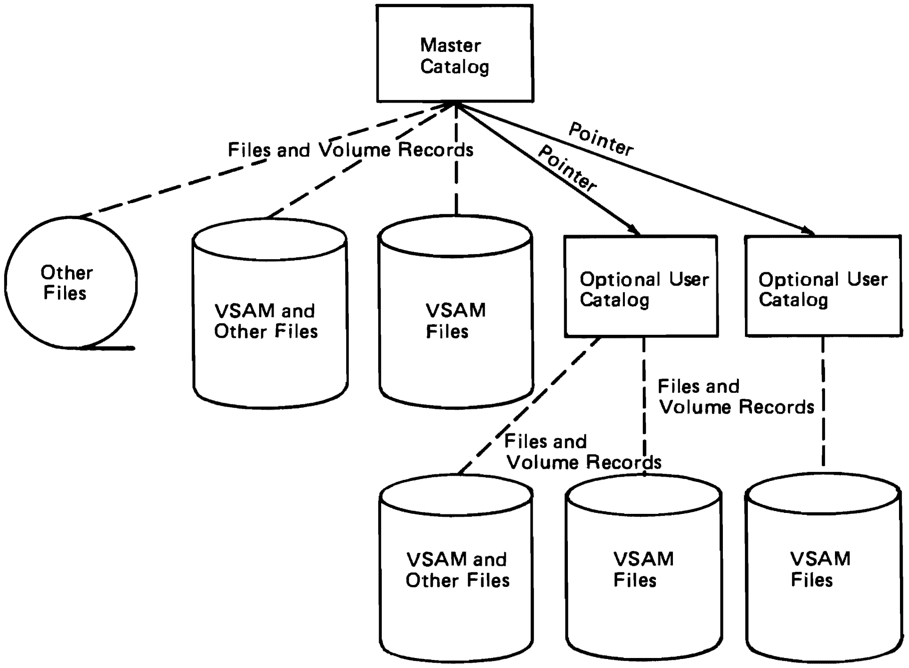


Figure 5-1. Catalog Relationships

A Catalog's Use in Data and Space Management

The information contained in VSAM catalogs is extensive enough to enable VSAM to suballocate and deallocate space for files on the available volumes without these volumes being mounted on a device of the system. (The only exceptions are volumes containing a unique file and recovery volumes. Both are subsequently discussed.) Consequently, the management of files is less dependent on job control information or information specified in processing programs.

You use Access Method Services to define VSAM data space(s) on a volume.

A VSAM data space is owned entirely by the VSAM catalog in which it is defined, and you use Access Method Services to suballocate one or more VSAM files from the data space. The space is identified in the volume's VTOC (volume table of contents). (You can also define nonVSAM files in a VSAM catalog defined without the RECOVERABLE attribute, but no nonVSAM files may be allocated within a VSAM data space.) When you define a VSAM *data space,* use DEFINE SPACE command parameters to actually allocate the space and to identify the volumes that will contain VSAM files. These volumes should be mounted.

All VSAM files of an installation must be cataloged in a VSAM catalog. You catalog a file by defining it with the Access Method Services command DEFINE CLUSTER. Access Method Services enters the file's name and other characteristics into the catalog.

When you define VSAM *files,* you normally do not need any DLBL and EXTENT statements, because VSAM automatically allocates space for them from existing data spaces. However, when you define a file with the UNIQUE attribute to enable the file to be allocated a space of its own, you do not define the data space beforehand, but provide extent information (via DLBL and EXTENT statements) in the Access Method Services job stream that defines that file. The data space is then set up at the same time as the entry for the file is created. The volume(s) must be mounted, as in defining a data space.

Except for unique files, VSAM can allocate and deallocate space for files on cataloged volumes that are not mounted. However, if there is not enough unused data space to contain a file, you must mount a volume or volumes to allocate new data space or assign to the file other volumes that contain unused data space. In addition, if the catalog is recoverable, the volume containing the catalog recovery area for the file being defined must be mounted.

VSAM Volume Ownership

The direct-access space occupied by VSAM files is recorded in a catalog. As a result a catalog controls (owns) not only the volume on which it resides, but any other volumes that contain part or all of any VSAM cluster, alternate index, or data space that is cataloged in the catalog.

VSAM's ownership and usage of space on a volume are indicated by label entries in the volume's VTOC. The "data secure file bit" in the format-1 VTOC (identifier) label of each VSAM data space on the volume is set to indicate both read and write protection. The "ownership bit" in the format-4 VTOC label is set to 1 if the volume is owned by a catalog (if the volume contains a VSAM data space or if the volume is a candidate volume for a VSAM object). The ownership bit indicates that the volume is owned by a catalog, but does not identify the owning catalog. Each catalog contains a volume entry for each volume it owns. The volume entry describes the direct-access volume's characteristics, each extent of the volume's VSAM data space, and each VSAM object that uses the volume's space. Volumes with duplicate volume serial numbers cannot be owned by the same catalog.

In order for you to remove the volume's ownership from a catalog, you must delete all VSAM objects and data spaces on the volume. When you are unable to use the DELETE command because Access Method Services can no longer access the volume (due to the damage that resulted from a system or hardware failure), you can reset the ownership bit by using the IKQVDU program. For more information on the IKQVDU program, see *VSE/VSAM VSAM Logic, Volume 1.*

When you list the volume's VTOC, when you reinitialize the volume, or when you dump the volume to a magnetic tape, you want to be able to recognize VSAM names in the volume's VTOC. The VTOC contains the VSAM-generated name of each VSAM data space on the volume and (for unique files) contains names for the data and index components of a cluster or alternate index (the format-1 VTOC label is identified with the object's entry name).

The names generated by VSAM have the following format:

- For a data space containing suballocated VSAM objects, the VSAM-generated name is Z999999n.VSAMDSPC.Taaaaaaa.Tbbbbbbb

  where:

  n=2 if no catalog resides in the data space
  n=4 if a user catalog resides in the data space
  n=6 if the master catalog resides in the data space
  aaaaaaabbbbbbb is the time stamp value

- For a unique data space (defined as a data space that cannot contain more than one cataloged VSAM object), the VSAM-generated name is

  VSAMDSET.DFDyyddd.Taaaaaaa.Tbbbbbbb

  where:

  yyddd is the date (year and Julian day)
  aaaaaaabbbbbbb is the time stamp value

When you define a VSAM file with the UNIQUE attribute, VSAM creates a unique data space. If you specify a name for the data and/or index component, VSAM places the name you specify in the format -1 VTOC label rather than generating a name.

To relate the VSAM-generated name with a VSAM cluster, alternate index, catalog, or data space, you list the catalog that owns the volume. You issue a LISTCAT command to list the catalog's contents. The LISTCAT output listing relates the VSAM-generated names with user-assigned entry names for cataloged objects.

Some of the implications of VSAM volume ownership are:

- All VSAM clusters, alternate indexes, and data spaces on a volume must be cataloged in the catalog that owns the volume. Only one catalog can own the volume.
- VSAM volume ownership does not affect nonVSAM files that reside on the volume. NonVSAM files can exist on a volume owned by a catalog but can be cataloged as nonVSAM entries in a catalog that doesn't own the volume. (NonVSAM files do not have to be defined in a VSAM catalog.)

- In order to release a volume from ownership by a catalog, you must delete all VSAM objects that reside on the volume. The catalog also contains a volume entry, which describes the volume and its VSAM data spaces. After deleting the VSAM objects, you must issue the DELETE SPACE command. The DELETE SPACE command deletes the VSAM data spaces on the volume, removes the volume entry from the catalog, deletes the format-1 label, and revises the format-4 label in the VTOC.

Each volume owned by a catalog contains a time stamp that is written in the VTOC when the volume is first cataloged. Both the time stamp in the VTOC and the time stamp in the volume entry in the catalog are updated whenever the catalog is updated in response to the following Access Method Services commands:

DEFINE SPACE
DEFINE CLUSTER (with UNIQUE attribute)
DEFINE ALTERNATEINDEX (with UNIQUE attribute)
DELETE SPACE
DELETE CLUSTER (with UNIQUE attribute)
DELETE ALTERNATEINDEX (with UNIQUE attribute)
EXPORT PERMANENT a cluster or alternate index with the UNIQUE attribute. (The cluster or alternate index is deleted.)
IMPORT(RA) a cluster or alternate index with the UNIQUE attribute. (Any old copy, if present, is deleted, and a new version is defined.)

If the volume's time stamp is earlier than the catalog's time stamp, the volume is considered down-level. Access Methods Services will not open a file on a down-level volume.

The following fields in the format-4 VTOC label time stamp are cleared only when VSAM volume ownership is relinquished:

Offset	Length	Description
77	8	VSAM timestamp 1 is set to the system's time of day when VSAM acquires volume ownership in a catalog. This timestamp is modified whenever physical space allocated to VSAM is acquired either by allocation of an extent or any time VSAM physical space is returned to the VTOC by VSAM catalog management routines.
85	3	VSAM indicators: **Byte 1** Bit 0 set to 1 = a VSAM catalog owns the volume. Bit 1 set to 1 = no significance for VSE. Bits 2-7 = reserved (set to binary zeros). **Bytes 2-3** = relative track or block location of the catalog recovery area on the volume. These bytes are set only if a recoverable catalog owns the volume; otherwise, the bytes are set to binary zeros.
88	8	VSAM time stamp 2 is the VSAM-only timestamp. Set only for OS/VS compatibility and not used by VSE.

Volume Mounting Requirements

Volumes must be mounted in the following cases:

- In all cases of a recoverable catalog where new catalog entries are made or old entries are modified (for example, ALTER, DEFINE, DELETE, EXPORT, IMPORT), the recovery volume must be mounted. In these cases the CRA of the recovery volume is opened and updated by VSAM.
- In all cases where files are actually accessed (for example, VSAM application programs,PRINT, REPRO, DELETE ERASE, workfiles, EXPORT, IMPORT), volumes must be mounted.

- In all cases where a VTOC update is necessary (for example, DEFINE or DELETE SPACE, DEFINE or DELETE a UNIQUE file, ALTER NEWNAME NONVSAM, ALTER NEWNAME UNIQUE), the volume(s) for the affected VTOC(s) must be mounted.
- The owning VSAM catalog must always be mounted.
- In the case where a path defined in a recoverable catalog is to be deleted, only its recovery volume must be mounted.

Information Contained in the Entries of a Catalog

The VSAM catalog is a key-sequenced file composed of a data part and an index part. The data of the catalog consists of entries describing files and of entries describing direct-access volumes in terms of the allocation of data spaces and the location of available space. The index of the catalog allows VSAM to find the file entry through its 44-byte name (file-ID) or the volume entry through the volume serial number.

File entries contain the information VSAM requires to properly access a file, verify access authorization, if required, and provide statistics on operations performed on a file.

Volume information in a catalog enables VSAM to keep track of data spaces and free storage areas.

Transporting Files between Systems

Since all VSAM files must be cataloged, moving a file from one system (or set of systems if in a DASD sharing environment) to another requires that catalog information be moved along with it or that the copy of the file being moved be cataloged in the receiving system. You can move individual files and user catalogs from one VSE system to another VSE system or to an OS/VS system by using the EXPORT and IMPORT commands. When you move a user catalog from one system (or set of systems) to another, its VSAM volume ownership moves along with it. Thus, a VSAM volume (or volumes) is portable between systems together with all VSAM data spaces and files contained in the volume(s).

The entire VSAM master catalog and the VSAM volumes owned by the master catalog can be moved from one VSE system (or set of systems) to another VSE system or to an OS/VS system. To use a VSAM master catalog from another VSE system or from an OS/VS system, you need only assign it by use of the DEF SYSCAT=cuu command during initial program load. All VSAM volumes owned by that catalog are then available to the receiving VSE system. In addition, a DLBL statement for the master catalog must be provided either in the job stream or in the label area.

Catalog and File Migration

This section provides information useful to a person migrating VSAM catalogs and files from one device type to another (for example, from CKD volumes to fixed block volumes). Note that a catalog on a fixed block device can own CKD volumes (and their VSAM files), and a CKD catalog can own FBA volumes.

Defining a Catalog

This section applies to both master and user catalogs.

VSAM defines a VSAM data space from which the catalog (and catalog recovery area) is suballocated. This is done on CKD devices using the

DEDICATE, ORIGIN, CYLINDERS, TRACKS, or RECORDS subparameters of DEFINE MASTERCATALOG|USERCATALOG. Fixed block devices require the same process, except that the DEDICATE, ORIGIN, BLOCKS or RECORDS subparameters must be specified. (CYLINDERS or TRACKS is not accepted.) Therefore, you must convert a CYLINDERS|TRACKS value to a BLOCKS or RECORDS quantity.

If you specified DEDICATE for the CKD device, no conversion is necessary.

Convert the number of tracks or cylinders into the number of bytes, using LISTCAT to determine the number of bytes per track and tracks per cylinder. Divide the number of bytes by 512 to determine the BLOCKS value. Adjust it accordingly, if you want more or less space allocated.

The beginning-block-number specification in the ORIGIN parameter depends on where you want the data space to be on the volume. Use the LVTOC utility program to determine what space is available on the volume. The catalog will be located at the beginning of the defined data space, and the catalog recovery area (if applicable) will immediately follow it. The CRA size is always equal to the max-CA value for the particular fixed block device type. (See the LISTCAT volume entry for special fields BLKS/MAX-CA described in "Using VSE/VSAM Commands and Macros," Appendix B.)

You may wish to change other subparameters of MCAT|UCAT (for instance, the volume serial number), but there are no special considerations for fixed block devices.

Specify the actual space to be suballocated for your catalog using the BLOCKS or RECORDS subparameters of DATA and INDEX. Do *not* try to directly convert a CKD catalog size definition to a fixed block definition. Instead, calculate the desired values using the instructions in "Defining a Catalog" in this chapter. To avoid an overly small, inefficient control area size, make the secondary allocation value at least as large as the desired control area size.

Defining a Data Space

The considerations for data space definition are essentially the same as for catalog definition. Differences are:

- A catalog is not suballocated from the data space.
- A CRA is suballocated only if this is the first space defined on a volume owned by a recoverable catalog.

If CANDIDATE is specified with DEFINE SPACE, fixed block data space definition is the same as CKD data space definition.

Defining a Non-Unique Cluster or Alternate Index

Because these files (or their components) are suballocated from VSAM data spaces, there are no job control considerations for fixed block devices. For FBA devices, you must convert the TRACKS or CYLINDERS subparameters to BLOCKS or RECORDS. (The RECORDS subparameter does not require conversion.) This conversion is the same as described above for catalog conversion.

Defining a Unique Cluster or Alternate Index

If a cluster or alternate index contains both a non-unique component and a unique component, conversion considerations for the non-unique component are as described above.

For each unique component (data and, if present, index) you must convert EXTENT statement parameters and the TRACKS|CYLINDERS subparameters. Both conversions are required because a unique component occupies its own

VSAM data space.If the component is to be on more than one volume, specify a new EXTENT statement for each volume.

Migrating a Catalog to Another Device

For information about converting an imbedded catalog to a non-imbedded catalog, or converting a catalog to a fixed-head data space, refer to "Chapter 10: Data Interchange Considerations."

Moving a Master Catalog to Another Volume

1. Using EXPORT or EXPORTRA, create portable copies of all files and user catalog entries (EXPORTRA only) that are to be in the new catalog (procedure described below). For EXPORT, DISCONNECT any user catalogs to be reconnected to the new catalog.
2. IPL with the master catalog assigned to the new volume, using the IPL DEF SYSCAT=cuu command.
3. Define the new master catalog (procedure described above).
4. Define any VSAM data spaces required for the volumes. (For old VSAM volumes, any files and data spaces that belonged to another catalog must have already been deleted.) Note that the define catalog operation has already defined a data space on the catalog volume. Any space to be occupied by unique files should be left unallocated.
5. Using IMPORT or IMPORTRA, copy VSAM files and user catalog entries (IMPORTRA only) to volumes belonging to the new catalog. (The next section describes new device type considerations.) If IMPORT was used, you can IMPORT CONNECT user catalogs.

Moving a User Catalog to Another Volume

1. Using EXPORT or EXPORTRA, create portable copies of all files that are to be in the new catalog (procedure described below).
2. Delete or disconnect the previous user catalog entry unless it is owned by a different master catalog.
3. Define the new user catalog (procedure described above).
4. Define any VSAM data spaces required for the volumes. (For old VSAM volumes, any files and data spaces that belonged to another catalog must have already been deleted.) Note that the define catalog operation has already defined a data space on the catalog volume. Any space to be occupied by unique files should be left unallocated.
5. Using IMPORT or IMPORTRA, copy VSAM files to volumes belonging to the new catalog. (The next section describes new device type considerations.)

Migrating VSAM Files to Another Device

There are three ways to move VSAM files from one volume to another. They may or may not require moving from one catalog to another.

DEFINE/REPRO: To move files between two volumes owned by different catalogs, DEFINE each file on the new volume, using its old name. REPRO each file onto the new volume, and delete it from the old one.

To move files between two volumes owned by the same catalog, DEFINE each file on the new volume, using a temporary name that is not already in the catalog. REPRO each file onto the new volume, and delete it from the old volume. Rename the new copy, using the name the file had on the old volume.

In both cases, alternate indexes can be copied. You must redefine all paths for the new copy.

EXPORT/IMPORT: With EXPORT/IMPORT, each file to be migrated is first exported to a temporary SAM file (tape or DASD). For EXPORT PERMANENT, this frees space (and volumes if all files on them are exported) that is potentially reusable during the IMPORT phase.

To assure the desired space allocation, DEFINE the files before importing them. If files are imported but not defined, too much or too little space may be allocated to them. Then IMPORT the files.

Unique files require extent values specified on an EXTENT statement. Path definitions are implicitly transferred.

EXPORTRA/IMPORTRA (recoverable catalogs only): EXPORTRA/IMPORTRA is similar to EXPORT/IMPORT, the key difference being that files cannot be defined before IMPORTRA. This can produce significant problems when changing device types because the CKD device allocations (tracks or cylinders) will be used on the new fixed block device, with a track mapping to a minimum control area unit, and a cylinder mapping to a maximum control area unit. This may result in too much or too little space being allocated.

EXPORTRA/IMPORTRA does allow multiple files to be handled on a single command execution. Path definitions are implicitly transferred.

NONVSAM Migration

Catalog entries can be moved into catalogs on fixed block devices (as described above) via DEFINE NONVSAM and DELETE, but they cannot have fixed block specified as their device type.

Space Allocation via Modeling

If a user catalog, cluster, or alternate index being migrated from one device type to another had its space allocation defined by modeling, you should consider changing to explicit specification, or modeling it on a catalog, cluster, or alternate index on the new device type. Otherwise, you will allocate space based on the track and/or cylinder capacity of the CKD device type rather than the fixed block device type. This can cause wasted space, excessive secondary allocation, and inefficient or even invalid control area or control interval sizes.

For further information about modeling, refer to Chapter 6.

Chapter 6: How to Use One Object as Model for Another Object and Override System Defaults

You can use the entry of an already-defined alternate index, catalog, cluster, or path as a model for the definition of another object of the same type. When one entry is used as a model for another, its attributes are copied as the new entry is defined.

Modeling permits you to set your own parameter defaults to override system defaults. Once defaults are established, you need not respecify them each time new objects are defined. An explicit parameter specification, however, overrides defaults established by you (through modeling) and by the system.

The normal Access Method Services DEFINE CLUSTER or AIX procedure is greatly simplified by reducing the numbers of parameters required. This in turn can reduce the number of errors that are likely to occur and the number of parameters to which the average user needs to be exposed. At the same time, it permits application- and installation-associated standards.

There are three kinds of models, two of them *explicit*, in that you must specify the name of the model you wish to use. For the third kind, an *implicit* model, VSAM chooses a default model based on the kind of object you are trying to define.

Explicit Allocation Models

Example A:

Establishing the Model **Using the Model**

```
DEFINE CLUSTER                      DEFINE CLUSTER
    NAME (entryname)                    •
    •                                   •
    •                                   MODEL (entryname)
```

Example A shows a conventional (pre-VSE/VSAM Release 2) model that occupies data space and can be used as a normal VSAM object. You must explicitly specify the *entryname* subparameter of the MODEL parameter to identify the object to be used as a model. This is the only form of modeling valid for paths and user catalogs. If MODEL is specified as a parameter of PATH:

1. The attributes of the model are used for the path being defined.
2. Any attributes explicitly specified as parameters of the defined path are defined and override those of the model.

Figure 6-1 shows MODEL specified as a parameter of CLUSTER (that is, at the cluster level) but not specified as a subparameter of the DATA or INDEX parameter (*this is the usual case*). This figure illustrates how parameters are merged from a model (cluster X) and a DEFINE CLUSTER command. The result is a new list of cluster parameters, which VSAM uses to create a cluster.

1. The non-propagating cluster level attributes (*entryname,* passwords, AUTHORIZATION, ATTEMPTS, CODE, OWNER, TO, FOR, and allocation attributes) of the model are used for the user catalog, cluster or alternate index being defined.
2. Any non-propagating cluster level attributes *explicitly specified* as parameters of the defined object are applied to and override those of the model.

3. The attributes of the model are used for the data and index components of the alternate index, cluster, or user catalog.

4. Attributes explicitly specified at the cluster level are propagated to the cluster's data and index components.

5. Attributes explicitly specified for the object's data and index components (that is, specified as subparameters of the DATA or INDEX parameter) are defined.

Attributes specified for each step override the attributes specified by the previous step.

If MODEL is specified as a subparameter of the DATA or INDEX parameter (not applicable to a user catalog):

1. Attributes explicitly specified at the cluster level are propagated to the object's data and index components.

2. Attributes of the model specified for the data or index component are defined (that is, the model specified with the MODEL subparameter of the DATA or INDEX parameter).

3. Attributes explicitly specified for the data and index components are defined (that is, the attributes specified with subparameters of the DATA or INDEX parameter).

Attributes specified for each step override the attributes specified by the previous step.

The MODEL parameter is designed to let you easily define files that are identical except for their names and security attributes. When you use the MODEL parameter, you should take care to ensure that your job is not terminated because of allocation problems when you explicitly do any of the following:

- Specify a different type of device with the VOLUMES parameter.

- Change the length or position of the keys with the KEYS parameter.

- Change the size of records, buffer space, or control intervals with the RECORDSIZE, BUFFERSPACE, or CONTROLINTERVALSIZE parameters.

- Change the type of cluster (that is, entry-sequenced, key-sequenced, or relative-record), the type of alternate index (that is, key-pointer or RBA-pointer), or the allocation-type of the object (that is, unique or nonunique).

- Change the *unit of allocation* with the BLOCKS, TRACKS, CYLINDERS, or RECORDS parameters.

When you explicitly specify any of the above parameters for your to-be-defined object, you might have to make corresponding changes to other related parameters.

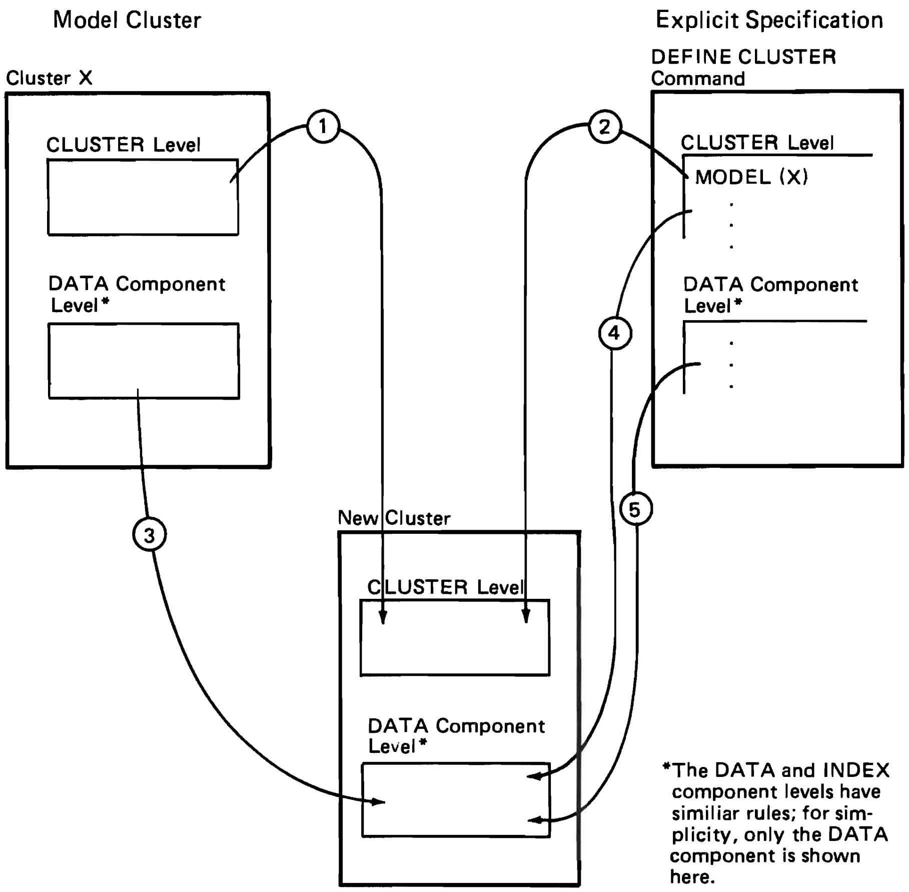


Figure 6-1. Specifying the MODEL Parameter at the CLUSTER Level Only

Noallocation Models

Using *explicit noallocation* and *default* models, the defined object exists *only* as a model; no space is suballocated to it. The model is represented by entries in the VSAM catalog.

Explicit Noallocation Models

Example B:

```
Establishing the Model              Using the Model

  DEFINE CLUSTER                      DEFINE CLUSTER
     NAME (entryname)                    •
     •                                   •
     •                                   MODEL (entryname)
     NOALLOCATION
```

Example B is an *explicit* model because you must specify MODEL *(entryname)* for the cluster you wish to use as a model. It is a *noallocation* model because no storage is allocated to it.

Default Models

The third kind of model is a default model. It is an *implicit* model because you do not have to specify the name of the model in order to reference it. It is a *noallocation* model because no storage is suballocated to it. When you define the model, specify the *entryname* subparameter of the NAME parameter as one of the following:

DEFAULT.MODEL.KSDS	(key-sequenced file)
DEFAULT.MODEL.ESDS	(VSAM entry-sequenced file)
DEFAULT.MODEL.ESDS.SAM	(managed-SAM file)
DEFAULT.MODEL.RRDS	(relative-record file)
DEFAULT.MODEL.AIX	(alternate index)

Each catalog may have five implicit models, one of each type. As shown below, you need only specify INDEXED, NONINDEXED, RECORDFORMAT(...)NONINDEXED, NUMBERED, or AIX for VSAM to locate the appropriate model.

Example C:

Establishing the Model **Using the Model**

```
DEFINE CLUSTER                          DEFINE CLUSTER
   NAME(DEFAULT.MODEL.KSDS)                NAME (entryname)
   •                                       •
   •                                       •
   NOALLOCATION                            INDEXED

DEFINE CLUSTER                          DEFINE CLUSTER
   NAME(DEFAULT.MODEL.ESDS)                NAME (entryname)
   •                                       •
   •                                       •
   NOALLOCATION                            NONINDEXED

DEFINE CLUSTER                          DEFINE CLUSTER
   NAME(DEFAULT.MODEL.ESDS.SAM)            NAME (entryname)
   •                                       •
   •                                       RECORDFORMAT(...)
   NOALLOCATION                            NONINDEXED

DEFINE CLUSTER                          DEFINE CLUSTER
   NAME(DEFAULT.MODEL.RRDS)                NAME (entryname)
   •                                       •
   •                                       •
   NOALLOCATION                            NUMBERED

DEFINE ALTERNATEINDEX                   DEFINE ALTERNATEINDEX
   NAME(DEFAULT.MODEL.AIX)                 NAME (entryname)
   •                                       •
   •                                       •
   NOALLOCATION
```

How VSAM Determines which Parameters to Use

VSAM goes through the following sequence in determining which parameter to use in the definition of a cluster or alternate index.

1. Did you explicitly specify a parameter in the define? If yes, VSAM uses it. (If you explicitly specify a space allocation parameter (CYLINDERS, TRACKS, BLOCKS, or RECORDS) at *any* level of DEFINE CLUSTER/AIX, the space allocation parameter(s) in your model are ignored.)

2. Did you specify MODEL parameter in the define (Example A above)? If yes, go to step 4; VSAM creates a file using the parameters specified in MODEL *(entryname)*.

3. Did you specify the NOALLOCATION parameter with a DEFAULT.MODEL.xxxx in a previous DEFINE command, thereby creating a default model (Example C above)? If yes and the file organization matches the entryname, VSAM uses the parameters specified in the default model.
4. If none of the above apply, VSAM uses the system default (if one exists).

Restrictions

The following restrictions exist for modeling of VSAM objects.

- If you specify DEFINE CLUSTER or DEFINE ALTERNATEINDEX and the cluster name begins with DEFAULT.MODEL., VSAM assumes that you are establishing a model. The rest of the name must be KSDS, ESDS, ESDS.SAM, RRDS, or AIX. It is not possible to open a file or component whose name begins with DEFAULT.MODEL.. DEFINE CLUSTER/AIX ignores user-specified DATA and INDEX component names for clusters that have the DEFAULT.MODEL. prefix. Instead these components are implicitly assigned a name constructed from the cluster or alternate index name with the additional qualifier of DATA or INDEX. A message will tell you any data/index names that have been generated in this way.
- If space parameters (CYLINDERS, TRACKS, RECORDS, or BLOCKS) are specified at any level of DEFINE CLUSTER/AIX, they override any modeled defaults.
- To model USECLASS for a cluster or alternate index, do one of the following:
  - Specify space parameters (CYL, TRK, REC, BLK) at levels corresponding to where modeling is to apply (cluster, data, index).
  - Do not specify space parameters at any level. (Both USECLASS and space specifications are modeled.)
- You cannot rename (via ALTER NEWNAME or IMPORT NEWNAME) any catalog entry such that the name is being changed to or from DEFAULT.MODEL.xxxx. An attempt to do so causes the command to terminate with an error message.

Figure 6-2 lists the various DEFINE parameters and indicates whether each can be modeled explicitly (MODEL*(entryname)* specification) and implicitly (DEFAULT.MODEL.xxxx).

Parameter	Modeling		System Default if Parameter Not Modeled	Notes
	Explicit	Implicit		
ATTEMPTS	Yes	Yes	Yes	Not propagated to other levels.
AUTHORIZATION	Yes	Yes*	No	Not propagated to other levels.
BLOCKS	Can model only if not explicitly specified at any level.		No	Propagated via algorithm from cluster or data levels.
BUFFERSPACE	Yes	No	Yes	
CLASS	No	No	Yes	Refer to USECLASS parameter.
CODE	Yes	Yes*	No	Not propagated to other levels.
CONTROLINTERVALSIZE	Yes	No	Yes	
CYLINDERS	Can model only if not explicitly specified at any level.		No	Propagated via algorithm from cluster or data levels.
DEDICATE	No	No	No	
DEFAULTVOLUMES	No	No	Yes	
ERASE	Yes	Yes	Yes	Propagated to data level only; NOERASE is the default.
EXCEPTIONEXIT	Yes	Yes	No	
FILE	No	No	No	
FOR	Yes	Yes	Yes	Specified at cluster level only; propagated to data or index.
FREESPACE	Yes	Yes	Yes	(0 0) is the default.
IMBED	Yes	Yes	Yes	NOIMBED is the default.
INDEXED	See note	No	Yes	KSDS is created if nothing or INDEXED is specified.
KEYRANGES	Yes	Yes	No	
KEYS (AIX)	Yes	Yes	Yes	
KEYS (cluster)	Yes	Yes	Yes	Not specified or modeled for INDEX.
NOALLOCATION	Yes	No	Yes	SUBALLOCATION is the default.
NOERASE	Yes	Yes	Yes	NOERASE is the default.
NOIMBED	Yes	Yes	Yes	NOIMBED is the default.
NONINDEXED	See note	No	Yes	Refer to INDEXED parameter.
NONSPANNED	Yes	Yes	Yes	NONSPANNED is the default.
NONUNIQUEKEY	Yes	Yes	Yes	NONUNIQUEKEY is the default.
NOREPLICATE	Yes	Yes	Yes	NOREPLICATE is the default.
NOREUSE	Yes	Yes	Yes	NOREUSE is the default.
NOTRECOVERABLE (cat)	Yes	No	Yes	
NOUPGRADE	Yes	Yes	Yes	UPGRADE is the default.
NOWRITECHECK	Yes	Yes	Yes	NOWRITECHECK is the default.
NUMBERED	See note	No	Yes	Refer to INDEXED parameter.
ORDERED	Yes	Yes	Yes	UNORDERED is the default.
OWNER	Yes	Yes	No	Not propagated to other levels.
Passwords	Yes	No	No	No propagation from cluster level, but lower level password is propagated to master if no master password is specified.

* To implicitly model this parameter, the object must be defined with at least one password, and the master catalog password must be specified in the CATALOG parameter.

Figure 6-2. Modeling of DEFINE Parameters (Part 1 of 2)

Parameter	Modeling		System Default if Parameter Not Modeled	Notes
	Explicit	Implicit		
RECORDFORMAT	Yes	NA	Yes	For SAM ESDS models only.
RECORDS	Can model only if not explicitly specified at any level.		No	Propagated via algorithm from cluster or data levels.
RECORDSIZE	Yes	No	Yes	
RECOVERABLE (cat)	Yes	No	Yes	
RECOVERY	Yes	Yes	Yes	
RELATE	No	No	NA	
REPLICATE	Yes	Yes	Yes	NOREPLICATE is the default.
REUSE	Yes	Yes	Yes	NOREUSE is the default.
SHAREOPTIONS	Yes	Yes	Yes	
SPANNED	Yes	Yes	Yes	NONSPANNED is the default.
SPEED	Yes	Yes	Yes	
SUBALLOCATION	Yes	No	Yes	SUBALLOCATION is the default.
TO	Yes	Yes	Yes	Specified at cluster level only; propagated to data and index.
TRACKS	Can model only if not explicitly specified at any level.		No	Propagated via algorithm from cluster or data levels.
UNIQUE	Yes	No	Yes	SUBALLOCATION is the default.
UNIQUEKEY	Yes	Yes	Yes	NONUNIQUEKEY is the default.
UNORDERED	Yes	Yes	Yes	UNORDERED is the default.
UPGRADE	Yes	Yes	Yes	UPGRADE is the default.
USECLASS	Only if space parms are specified at same level, or if no space parms are specified.		Yes	
VOLUMES	Yes	Yes	No	
WRITECHECK	Yes	Yes	Yes	NOWRITECHECK is the default.

Figure 6-2. Modeling of DEFINE Parameters (Part 2 of 2)

Default Volumes

Default volume lists are derived from the volumes list of a default model that is of the same type as the object being defined. For example, if a VSAM ESDS cluster is defined without a VOLUMES parameter, an ESDS default model (DEFAULT.MODEL.ESDS.DATA) is used to build the volumes list for the ESDS. Because volume selection from the default volume list is done randomly for each component, the data and index components of a KSDS or AIX could reside on different volumes and even different device types. You can eliminate the possiblilty of different device types by including devices of only one type when defining the KSDS or AIX model.

When a file is defined implicitly (via managed-SAM) and if you have not provided volume information in an EXTENT statement, VSAM attempts to construct a volumes list from a managed-SAM ESDS default model (DEFAULT.MODEL.ESDS.SAM). No other information is used (from the SAM ESDS default model) for an implicit define.

DEFAULTVOLUMES forces a default model to override an explicit model for purposes of determining the volumes list. There are three sources of volumes lists:

1. Explicit specification (VOLUMES parameter)

2. Explicit model (MODEL parameter)
3. Default model (DEFAULT.MODEL.xxxx plus VOLUMES parameter)

These sources are listed in order of precedence. 1 overrides 2, and 3 takes effect if 1 and 2 are missing. If only 2 and 3 are present, however, specifying DEFAULTVOLUMES causes the volumes list in 2 to be bypassed in favor of the volumes list in 3. You cannot specify the DEFAULTVOLUMES parameter to bypass 2 if 3 does not exist. (*At least one of these options (1,2, or 3) must be specified or modeled.*)

DEFAULTVOLUMES cannot be explicitly modeled because it is not retained as an attribute in the catalog. Do not try to use default volume lists with KEYRANGES because VSE/VSAM does not order the volumes in any way when allocating space to them.

Chapter 7: VSAM Labels

VSAM maintains identifying information for its files in a central location that is separate from the internal DASD labels that describe files of other access methods. This location is a DASD file called the VSAM catalog. Volumes that contain VSAM files have the same internal labels as other volumes. Most of the identifying information for VSAM files, however, is in the VSAM catalog.

VSAM volumes use the volume label and the format-1, format-3, and format-4 labels. A brief description of each of these labels follows. They are described in greater detail later in this chapter. VSAM does not support user-standard labels.

- The *volume label* (VOL1) is generally written when the disk pack is received at an installation. At that time a permanent volume serial number (volser) is assigned to the volume. The volser is written on the volume as part of the label to provide permanent identification of the volume.
- The VSAM *format-1 label* describes direct access space; the characteristics of the logical files that occupy that space are described in the VSAM catalog. There is a format-1 label for each VSAM data space on the volume. Each data space consists of one or more separate extents. Up to three extents are described in the format-1 label, and additional extents are described in a format-3 label (pointed to by the format-1 label).

You do not usually name a VSAM data space; the 1 - 44-byte key area contains a name assigned by VSAM. If you allocate a data space to contain the data or the index of only one specific VSAM file (called a unique file), the 44-byte key area will contain the name given to the data or the index when you define it.

When a new VSAM data space is created (Access Method Services DEFINE command), existing format-1 labels are read and checked, and new labels are created by the catalog and space management routines.

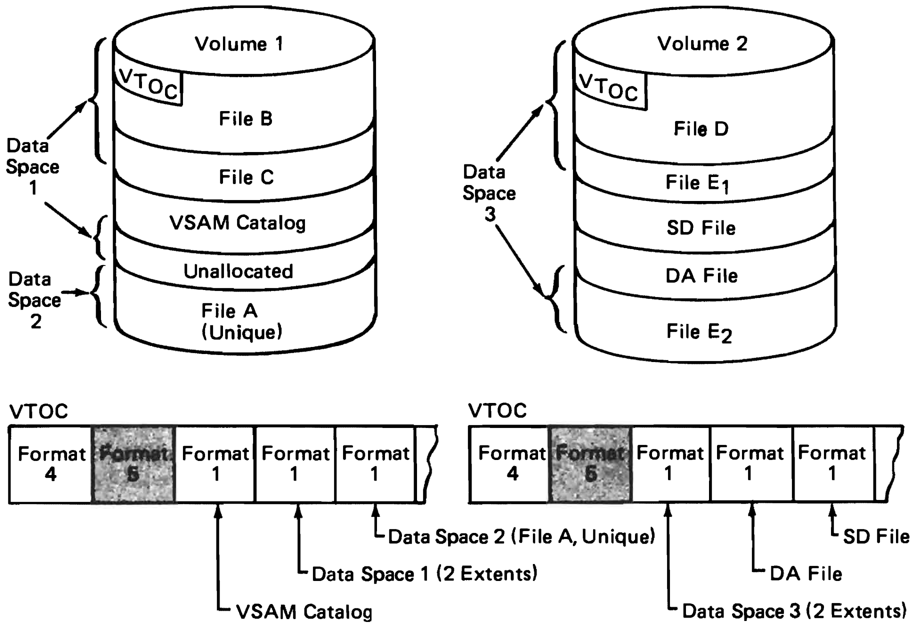
- A *format-3 label* is written whenever a VSAM data space occupies more than three separate areas (extents) of a volume. It is used to supply the limits (starting and ending addresses) of the additional extents. Thirteen separate extents can be defined by one format-3 label. This label is pointed to by the format-1 label.

When a new VSAM data space is created (via the Access Method Services DEFINE command), existing format-3 labels are read and checked, and new labels are created by the catalog and space management routines.

- A *format-4 label* defines the Volume Table of Contents (VTOC) (the area where the format-1 and format-3 labels are stored) and identifies the volume as a VSAM volume if it contains VSAM spaces. The format-4 label is always the first record in the VTOC. It is written when you initialize your disk pack by using the Initialize Disk program. Open/close routines refer to this label to determine the extent of the VTOC.

For VSAM, the VTOC is essentially a directory of all direct access space owned by VSAM on the volume. The catalog is the directory to all VSAM files on the volume.

Volume Layouts



Notes:

1. This figure shows two volumes that contain VSAM data spaces and the contents of the VTOC for each volume. The VTOC describes data spaces owned by VSAM; files are described in the VSAM catalog.
2. Data Space 2 is occupied by File A, which is Unique. No other file can be suballocated space in Data Space 2 and File A cannot be extended to any other data space.
3. The 44-byte name field of the label for Data Space 2 contains the name (file-ID) of File A. The 44-byte name fields of the labels of the other data spaces contain a VSAM-generated data space name.
4. VSE does not use the format-5 VTOC label, but space is reserved for it for OS/VS compatibility.

Figure 7-1. Volume Layout of VSAM Files

Label Information Area

VSE/VSAM file label information, as well as standard labels for a user catalog, can be submitted under both //OPTION STDLABEL and //OPTION PARSTD. VSAM searches the partition temporary user label area (USRLABEL), the partition standard label area (PARSTD), and the standard label area (STDLABEL), in that order. Thus, it is possible to override permanent label sets for a single job by submitting the new label set under //OPTION USRLABEL. //OPTION USRLABEL is the default and can be omitted.

Refer to *VSE/Advanced Functions DASD Labels* for the layout of the label information area.

VTOC Label Processing

General

VSAM Data Spaces: The format-1 and format-3 labels describe VSAM data spaces. A data space consists of one or more extents on a single volume allocated to VSAM and controlled by a VSAM catalog. VSAM files are written in data spaces.

Even if it does not contain any files, a data space is owned by VSAM and is not available for files of other access methods.

Label processing is done when data spaces (including catalogs and uniques) are created or deleted during RESETCAT processing, and during ALTER NEWNAME for UNIQUE.

The format-1 and format-3 labels are created and checked (for overlap or duplicate name) only when data spaces are created (including data spaces for unique files). If data spaces are deleted, their format-1 and format-3 labels are removed from the VTOC. Labels are also altered during RESETCAT processing if the data in the label and the catalog do not agree. When VSAM files are processed, the *file-ID* must be supplied through the DLBL statement, but the VSAM catalog is used for checking the location and characteristics of the files.

VSAM Files: VTOC label processing takes place only for UNIQUE VSAM files that are being defined, deleted, or renamed.

VSAM files are normally defined after data spaces have been defined. The direct access space for the files is suballocated by VSAM from one or more data spaces. You may select the volume or volumes the file will reside on. You tell VSAM how much space to suballocate to the file initially and, optionally, how much additional space to suballocate when the file must be extended. VSAM decides which data spaces or portions of data spaces to suballocate to a file.

You can, however, specify the size and exact location of the file when you define it. In this case, the file is called unique and occupies its own data space which is defined as the file is defined. No other files can occupy that data space. If the file extends across more than one volume, it occupies one data space on each volume. The format-1 and format-3 labels still describe the data space(s) occupied by the unique file. A key-sequenced unique file requires separate data spaces for the data and the index components.

The *file-ID* parameter of the DLBL statement indicates the file you want to process. It is the same as the name of the file, stored in the catalog, which was specified in the NAME (*entryname*) parameter of the DEFINE statement. For VSAM data spaces, the format-1 label contains a VSAM-generated data space name.

VTOC Labels for FBA Devices: The physical block is the basic unit of storage on an FBA device. A DASD address is a physical block number relative to the beginning of the volume.

An FBA VTOC is divided into control intervals of the VSAM relative record format; the VTOC labels reside in these control intervals. There is a slot for the VTOC label and its corresponding RDF in the control interval. The control interval size is a multiple of FBA physical blocksize; a control interval always starts on a block boundary. Specify VTOC size via the Initialize Disk Utility program.

The VOL1 label contains the VTOC control interval size, the number of blocks per CI, and the number of labels per CI. VTOC labels are referenced according to relative record number (beginning with 1).

For information about individual fields in the VOL1 and VTOC labels, refer to *VSE/Advanced Functions DASD Labels.*

VSAM Data Space

VOL1 Label Processing

The VSAM VOL1 label fields are the same as for the other access methods.

The standard volume label (VOL1) must be on cylinder 0, track 0, record 3 (CKD) or in physical block 1, called the volume label block (FBA). If it is not, the job is canceled.

The VOL1 label, written by the IBM-supplied Initialize Disk utility program, contains a permanent volume serial number (volser).

VSAM determines the location of the VTOC from the VOL1 label.

If any additional volume labels follow the VOL1 label, VSAM ignores them.

Format-1 Label Processing for UNIQUE Files

You must supply one DLBL statement when creating a unique file and one EXTENT statement for each separate extent on the volume that the data space will occupy. A multivolume unique file requires only one DLBL statement, even though it occupies a data space on each volume.

DLBL Statement: The DLBL statement for defining a data space under VSAM requires only the *filename* parameter and the VSAM code. DLBL *filename* is identical to the *dname* specified via the FILE parameter of the DEFINE command.

The *file-ID* parameter is not required and is ignored if you specify it. The *date* parameter can be specified but it has no real function; VSAM data spaces and files can be deleted only by using the DELETE command of Access Method Services.

EXTENT Statement: An EXTENT statement defines a continuous extent of the volume that is to be allocated to VSAM. There can be up to 16 extents in a data space, and a data space is contained entirely on one volume. The EXTENT statement provides the starting address (relative address) and the number of tracks (CKD) or blocks (FBA), which indirectly give the ending address. The EXTENT statement also provides the order in which this extent should be processed in a multi-extent unique file.

VSAM validates the EXTENT specifications by checking the extent limits against the limits of the format-4 label and each format-1 and format-3 label already written in the VTOC. If the new extent overlaps an existing extent, VSAM issues a message to the operator. If the overlapped extent is part of a file of another access method (expired or unexpired), the operator can delete the file or terminate the job. If the overlapped extent is part of a VSAM data space (or unique file), the operator can only cancel the job. VSAM data spaces or files (expired or unexpired) can only be deleted through the Access Method Services DELETE command.

If all extents of the new unique file are valid, VSAM writes one (or two, for a KSDS) format-1 label, and (if necessary) the format-3 label into an available location in the VTOC.

For the data or the index of a unique file, you may specify a data space name in the DEFINE command. If specified, this name is entered in the catalog and in the label. Remember that even though the name of a unique file is entered in the labels of the data space it occupies, the information describing the file is in the catalog.

Bytes 45-60, 63-75, 83-84, and 94 are written in the format-1 label for VSAM. This information is for compatibility with the format-1 labels of other access methods; during processing, VSAM uses the catalog, rather than using information from the VTOC.

Bytes 106-115 define the first (or only) extent allocated to the unique file component. If there is more than one extent, bytes 116-125 define the second extent, and bytes 126-135 define the third extent. These fields are written from the EXTENT statements you supply.

If you have included more than three EXTENT statements, VSAM writes a format-3 label and writes the address of that label in the pointer field (bytes 136-140) of the format-1 label.

If the unique file is deleted, the format-1 label (and if present, the format-3 label) is removed from the VTOC.

Format-3 Label Processing

The VSAM format-3 label fields are the same as for the other access methods, but a VSAM data space can have only one format-3 label.

If more than three extents are required for the data space (or unique file), VSAM sets up a format-3 label for the additional extents. A data space can consist of up to 16 extents, so only one format-3 label is allowed. VSAM processes the extent fields of the format-3 labels in the same manner as those of the format-1 label.

If the data space is deleted, the format-3 label is removed from the VTOC, along with the format-1 label.

Format-4 Label Processing

The format-4 label describes the VTOC, not the files or data spaces of individual access methods. However, a VSAM indicator field (bytes 77-87) is written in the format-4 label of any volume that contains VSAM files or data spaces. This field (volume timestamp) indicates the date and time the most recent VSAM data space was added to or deleted from the volume. For OS/VS compatibility reasons, this timestamp is repeated in bytes 88 - 95.

The same date and time are entered in the catalog. VSAM open routines check to see if the volume timestamp matches the timestamp for it in the catalog. If they do not match, processing continues, but an error code is issued to indicate that the VTOC might not agree with the data space information in the volume's catalog entry.

Bit 0 of byte 85 indicates that this volume is owned by a VSAM catalog. The ownership is established normally by defining the first data space on the volume, but it can also be established by using the CANDIDATE parameter of the Access Method Services DEFINE SPACE command.

If the volume belongs to a recoverable catalog, bytes 86 - 87 indicate where the catalog recovery area is.

If all VSAM data space is deleted from a volume, the VSAM indicator field (bytes 77-87) is erased. The volume can then be used by another VSAM catalog. The deleted space can also be used by other VSE access methods.

VSAM Files

Defining a File: Suballocation

When a non-unique file is defined, the space for it can be suballocated from one or more existing data spaces on one or more volumes. This is illustrated in Figure 7-3. VTOC label processing is not performed for the following reasons:

- Information needed to set up the file is in the DEFINE command.
- Information about data spaces to be suballocated to the file is in the VSAM catalog.

The resulting description of the file is entered in the catalog. DLBL and EXTENT statements are not required; they are ignored if specified for a non-recoverable catalog. The volume containing the catalog must be mounted, but the volumes on which the file is defined need not be mounted. Additional information about volume mounting requirements appears in Chapter 5.

If the catalog in which the file is defined is recoverable, the recovery volume must be mounted for the DEFINE operation (see Figure 7-4).

You indicate the volume(s) on which the file will reside, the amount of space to be initially suballocated to the file and, optionally, the amount of space to be suballocated if the file must be extended. VSAM selects the extent(s) on the volume on which to write the file. If you specify more volumes than necessary for the primary space, the additional volumes can be used when the file is extended, if they contain free data space. If none of the volumes contains free data space, new data spaces must be defined, or volumes with free data space can be made available to the file through the ALTER command of Access Method Services. You can indicate in which order the volumes should be used. You can also decide to place certain portions of the file (key ranges) on certain volumes. If the file must be extended, VSAM can use only the volumes you indicated. If there is no free space on those volumes, you will have to define more data space or make other volumes available to VSAM through the ALTER command. Additional information appears in the "Multiple Volume Support" section in Chapter 9 of this manual.

Loading a file is a separate step from defining it. Records can be loaded into a file by a VSAM processing program, using the PUT macro, or by the REPRO command of Access Method Services.

Defining a File: Unique

A unique file, like those of other access methods, occupies space described in the VTOC through DLBL and EXTENT statements. The data space for a unique file is defined (implicitly) in the same DEFINE command as the file itself.

Characteristics of the file, such as logical record length, are specified in the DEFINE command, just as with a suballocated file. Space and volume information is taken from DLBL and EXTENT statements instead of from the DEFINE command.

The data and index of a unique key-sequenced file or alternate index require separate data spaces, and hence, separate DLBL and EXTENT statements.

Label processing is performed for the data spaces of a unique file as described in the previous section "VSAM Data Space." The only difference is that the 44-byte names of the data and index are placed in the labels as well as in the file's catalog entry. The data spaces of unique files are described in the VSAM catalog as well as in the VTOC.

A unique file cannot be extended. The extents of the file are the same as the extents of the data spaces and, since they are described in the VTOC, cannot be changed without deleting the file. See Figure 7-5 for defining a unique file.

Processing a File

When a previously defined file is processed by a VSAM application program or by a PRINT or REPRO command, a DLBL statement is required for the file. It is retrieved by VSAM open from the label area. Open obtains the DLBL statement from the filename given in the Access Method Control Block (ACB) in the processing program. All the information required to process the file is in the VSAM catalog or the label area; no VTOC processing is performed (see Figure 7-6).

DLBL Statement: The DLBL statement is used to find the 44-byte name of the file in the catalog. The 44-byte name matches the *file-ID* parameter. For PRINT and REPRO and VSAM application programs, the CAT operand is required only if you want to override the system's assumption that the job catalog, or, if there is none, the master catalog, owns the file. The function of the job catalog is explained under "Using a Job Catalog" in Chapter 5.

The BUFSP operand can be used to increase the amount of buffer space to be allocated for the file (see Figure 7-6).

Volume Mounting: If the volumes allocated to the file are not mounted, messages are issued to the operator to mount the required volumes or terminate the job. A file can span a maximum of 16 volumes. If a multivolume file is opened for direct or keyed-sequential processing, all volumes must be mounted. If it is opened for addressed-sequential processing, only one volume at a time need be mounted.

You can extend a suballocated file if:

- Secondary space allocation was specified when the file was defined.
- A volume that contains or can contain part of the file has unused data space of the required class.

Use the ALTER command of Access Method Services to make more volumes available to the file after it has been defined.

The VOL1 label is checked to verify that the correct volume is mounted (volume serial number), and the format-4 label is checked to verify that the catalog is at the proper level (volume timestamp). Processing for these labels is described under "VSAM Data Space" above.

Job Stream Examples

Figures 7-2 through 7-6 show examples of the job streams you must supply to:

1. Define a data space.
2. Define a file into a non-recoverable catalog.
3. Define a file into a recoverable catalog.
4. Define a unique file.
5. Process a file.

Figure 7-2 shows examples of the job streams you must supply to define data spaces. It shows allocation of an entire volume to VSAM (as a single data space) and allocation of a data space that is smaller than a single volume. The third and sixth examples show allocation of data spaces on different volumes of the same device type. DEFINE command parameters supply the data space information.

A DLBL statement for the master catalog is required. In the following examples, it is assumed to be in the label information area.

```
// JOB      ALLOCATE A VOLUME TO VSAM
* VOLUME IS OWNED BY MASTER CATALOG
* ALL UNALLOCATED SPACE IS GIVEN TO VSAM
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   SPACE(DEDICATE-
            VOLUMES(333001))
/*
/&

// JOB      DEFINE A VSAM DATA SPACE ON A 3330 VOLUME
* VOLUME IS OWNED BY MASTER CATALOG
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   SPACE(ORIGIN(760) TRACKS(570)-
            VOLUMES(333002))
/*
/&

// JOB      DEFINE VSAM DATA SPACES ON SEVERAL 3330 VOLUMES
* VOLUMES ARE OWNED BY USER CATALOG MYUCAT
// DLBL IJSYSUC,'MYUCAT',, VSAM
* DEFAULT ORIGIN USED FOR DATA SPACE ALLOCATION
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   SPACE(TRACKS(190)-
            VOLUMES(333003,333004))
/*
/&

// JOB      ALLOCATE A 3310 VOLUME TO VSAM
* VOLUME IS OWNED BY MASTER CATALOG
* ALL UNALLOCATED SPACE IS GIVEN TO VSAM
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   SPACE(DEDICATE-
            VOLUMES(331001))
/*
/&

// JOB      DEFINE A VSAM DATA SPACE ON A 3310 VOLUME
* VOLUME IS OWNED BY MASTER CATALOG
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   SPACE(ORIGIN(960) BLOCKS(2240)-
            VOLUMES(331002))
/*
/&

// JOB      DEFINE VSAM DATA SPACES ON SEVERAL 3370 VOLUMES
* VOLUMES ARE OWNED BY USER CATALOG MYUCAT
// DLBL IJSYSUC,'MYUCAT',, VSAM
* DEFAULT ORIGIN USED FOR DATA SPACE ALLOCATION
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   SPACE(BLOCKS(3100)-
            VOLUMES(337001,337002))
/*
/&
```

Figure 7-2. Examples of Job Streams to Create (Define) VSAM Data Spaces

Figure 7-3 is an example of the job stream you must submit to define a file that is suballocated from an existing data space. This file is recorded in the master catalog.

```
// JOB      SUB-ALLOCATE VSAM FILE
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   CLUSTER-
            NAME(MSTRFIL1)-
            VOLUME(333002)TRACKS(285 19))
/*
/&
```

Figure 7-3. Example of a Job Stream to Create (Define) a VSAM File (with file-ID MSTRFIL1) that is Suballocated from an Existing Data Space

Figure 7-4 is an example of defining a suballocated file into a recoverable user catalog. The volume described must be mounted during the DEFINE operation because the catalog information about the volume is duplicated into the volume's catalog recovery area (CRA).

```
// JOB      DEFINE A FILE INTO A RECOVERABLE CATALOG
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   CLUSTER(NAME(MSTRFIL2)-
            VOLUMES(333003)-
            TRACKS(100 20))-
            CATALOG(MYUCAT)
/*
/&
```

Figure 7-4. Example of a Job Stream to Create (Define) a VSAM File into a Recoverable Catalog.

In Figure 7-5, EXTENT statements supply data space information. Access Method Services requires the VOLUMES and CYLINDERS (BLOCKS, TRACKS, or RECORDS) parameters in the DEFINE command.

```
// JOB      ALLOCATE A UNIQUE VSAM FILE
// DLBL     VDATANM,,,VSAM
// EXTENT   ,333002,1,,1330,380
// DLBL     VINDXNM,,,VSAM
// EXTENT   ,333002,1,,1710,190
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   CLUSTER(NAME(MSTRFIL3)UNIQUE)-
            DATA(FILE(VDATANM)VOLUMES(333002)CYLINDERS(20))-
            INDEX(FILE(VINDXNM)VOLUMES(333002)CYLINDERS(10))
/*
/&
```

Figure 7-5. Example of a Job Stream to Create (Define) a Unique VSAM File (with file-ID MSTRFIL3).

In the example shown in Figure 7-6, the CAT parameter of the DLBL statement indicates the filename of the user catalog in which the file is recorded. The CAT parameter is written into the label information area. For details on the use of this parameter, refer to the Job Control Language chapter in this manual.

```
// JOB      PROCESS A VSAM FILE
// DLBL     VFILENM,'MSTRFILE',,VSAM,CAT=PRIVCAT
            (for the file)
// DLBL     PRIVCAT,'MYUCAT',,VSAM
// EXEC     USERPGM,SIZE=20K
   CSECT
     .
     .
     .
   ACB  DDNAME=VFILENM,...
     .
     .
     .
   END
/*
/&
```

Figure 7-6. Example of a Job Stream to Process an Existing VSAM File with an Assembler Program

Notes:

1. Other parameters are required in the DEFINE command in Figures 7-3, 7-4, and 7-5 to specify the characteristics, such as logical record length, of the VSAM file. These parameters do not affect space allocation and label processing, so they are not shown.
2. *Using VSE/VSAM Commands and Macros* describes the DEFINE command and provides more information about the job control statements required for VSAM.

Chapter 8: ISAM Interface Program

This section is for users of ISAM who are converting to VSAM. It contains detailed information which you need to decide whether existing ISAM programs can use the ISAM Interface Program (IIP) to process files that have been converted from ISAM format to VSAM format. Specifically, the IIP minimizes your conversion costs and scheduling problems by permitting ISAM programs to process VSAM files. ISAM programs can process ISAM files and VSAM files concurrently through the IIP.

Comparison of VSAM and ISAM

In most cases you can get better performance with VSAM while achieving essentially the same results that can be achieved with ISAM; VSAM can also achieve results that cannot be achieved with ISAM. The extent to which you can use your existing ISAM processing programs to process key-sequenced files relates to the similarities between ISAM and VSAM, as well as to limitations of the IIP. The following sections describe the similarities and differences between VSAM and ISAM in the areas that you are familiar with from using ISAM, and indicate the functions of VSAM that have no counterpart in ISAM.

Differences Between ISAM and VSAM

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

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 can be kept in storage. VSAM keeps individual index records in storage during processing, the number depending on the amount of buffer space provided.

Relation of Index to Data

The relation of a VSAM index to the direct access storage space whose records it controls is quite different from the corresponding relation for ISAM, in particular with regard to overflow areas for record insertion.

ISAM keeps a two-part index entry for each primary track on which a file is stored. 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 overflow record from that primary track. All the records in the overflow area from a primary track are chained together, from the lowest-keyed to the highest-keyed, by pointers that ISAM follows to locate an overflow record. Overflow records are unblocked, even if primary records are blocked.

VSAM does not distinguish between primary and overflow areas. A control interval, whether used or free, has an entry in the sequence set, and after records are stored in a free control interval, it is processed in exactly the same way 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.

Defining and Loading a File

All VSAM files are defined in a catalog. Records are loaded into a file with Access Method Services or with the processing program, in one execution or in stages. When loading new records into an empty key-sequenced file, the index is built automatically. Access Method Services does not merge input files. For a key-sequenced file, however, input records are merged in key sequence with existing records of the output file.

Deletion of Records

With ISAM, records cannot be deleted until the file is reorganized; you must mark the records you want to delete. VSAM automatically reclaims the space in a key-sequenced file and combines it with any existing free space in the affected control interval. VSAM's use of distributed free space for insertions and deletions requires less file reorganization than ISAM does.

VSAM Functions That Go beyond ISAM

VSAM Functions Available via IIP

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, beyond the primary space allocation. You can specify the amount of secondary space in number of data records or in number of blocks (FBA), tracks, or cylinders (CKD).

Automatic File Reorganization: VSAM partially reorganizes a key-sequenced file by splitting a control area when it has no more free control intervals and one is needed to insert a record. VSAM allocates a new control area and gives it the contents of approximately half of the control intervals of the old control area; about half of the control intervals of each control area are then free.

Keyrange Allocation: With a multi-volume key-sequenced file, you can assign data to various volumes according to the ranges of key values in the data records. For a file that resides on three volumes, you might assign keys A-E to the first volume, F-M to the second volume, and N-Z to the third.

Automatic Close: Because it is essential for the integrity of a file that it be closed properly, VSAM attempts to close all open VSAM files within the partition at both normal or abnormal termination of the job step. It also restores control blocks to their status before the file was opened, and frees storage that open routines used for VSAM control blocks.

Job Control: ASSGN or EXTENT statements are not required for file access. The IIP supports disposition processing (DISP parameter on DLBL statement) for reuseable and dynamic files.

VSAM Functions Requiring Conversion from ISAM

If you convert your ISAM programs to VSAM, these additional VSAM functions become available to you.

Addressed Sequential Access: You can retrieve and store the records of a key-sequenced file by RBA, as well as by key. With ISAM you can position by physical address, but you must retrieve in a separate request.

Direct Retrieval by Generic Key: With VSAM, you can retrieve a record directly, not only with a full-key search argument, but also with a generic search argument. ISAM can only position a record by generic argument; you must retrieve the record separately.

Concurrent Request Processing: A processing program can issue concurrent requests for a single ACB. The requests can be sequential or direct, or both, for the same part or different parts of the file. VSAM maintains a position in the file for each concurrent request.

No Abnormal Terminations by Open: The 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.

Alternate Indexes for Key-Sequenced and Entry-Sequenced Files: Instead of only one index, you can build several indexes, called alternate indexes, for a single data file. Each index can access the file in a different way so that you need not keep multiple copies of the same information organized differently for different applications.

Variable-Length and Spanned Records: In addition to fixed-length records, VSAM supports variable-length and spanned records.

Skip Sequential Access: You can process a key-sequenced file sequentially and skip records automatically, as though you were using direct access.

How To Use The ISAM Interface

To use the IIP, you must convert the ISAM files of your programs to VSAM files. You must also change the job control statements for your ISAM programs to meet the requirements of 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. If they do, there is no need to reassemble or re-linkedit these programs.

Converting an ISAM File to a VSAM File

To convert an ISAM file to a VSAM file, you must first use the Access Method Services DEFINE command to define a key-sequenced VSAM file. The VSAM file may be defined on a volume that already contains enough free VSAM data space for it, or data space may be defined along with the file (unique file). The use of the DEFINE command is fully explained in *Using VSE/VSAM Commands and Macros.*

Some of the information given in the DTFIS parameters must, for VSAM, be specified in the DEFINE command. These parameters and the corresponding DEFINE command options are:

DTFIS parameter	VSAM DEFINE option
HOLD=YES	SHAREOPTIONS(4).

KEYLEN=n and
KEYLOC=n
	KEYS (length, offset)
	length should always be set to KEYLEN; offset should be set to KEYLOC-1 if DTFIS RECFORM=FIXBLK, or to 0 if RECFORM=FIXUNB.

RECSIZE=n	RECORDSIZE (average, maximum). The average and maximum values must be equal. If RECFORM=FIXBLK in the DTFIS, RECORDSIZE should be set to RECSIZE. If RECFORM=FIXUNB, RECORDSIZE should be set to RECSIZE + KEYLEN.
VERIFY=YES	WRITECHECK.

If you have a file that will require rebuilding, initially specify the REUSE parameter on the Access Method Services DEFINE command; specify DISP=NEW on the DLBL statement when reloading the file.

The BUFFERSPACE parameter in the DEFINE command specifies how much space VSAM will have for I/O buffers. If you do not specify the BUFFERSPACE parameter, the default is at least two data buffers and one index buffer. For better performance, however, you can specify space for more than two data buffers and one index buffer.

The following DTFIS parameters are used by the IIP; all other parameters are ignored:

```
ERREXT=YES (see Figure 8-2 for a description of the
   ERREXT parameter list with IIP)
IOAREA=name (used when IOROUT=LOAD)
IOAREAS=name (used if SETL BOF is issued)
IOREG=(r)
IOROUT=LOAD, ADD, RETRVE, ADDRTR
KEYARG=name
RECFORM=FIXUNB, FIXBLK
WORKL=name
WORKR=name
WORKS=YES
```

After you have defined your VSAM file, you must load the VSAM file by copying the ISAM file into it. You may use your ISAM load program, by way of the IIP, or you may use the Access Method Services REPRO command. The REPRO command is described in *Using VSE/VSAM Commands and Macros.* If you have records marked for deletion in the ISAM file and do not want them copied into the VSAM file, you should use your ISAM load program because the REPRO command will copy all records from the ISAM file, including those marked for deletion.

Note: REPRO of a fixed, unblocked ISAM file creates records consisting of the original record preceded by its key. The IIP strips this leading key when a program specifying fixed unblocked ISAM is executed, and returns only the original record to you. The leading key is returned with the record, however, when the file is accessed in native VSAM mode.

Figure 8-1 summarizes the steps required to convert indexed sequential files to key-sequenced files and processing them either with programs that have been converted from ISAM to VSAM or with programs that still use ISAM.

Changing ISAM Job Control Statements

The job control statements for ISAM must be replaced by VSAM job control statements. An example of VSAM job control statements used with an ISAM program is shown below:

```
// JOB    PROCESS A VSAM FILE
// DLBL   IFN,'MSTRFILE',,VSAM
// EXEC   ISAMPGM,SIZE=nK
CSECT
 .
 .
IFN DTFIS . . .
 .
 .
END
/*
/&
```

One DLBL statement is required for the file; it connects the ISAM filename (IFN) to the VSAM cluster name (MSTRFILE) stored in the catalog. The DLBL type code parameter (VSAM) causes the ISAM Interface Program to be called. The same VSAM job control statements are required regardless of the type of ISAM program.

What the ISAM Interface Program Does

When a processing program that uses ISAM opens a VSAM file, the VSE open routine detects the need for the ISAM Interface Program by the type code VSAM in the DLBL statement. It calls the IIP open routine to build control blocks required by VSAM, to load the ISAM command processor, and to flag the DTFIS for the IIP to intercept ISAM requests.

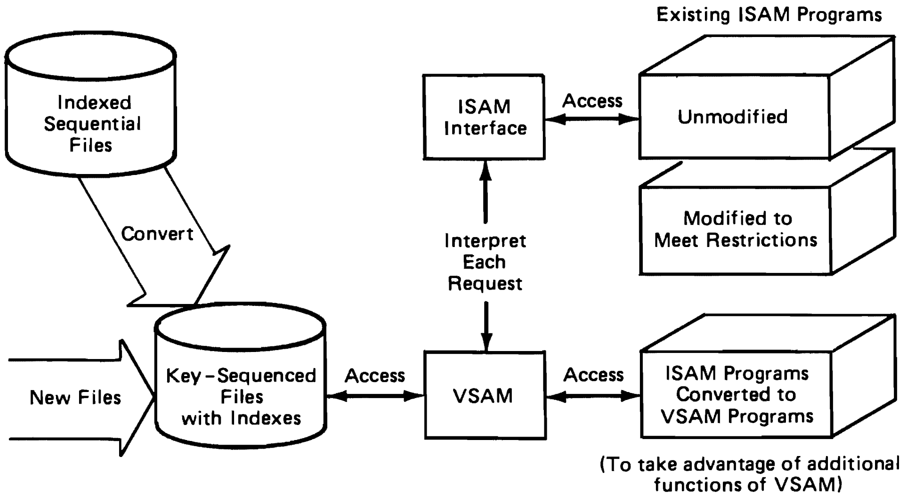

Existing ISAM Programs
Indexed Sequential Files
Convert
New Files
Key-Sequenced Files with Indexes
Access
ISAM Interface
Access
Interpret Each Request
VSAM
Access
Unmodified
Modified to Meet Restrictions
ISAM Programs Converted to VSAM Programs
(To take advantage of additional functions of VSAM)

Figure 8-1. Using the ISAM Interface: Most existing programs that use ISAM can process VSAM files through the interface with little or no change.

The IIP intercepts each subsequent ISAM request, analyzes it to determine the equivalent keyed VSAM request, which it defines in the RPL constructed by OPEN, and then initiates the request.

The IIP interprets VSAM's return codes and, if the VSAM condition corresponds to an ISAM condition, turns on the respective bit in the filenameC byte in the DTFIS. For irrecoverable errors that cannot be posted in the filenameC byte, the IIP prints a message, closes the VSAM file (by the VSAM close routine), and ends the job. If a physical I/O error occurs and ERREXT=YES was specified in the DTFIS, the IIP transfers additional error information to the processing program.

Figure 8-2 shows the format of the ERREXT parameter list, and Figures 8-3 and 8-4 show the formats of the filenameC byte for ISAM processing through the IIP.

Bytes	Bits	Contents
0-3	-	DTF address
4-15	-	Not supported by the IIP
16	0	Data
	1	VSAM sequence set
	2	VSAM index set
	3-5	Not used
	6	Read operation
	7	Write operation
17	-	Not supported by the IIP

Figure 8-2. ERREXT parameter list for ISAM programs with IIP.

Bit	Cause in ISAM	Cause in IIP/VSAM
0	DASD error	DASD error
1	Wrong length record	Not set
2	End of file	End of file
3	No record found	No record found
4	Illegal ID specified	Not supported by IIP
5	Duplicate record	Duplicate record
6	Overflow area full	No more VSAM data space available
7	Overflow	Not set

Figure 8-3. FilenameC with IIP when IOROUT=ADD, RETRVE, or ADDRTR

Bit	Cause in ISAM	Cause in IIP/VSAM
0	DASD error	DASD error
1	Wrong length record	Not set
2	Prime data area full	No more VSAM data space
3	Cylinder index area full	No more VSAM data space
4	Master index full	No more VSAM data space
5	Duplicate record	Duplicate record
6	Sequence check	Sequence check
7	Prime data area overflow	Not set

If there is no more VSAM data space, bits 2 through 4 are set.

Figure 8-4. FilenameC with IIP when IOROUT=LOAD

Restrictions in the Use of the ISAM Interface Program

Most programs that use ISAM require little or no modification when using the IIP to process VSAM files. Following is a list of ISAM functions for which there is no VSAM equivalent or which cannot be simulated by the IIP.

- The program must run successfully under ISAM. IIP 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 VSAM does not use the record ID functions.
- VSAM does not return device-dependent information or the storage or DASD address of the record containing the error in the ERREXT parameter list.
- The ISAM program cannot open a DTF while another ISAM DTF or VSAM ACB is already open for the same file unless VSAM SHAREOPTIONS(3) or SHAREOPTIONS (4) was specified for the file in the DEFINE or ALTER command. If you select SHAREOPTIONS (3), you must accept the responsibility of maintaining file integrity.
- Files defined with SHAREOPTIONS(4) cannot be shared between IIP users in different systems because IIP always opens a file for output.

Chapter 9: Optimizing VSAM's Performance

This chapter contains *guidelines,* not rules, for enhancing your system's efficiency. This kind of information is difficult to present because of the great number of variables that exist; not everything will be true for all installations under all conditions. We do hope you find this chapter generally useful, but you should be aware that the material it contains may require altering, depending on the needs of your installation.

This section explains VSAM options that affect performance and requirements for virtual storage and direct access storage: the classification of data space, the size of control intervals, control areas, and I/O buffers; the percentage of distributed free space; the division of key-sequenced data into key ranges; and the handling of indexes and user catalogs. These options are specified in the DEFINE command when a file is created and in the ACB or GENCB macro when a processing program prepares to open a file. This section also discusses file statistics that VSAM makes available.

Data Space Classification

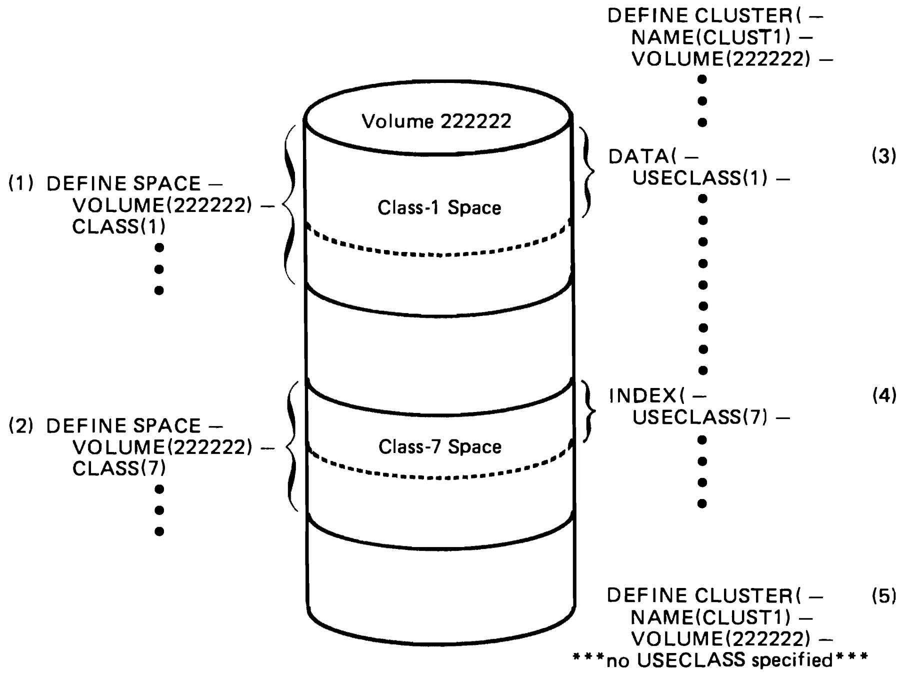
VSAM data space can be classified (assigned a value) in order to direct the suballocation of data space to VSAM objects and thus maximize performance. You assign a value to a data space with the CLASS (value) parameter of the DEFINE SPACE, DEFINE MASTERCATALOG, and DEFINE USERCATALOG commands.

After you have assigned a value to a data space, you can request that it be available for suballocation to an alternate index or cluster (or their components) through the USECLASS parameter of the DEFINE CLUSTER, DEFINE ALTERNATEINDEX, IMPORT, or IMPORTRA commands.

You can assign a data space to any one of eight performance classes (0 is the default):

- 0 - General use. (Data spaces defined under DOS/VS Release 34 and prior releases, and OS/VS are treated as class-0 data spaces.)
- 1 - High performance (suggested for fixed-head areas specifically).
- 2-7 - User-defined classes (for example, data space in the middle of a volume).

Figure 9-1 illustrates the classification of data space and the use of classified data space.



```
(1) DEFINE SPACE -
      VOLUME(222222) -
      CLASS(1)
      •
      •
      •
```

```
(2) DEFINE SPACE -
      VOLUME(222222) -
      CLASS(7)
      •
      •
      •
```

```
DEFINE CLUSTER( -
  NAME(CLUST1) -
  VOLUME(222222) -
  •
  •
  •
DATA( -                    (3)
  USECLASS(1) -
  •
  •
  •
  •
  •
  •
  •
  •
  •
INDEX( -                   (4)
  USECLASS(7) -
  •
  •
  •
  •
```

```
DEFINE CLUSTER( -          (5)
  NAME(CLUST1) -
  VOLUME(222222) -
***no USECLASS specified***
```

Notes: (1) Class-1 data space defined.

(2) Class-7 data space defined.

(3) Class-1 data space suballocated to the data component of CLUST1.

(4) Class-7 data space suballocated to the index component of CLUST1.

(5) This DEFINE command fails because the default class (0) is not available on volume 222222.

Figure 9-1. Classification of Data Space

Since the definition of VSAM catalogs involves the implicit allocation of data space and the suballocation of some (or all) of that data space to the catalog itself, you need only specify the CLASS parameter if you want to assign a catalog's data space to a certain performance class. You do not have the option of specifying the USECLASS parameter. The catalog is automatically suballocated from the same data space and the same performance class.

A new class can be requested via the IMPORT or IMPORTRA commands (USECLASS parameter) when an object is implicitly defined through those commands.

The following restrictions apply:

- Classes other than 0 are not permitted for unique objects.
- In the case of imbedded sequence sets, the sequence set is suballocated from space of the same class as the data component.

For the DEFINE command, USECLASS must be specified concurrently (at the same level) with the space parameters (TRACKS, BLOCKS, etc.). For example, USECLASS specified at the data level of DEFINE CLUSTER is ineffective unless CYLINDERS, TRACKS, BLOCKS, or RECORDS is also specified at the data level. Following are the three possible combinations of levels at which space may be specified for DEFINE CLUSTER or DEFINE ALTERNATEINDEX:

(a) CLUSTER|ALTERNATEINDEX level only

(b) DATA level only

(c) DATA and INDEX levels

Therefore these are also the levels that are effective for USECLASS.

In case (a), the USECLASS specified (or defaulted to) is also applied to the data and index components.

In case (b), the USECLASS specified, defaulted to, or modeled for the data level is also applied to the index level. This permits you to apply the same class of data space to both components while leaving the calculation of the index allocation to VSAM.

In case (c), the data and index components may be assigned (or modeled or defaulted) to a separate or to the same class of data space, depending on the values chosen.

See Appendix A of *Using VSE/VSAM Commands and Macros* for examples of assigning classes of data space.

Min-CA, Max-CA

The terms "tracks" and "cylinders" as used for CKD (count-key-data) devices are not necessarily meaningful for an FBA (Fixed Block Architecture) device because an FBA device stores data on "blocks". FBA uses a linear addressing scheme whereby blocks are not necessarily associated with physical characteristics, such as cylinders or tracks. The following new terms that are common to both CKD and FBA devices are being introduced into the documentation to describe VSAM's use of the track and cylinder concepts to optimize performance and to control allocation, while incorporating FBA device support (in which there is no dependency on the physical characteristics of a device).

Min-CA replaces the former term "track"; it represents minimum control area size for both CKD and FBA devices. Max-CA replaces the former term "cylinder"; it represents maximum control area size for both CKD and FBA devices. Min-CA and max-CA are units of allocation. The size of these units of allocation depends upon the device being used (see Figure 9-2). Note that the CKD devices have more than one min-CA and max-CA value listed; this is because the min-CA (formerly track) and max-CA (formerly cylinder) sizes for CKD devices are dependent upon the size and number of physical records contained in them (a function of the control interval size selected). Refer to Figure 9-4 to determine the appropriate min-CA or max-CA size for a particular control interval size. For example (referring to Figure 9-4), a 3340 with a control interval size of 6K bytes uses 7.5K bytes of track space; therefore, it has a min-CA of 7.5K bytes.

Device Type	Min-CA and Max-CA values	
	FBA Devices	
3310	Min-CA	32 blocks (16K bytes)
3310	Max-CA	352 blocks (176K bytes)
3370	Min-CA	62 blocks (31K bytes)
3370	Max-CA	744 blocks (372K bytes)
	CKD Devices	
2314/2319	Min-CA (trk)	5.5K, 6K, 6.5K, or 7K bytes
2314/2319	Max-CA (cyl)	110K, 120K, 130K, or 140K bytes
3330	Min-CA (trk)	10K, 10.5K, 11K, or 12K bytes
3330	Max-CA (cyl)	190K, 199.5K, 209K, or 228K bytes
3340	Min-CA (trk)	6K, 6.5K, 7K, 7.5K, or 8K bytes
3340	Max-CA (cyl)	72K, 78K, 84K, 90K, or 96K bytes
3350	Min-CA (trk)	13.5K, 15K, 16K, 16.5K, 17.5K, or 18K bytes
3350	Max-CA (cyl)	405K, 450K, 480K, 495K, 525K, or 540K bytes

Figure 9-2. Table of Min-CA and Max-CA Values

Control Interval Size

A file's control-interval size affects performance. The CI is VSAM's unit of transmission between DASD and main storage. Using large CIs permits more data to be transferred at once, resulting in less system overhead. As a general rule, the larger the data control interval the better the *sequential* performance, for a number of reasons:

- Fewer index records required for a key-sequenced file.
- Fewer control-interval accesses (significant only for sequential or skip sequential access).
- More efficient distribution of free space in a key-sequenced file.

Note that EXPORT(RA)/IMPORTA(RA) is a sequential application. If you use them frequently, take this into account when selecting data CI size.

Control interval size, which can be specified for entry-sequenced, key-sequenced, and relative-record files, as well as for alternate indexes, also affects record-processing speed and requirements for main and direct access storage, as follows:

- As the size of your nonspanned data records increases, you may need larger data control intervals.
- As data and index control interval size increases and record size remains unchanged, more buffer space is required in storage for each control interval.
- As data control interval size increases, fewer I/O operations (control interval accesses) are required to bring a given number of records into storage; fewer index records must be read. This is usually significant only for sequential and skip sequential access.
- Free space will probably be used more efficiently (fewer control interval splits and less wasted space) as data control interval size increases relative to data record size, especially with variable-length records.

  (Free space in a nonspanned data control interval isn't used if there isn't enough for a complete data record. In any event, free space in the last control interval of a spanned record is never used for any other record, even if there is room enough to hold a complete data record.)

Direct processing is less sensitive to data CI size. Smaller data CIs generally improve performance because unused records don't have to be transferred. An exception to this case occurs if you are processing presorted input. VSAM checks its buffers for data before reading the DASD, so in this case large data CIs may be beneficial.

If you have a choice between a large index CI or a large data CI (during direct processing), choose the combination that yields the smallest buffer space value. This combination requires the least amount of active storage and the least amount of data transfer time.

You can let Access Method Services select the size of a control interval for a data or index component or you can request a particular control interval size in the DEFINE command. Control interval size should be specified at the DATA and INDEX levels; if it is specified at the CLUSTER or ALTERNATEINDEX level, the size applies not only to the data component, but to the index component as well. The size you specify must, however, fall within acceptable limits determined by Access Method Services, or the DEFINE will fail. These limits depend on the maximum size for nonspanned records or the average size for spanned records, which you specify by the RECORDSIZE parameter of the DEFINE command.

The size of a control interval must always be a multiple of 512 bytes, because a control interval is a whole number of physical records. Physical record size for the data component includes all multiples of 512 bytes up to 8,192 bytes. Figure 9-4 illustrates the relationship of control interval sizes to physical record sizes for the data component for the different types of CKD devices.

The size of a control interval in the INDEX component can be any multiple of 512, up to 8,192 bytes.

The size of a control interval in the DATA component of a cluster can be any multiple of 512, up to 32,768, except that if it is over 8192 bytes, it must be a multiple of 2048:

512, 1024, 1536, ...,
8192, 10240, 12288, ..., 32768.

If you specify a control interval size that is not a proper multiple, Access Method Services increases it to the next multiple. For example, 2050 is increased to 2560.

For nonspanned records, the size of a control interval in a data component must be large enough to hold a data record of the maximum size specified in the RECORDSIZE parameter. Control information requires seven bytes; therefore, the control interval must be at least seven bytes larger than the largest record in the data component.

For spanned records, the size of a control interval in a data component must be as large as the average size specified in the RECORDSIZE parameter. Control information requires ten bytes for each control interval that the spanned record contains; therefore, the control interval must be at least ten bytes larger than the average record in the data component.

Index CI Size

Pick the smallest index CI you can. Generally a 512-byte index CI is adequate if:

- The number of data CIs per control area is small;
- The full key size is not too large; and
- The key compresses well (usually the case when the data CI is 4K or greater).

To determine whether a 512-byte index CI size is sufficient, do the following experiment using your chosen data CI size and a 512-byte index CI. Allow no freespace, and load enough records to equal one control area. At the end of the run, perform a LISTCAT; if there is only one level of index, a 512-byte index CI is large enough. For n control areas, there should be two levels of index with the number of index CIs equal to n + 1.

A smaller data CI may require a large index CI. The sequence set index CI contains pointers to the data CIs in a control area. If the data CI is made smaller (when the control area stays the same size), there will be more data CIs per control area, and therefore more entries in the sequence set. As an example, assume a one cylinder control area size on a 3340. Using 4096-byte data CIs, one control area can contain 24 data CIs. If the data CI size were changed to 1024 bytes, the control area could contain 84 data CIs. The sequence set would now require 84 pointers instead of 24.

For a key-sequenced file, after control interval size has been set, Access Method Services determines the number of bytes to be reserved for free space, if any. For example, if the control interval size is 4096, and the percentage of free space in a control interval is twenty, 820 bytes are reserved (4096 x 20% = 820.)

If you don't specify a size for data control intervals, Access Method Services uses 2048, if possible. If you don't specify a size for index control intervals, Access Method Services uses 512, if possible. After Access Method Services determines the number of control intervals in a control area (see "Control Area Size"), it estimates whether one index control interval is large enough to handle all of the data control intervals in a control area. If the index control interval is not large enough, its size is increased, if possible. If it is not possible, the number of control intervals in a control area is decreased.

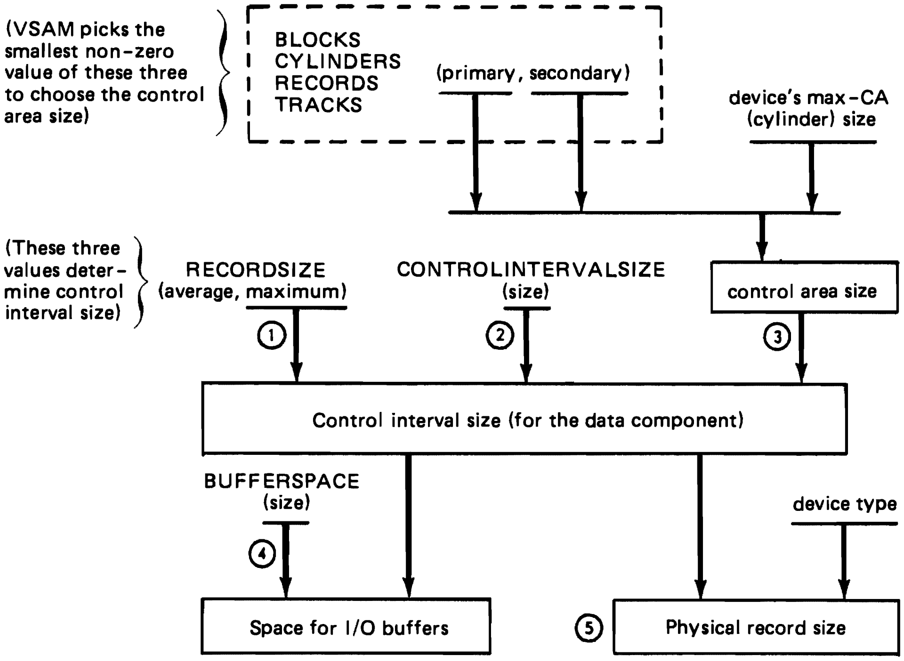
To find out what values are actually set in a defined file, you can issue the Access Method Services LISTCAT command or, while your program is being executed, the SHOWCB macro.

For example, for a file without an index, if control interval size space is not specified and the maximum record size is specified to be 200 bytes, Access Method Services sets data control interval size to 2048 bytes. For a key-sequenced file, Access Method Services additionally sets index control interval size to 512 bytes.

If spanning is not specified and the maximum data record size specified in RECORDSIZE is 2500 bytes, and 2500 is also specified for the data control interval size, the system adjusts the 2500-byte control interval size to the next higher multiple of 512: 2560.

Data CI Size and Physical Record Size

Figure 9-3 shows how VSAM computes physical record size, using DEFINE attributes and device type.


(VSAM picks the smallest non-zero value of these three to choose the control area size)
BLOCKS
CYLINDERS
RECORDS
TRACKS
(primary, secondary)
device's max-CA (cylinder) size
(These three values determine control interval size)
RECORDSIZE (average, maximum)
CONTROLINTERVALSIZE (size)
control area size
①
②
③
Control interval size (for the data component)
BUFFERSPACE (size)
device type
④
Space for I/O buffers
⑤
Physical record size

① Maximum record size can be specified as 1 through 32761 bytes; the default is 4089 bytes. RECORDSIZE (maximum) plus 7 must be less than or equal to control interval size.

② The data component's CONTROLINTERVALSIZE can be specified as 512 through 32768 bytes. The default is:

- 2048 if RECORDSIZE is specified.
- 4096 if RECORDSIZE is not specified.

③ The control area size, chosen by VSAM, is never larger than 1 max-CA (cylinder).

④ BUFFERSPACE (size) must be enough space to contain two data control intervals and, if key-sequenced, one index control interval. This is also the default. If you specify a value less than the default, the command is terminated.

⑤ The physical record size, chosen by VSAM, depends on the device type being used and the control interval size. VSAM chooses the following:

- For FBA devices - - always 512.
- For CKD devices - - 512 to 8192 in multiples of 512.

Figure 9-3. VSAM's Determination of Physical Record Size

Figure 9-4 shows the physical record size VSAM uses for a data CI, depending on the device and the specified CI size, and the number of K bytes of user data that can be accommodated on the track. Given a 6K control interval size on a 3350, VSAM chooses a physical record size of 6K that results in 18K bytes (plus overhead) of data on a 19,069-byte track.

Control interval size affects space utilization because of the way VSAM chooses physical record sizes on CKD devices. (There are no similar considerations for FBA devices.) For a given CI size, VSAM chooses the physical record size that results in the most efficient use of track capacity.

CI Size	Physical Record Size				Track Space Used			
	2314/ 2319	3340	3330	3350	2314/ 2319	3340	3330	3350
0.5	0.5	0.5	0.5	0.5	5.5	6	10	13.5
1	1	1	1	1	6	7	11	15
1.5	1.5	1.5	1.5	1.5	6	7.5	10.5	16.5
2	2	1	2	2	6	7	12	16
2.5	0.5	2.5	2.5	2.5	5.5	7.5	10	17.5
3	3	1.5	3	1.5	6	7.5	12	16.5
3.5	0.5	3.5	3.5	3.5	5.5	7	10.5	17.5
4	2	4	4	4	6	8	12	16
4.5	1.5	1.5	1.5	4.5	6	7.5	10.5	18
5	1	2.5	1	2.5	6	7.5	11	17.5
5.5	5.5	0.5	5.5	5.5	5.5	6	11	16.5
6	6	1.5	6	6	6	7.5	12	18
6.5	6.5	6.5	0.5	0.5	6.5	6.5	10	13.5
7	7	7	1	3.5	7	7	11	17.5
7.5	1.5	7.5	1.5	2.5	6	7.5	10.5	17.5
8	2	8	4	8	6	8	12	16
10	2	2.5	2	2.5	6	7.5	12	17.5
12	6	4	6	6	6	8	12	18
14	7	7	2	3.5	7	7	12	17.5
16	2	8	4	8	6	8	12	16
18	6	1.5	6	6	6	7.5	12	18
20	2	4	4	2.5	6	8	12	17.5
22	2	1	2	5.5	6	7	12	16.5
24	6	8	6	6	6	8	12	18
26	6.5	1	2	2	6.5	7	12	16
28	7	4	4	3.5	7	8	12	17.5
30	6	7.5	6	6	6	7.5	12	18
32	2	8	4	8	6	8	12	16

Figure 9-4. Relationship of Control Interval Size to Physical Record Size for Data Component (Numbers in K Bytes). Applies only to CKD Devices.

Note: A file with a data physical record size or index CI size other than .5, 1, 2, or 4K cannot be directly processed by OS/VS. (File portability between VSE/VSAM and OS/VS via EXPORT/IMPORT (or EXPORTRA/IMPORTRA) is not impacted by data physical record size, but it does require an OS/VS-compatible CI size.

Key Compression

VSAM increases the number of entries that an index record can hold by key compression. Compression makes an index smaller by reducing the size of the keys in the index entries. VSAM eliminates from the front and back of a key those characters that are not needed to distinguish it from the adjacent keys. For example, the keys in the sequence 1110, 1230, 1450 would compress to 11, 23, 45 repectively.

Front compression works best when the keys of the last records of each CI run in a series (for example, 100, 101, 102, 106, etc.). When several high keys have the same leading characters, those characters can be compressed.

Rear compression works best when adjacent keys have large differences at the back of the key.

If keys compress poorly, more room is required in the index CI to store the compressed key. The index CI may be too small for the data. If it is too small, more control areas are needed. When VSAM has no more room to insert compressed keys from the data CIs into the index CI, it continues to load data into the next control area, using its associated sequence set CI. The previous control area contains fewer "filled" data CIs than if the index CI had been adequate.

Poor key compression can occur under the following conditions:

- The key is comprised of multiple fields.
- Changes occur in the front of the key and the back of the key, but not in the middle.
- If the number of keys in a group is less than the number of keys in a data CI, the high key in each data CI does not repeat the high-order characters. Therefore, front compression is almost non-existent.
- If the last field of the key is long and very dense, poor rear compression results.

Single field keys *do* compress well. Larger keys (20 - 30 bytes) can compress to 8 or 9 bytes (including control information). Smaller key (5 - 15 bytes) can compress to 3 - 5 bytes (including control information).

Example of a Key that Compresses Poorly

NNN0000000000SS

NNN changes every 4 or 5 records; there are more than 4 or 5 records per data CI.

SS changes in every record.

0000000000 changes rarely.

The key would compress well if:

- NNN changed every 20 - 25 records;
- SS seldom changed;
- SS were located next to NNN (NNNSS0000000000) and changed frequently; or
- The entire key was one field and the bytes changed randomly.

Control Area Size

For a key-sequenced file the size of a control area depends on the size of control intervals in the index component. Control area size has significant performance implications. When a whole number of control areas occupies a max-CA (cylinder), performance is better than when control areas cross max-CA (cylinder) boundaries. If you allocate space in a DEFINE command using the CYLINDERS parameter, or if a CKD file is defined as unique (the only one in its data space), Access Method Services sets the control area size to one max-CA (cylinder). If a control area is smaller than a max-CA (cylinder), its size will be an integral multiple of min-CAs (tracks), and it can cross max-CA (cylinder) boundaries. However, a control area can never cross the extent boundaries of a file; that is, an extent of a file is made up of a whole number of control areas.

Aside from specifying space in terms of max-CAs (cylinders) or defining a CKD file as unique, you don't have a direct way of specifying that a whole number of control areas will occupy a max-CA (cylinder). But you can provide values in the DEFINE command that will influence the control area size as computed by Access Method Services.

Access Method Services checks the smaller of the primary and secondary space values against the specified device's max-CA (cylinder) size. If the smaller space quantity is less than or equal to the device's max-CA (cylinder) size, the size of the control area is set equal to the smaller space quantity. If the smaller space quantity is greater than the device's max-CA (cylinder) size, the control area size is set equal to the max-CA (cylinder) size.

You specify space in number of tracks, cylinders, blocks or records; the system preformats space in control areas (except for DEFINE CLUSTER/AIX SPEED). By calculating the size of a control area as it does, Access Method Services is able to meet your primary and secondary space requirements without overcommitting space for this file.

An index record must be large enough to address all of the control intervals in a control area. The more control intervals an index record addresses, the fewer reads for index records are required for sequential access. Generally, the greater the size of the control area, the better the performance and space utilization.

Figure 9-5 shows VSAM data capacities of the various DASD devices.

Storage device	Volumes per device	Max-CAs/ cylinders per volume	Min-CAs per max-CA or tracks per cylinder	Bytes per min-CA/track (data)	Bytes per min-CA/track (index)
2314 2319	8	200	20	5,632- 7,168*	5,632- 6,144*
3310	1	360	11	16,384	16,384
3330-1 3330-2 3333	8	404	19	10,240- 12,288*	10,240- 12,288*
3330-11	8	808	19	10,240- 12,288*	10,240- 12,288*
3340	2	696	12	6,144- 8,192*	6,144 7,680*
3350	8	555	30	13,824- 18,432*	13.824- 16,384*
3370	2	750	12	31,744	31,744
* Depending on number of physical records (see Figure 9-4).					

Figure 9-5. Disk Storage Capacity Table for VSAM

I/O Buffer Space

I/O buffer space is important because VSAM transmits the contents of a control interval to a buffer in main or virtual storage; therefore, control interval size affects the use and size of I/O buffers and the amount of storage space for I/O buffers.

If you do not specify buffer space, VSAM allocates buffer space for two data control intervals and (if the file is indexed) one index control interval. You may not specify less space, but to optimize performance, you may want to provide additional buffer space.

Sequential Processing

Increasing the space to hold three data control intervals generally improves performance due to I/O command chaining, but it does not allow for I/O overlap with processing. To achieve both command chaining and overlap with processing, specify enough space for four or more data control intervals. More than four or five data buffers may cause excessive paging.

If there is an index component, buffer space must be large enough to hold an index control interval also. An index control interval is generally 512 or 1024 bytes. Because VSAM rounds buffer space to a 2K boundary, you can often provide space for the index control interval by simply allowing normal rounding beyond the space provided for data control intervals.

Direct Processing

Any remaining buffer space beyond that required for two data control intervals is used for index control intervals. To optimize performance, specify enough buffer space to accommodate one index control interval for each level of index. If the index control interval size or the number of index levels is not known, specify 2K of buffer space for the index (default BUFFERSPACE, which rounds to a 2K boundry, may in some cases accomplish this for you), and check the result with LISTCAT output. Make adjustments with ALTER, if necessary.

Buffer space can be specified in the Access Method Services DEFINE command, the ACB macro, and the DLBL statement. The buffer space entry in the catalog was either specified or defaulted to when the cluster was defined or modified with the ALTER command.

Buffer Specification

DEFINE:

Using DEFINE, you can specify the BUFFERSPACE parameter at the cluster or data level, but not both. The default buffer space allocation is two data buffers and one index buffer (key-sequenced data sets only). For ESDS and RRDS, the default is two data buffers.

ACB:

You can specify buffer space values or cause a default buffer space through the ACB macro.

```
ACB      .
         .
         .
         BUFSP=n
         BUFNI=n
         BUFND=n
```

To use the ACB buffer space, the value selected must be larger than the catalog entry buffer space. The use of ACB parameters is explained under "Buffer Allocation" later in this chapter.

DLBL:

Another way of specifying buffer space is through the use of the DLBL statement:

```
// DLBL filename,'file-ID',,VSAM,BUFSP=size
```

To be effective, the value specified for the DLBL buffer space must be larger than both the catalog entry buffer space and the ACB buffer space.

At execution time you may require more than the buffer space specified in the catalog, DLBL, or ACB. The minimum requirements for execution time buffers are as follows. (Default STRNO=1.)

data buffers = ACB STRNO + 1

index buffers = ACB STRNO

If STRNO = 2 (that is, you require concurrent file positioning), the minimum buffer space required for output is three data CIs and two index CIs.

If the amount of buffer space specified is greater than the minimum required, VSAM uses the remainder for additional index buffers (direct processing) or additional data buffers (sequential or skip sequential processing).

Buffer Allocation (Using Nonshared Resources)

This discussion explains how VSAM allocates buffer space according to ACB specification. The following ACB parameters relate to buffer allocation:

```
ACB  MACRF=(IN|OUT,SEQ|DIR|SKP)
     STRNO=n
     BUFSP=n
     BUFND=n
     BUFNI=n
```

Minimum Buffer Allocation

Data buffers:

`MACRF=(...,IN,...)`

The number of data buffers for ESDS and RRDS is the greater of BUFND or STRNO; for KSDS, the number of data buffers is the greater of BUFND or STRNO + 1.

`MACRF=(...,OUT,...)`

The number of data buffers is the greater of BUFND or STRNO + 1.

Index buffers:
The number of index buffers is the greater of BUFNI or STRNO.

OPEN calculates:

`Remainder=BUFSP-((NDB*DCI)+(NIB*ICI))`

NDB = number of data buffers

DCI = size of a data CI

NIB = number of index buffers

ICI = size of an index CI

If the remainder ≤ 0, then OPEN allocates the number of data buffers and index buffers and increases BUFSP to hold them.

If the remainder > 0, go to the next step (below) to calculate additional buffers.

No indication is given that the BUFSP used for the minimum allocation is greater than that specified in DEFINE, DLBL, or ACB.

Remainder > 0

1. MACRF=(...,SEQ,...)

VSAM allocates data buffers until there is a remainder that is less than the data CI size; then it allocates more index buffers. (This is only possible when the index CI size is less than the data CI size. If the index CI size is larger, see item 2 below.)

Example:

```
BUFSP=13824
data CI size=4096
index CI size=512
STRNO=1
MACRF=(...,SEQ,OUT,...)
```

			Cumulative Totals
Minimum allocation =	2 data buffers	8192	
	1 index buffer	512	8704
Additional allocation =	1 data buffer	4096	12800
(resulting from	2 index buffers	1024	13824
MACRF specification)			

2. MACRF=(...,DIR,...)

VSAM allocates more index buffers until there is a remainder that is less than the size of one index CI; then it allocates more data buffers. (This is possible only when the data CI size is less than the index CI size.)

Example:

```
BUFSP=13824
data CI size=4096
index CI size=512
STRNO=1
MACRF=(...,DIR,OUT,...)
```

			Cumulative Totals
Minimum allocation =	2 data buffers	8192	
	1 index buffer	512	8704
Additional allocation =	10 index buffers	5120	13824
(resulting from MACRF			
specification)			

3. MACRF=(...,SEQ,DIR,...)

VSAM increases the number of index buffers to twice STRNO. (If this is not possible, VSAM uses the procedure described in item 2 above.) If there is still a remainder, VSAM uses the procedure described in item 1 above to allocate the remainder.

Example:

```
BUFSP=13824
data CI size=4096
index CI size=512
STRNO=1
MACRF=(...,SEQ,DIR,OUT,...)
```

			Cumulative Totals
Minimum allocation =	2 data buffers	8192	
	1 index buffer	512	8704
Additional allocation =	1 index buffer	512	9216
(resulting from MACRF	1 data buffer	4096	13312
specification)	1 index buffer	512	13824

Later modifications of RPLs does not change buffer allocations.

Buffer Allocation for a Path

Path Entry for AIX

If the path entry is not a member of the upgrade set, buffers are allocated in the same manner as for a normal KSDS. Your ACB is used for the path entry.

If the path entry is a member of the upgrade set, then buffers are allocated as for a normal KSDS, but minimum allocations are increased by one for both the number of data buffers and the number of index buffers. Your ACB is used for the path entry.

Buffer Allocation for Path Entry when the Base Cluster is a KSDS

Buffers are allocated in the same manner as for a normal KSDS with the following ACB specifications:

```
BUFND=0
BUFNI=0
STRNO=number of strings specified in the ACB
```

You can influence buffer allocation only via the BUFFERSPACE parameter of DEFINE CLUSTER.

If you open the path for *input* only, the base cluster uses MACRF=(...,DIR,IN,...). If you open the path for *output*, the base cluster uses MACRF=(...,DIR,OUT,...).

Buffer Allocation for an Upgrade Set

The buffer allocation is always two data buffers and one index buffer. You cannot influence buffer allocation for the upgrade set.

Miscellaneous Notes on Buffer Allocation

- Data and index buffers are acquired and allocated only at open time. Buffer space is freed at close time.
- Buffer space is aligned on page boundaries. Data buffers are allocated first, then the index buffers.
- Writing a buffer does not free buffer space. The CI is still in storage, so if you again reference that CI, VSAM does not reread the CI. Because VSAM checks to see if the CI is in storage, processing directly in a limited key range may increase throughput if extra data buffers are provided.
- The POINT macro does not cause read ahead because its purpose is to position for subsequent sequential retrieval. It fills only one data buffer.
- When processing *directly*, VSAM reads only one data CI. It does not reread data or index CIs if they reside in storage, except when SHAREOPTIONS(4) is specified. VSAM will immediately write a data buffer if PUT (UPD,DIR) or PUT (NUP,DIR) was issued. VSAM will write immediately for a sequential PUT if PUT (SEQ) follows GET (DIR) for the same RBA.
- Although VSAM does not read index buffers ahead, the effect is similar. Index buffers are loaded when referenced. If multiple index buffers are provided, index CIs are not reread because there is room for the CIs in storage. VSAM reuses buffers on a least-recently-used basis.
- For SHAREOPTIONS(4) processing, VSAM usually reads data and sequence-set CIs on each request. Exceptions are:
  - Consecutive retrievals, not for update, from the same CI do not cause a reread in sequential or skip-sequential mode.

Consecutive inserts or retrievals for update, in sequential or skip-sequential mode, do not cause rereads, unless the SHAREOPTIONS(4) lock has been held for a period longer than approximately 0.5 seconds. (The SHAREOPTIONS(4) lock is for a control area.)

High level index CIs are not reread unless they have gotten out of date.

Read-ahead is not done under SHAREOPTIONS(4); therefore extra data buffers are of no benefit.

Multiple Volume Support

The records of a key-sequenced file, including alternate indexes, 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. Each portion of a multivolume file can be on a separate volume. Each key range of a file, as well as the end of the file, is preformatted.

Multiple volume support is affected by the following DEFINE parameters: VOLUMES, ORDERED|UNORDERED, CYLINDERS|RECORDS|TRACKS|BLOCKS, and KEYRANGES.

The first allocation made on every volume is always the primary allocation. To place the index and data on separate volumes, specify the VOLUMES parameter for both data and index components.

Your use class specification in the DEFINE command can affect suballocation. Refer to "Data Space Classification" in this chapter for further information.

Suballocation when no Key Range is Specified

Primary space is acquired from the first volume at define time. If VSAM needs more space during loading or processing of the file, and if secondary allocation was specified, VSAM uses the secondary extents on the first volume. When VSAM has acquired all the secondary space it can on the first volume and still needs more space, then primary space from the second volume is acquired. If more space is needed, secondary space is acquired on the second volume.

If no secondary allocation is specified, the file cannot be extended.

Suballocation when a Key Range is Specified

Primary space is acquired from each volume at define time. Each key range is assigned to a volume. There is a primary allocation for each key range. If there are fewer volumes than key ranges, the extra key ranges are grouped together on the last volume. If there are more volumes than the number of key ranges, the excess volumes become overflow volumes. A key range is associated with the primary allocation volume and can extend to any overflow volumes.

A key range is extended first by acquiring secondary extents on its volume of primary allocation, next by acquiring primary allocation on the first overflow volume, then secondary extents on the first overflow volume. Primary allocation is then acquired on the second overflow volume, followed by acquiring secondary extents on the second overflow volume. If there is not enough room on an overflow volume to acquire primary space for that key range, VSAM does not acquire any secondary space for that key range. VSAM just skips that overflow volume and goes to the next overflow volume to try to obtain primary space.

VSAM searches for space on volumes in the order they were specified in the VOLUMES parameter. This does not mean that the volumes are allocated or suballocated in that order; that depends upon whether ORDERED or UNORDERED was specified.

UNORDERED

If no key range was specified:
UNORDERED means VSAM must find a primary allocation (or the DEFINE command will fail), but not necessarily on the first volume listed in the VOLUMES parameter. If there is no room for a primary allocation on the first volume, successive volumes are checked for primary space.

If key range was specified:
UNORDERED means that VSAM must find room for a primary allocation for each key range, but not necessarily the first key range on the first volume, the second key range on the second volume, etc.

ORDERED

ORDERED means VSAM must suballocate space on the volumes in the order in which the volumes are listed in the VOLUMES parameter.

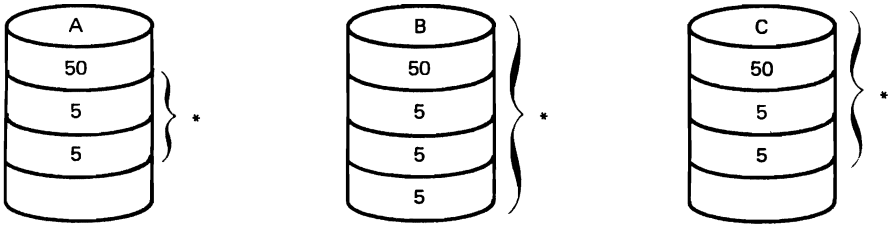
If secondary allocation is specified, space for a component can be expanded to include a maximum of 123 extents. Each primary and each secondary allocation can be made up of up to five non-contiguous areas (extents).

Allocation of Space on Multiple Volumes

The following examples show various combinations of ORDERED|UNORDERED, VOLUMES, and secondary vs. no secondary allocations.

Example 1:

```
VOLUMES(A B C)
ORDERED
CYLINDERS(50 5)
SUBALLOCATION
```



*-extended at execution time

Figure 9-6. Example 1.

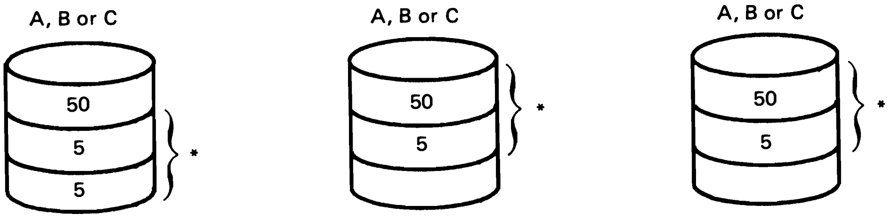
Volume A is the primary volume; volumes B and C are overflow volumes. Fifty cylinders of primary space must be available on volume A, or the DEFINE command will fail.

If the file is extended, a 5-cylinder secondary allocation is made on volume A, if volume A has enough available VSAM space of the required class. Otherwise, an allocation of 50 cylinders (primary amount) is made on volume B. If volume B does not have enough data space for this allocation, the request for extension is rejected.

If volume B has 50 cylinders for allocation (primary amount) and the file needs to be extended further, secondary allocations are made from volume B. Volume B must have enough space available of the required class. Otherwise, a 50-cylinder allocation is made on volume C.

Example 2:

```
VOLUMES(A B C)
UNORDERED
CYLINDERS(50 5)
SUBALLOCATION
```



*-extended at execution time

Figure 9-7. Example 2.

Fifty cylinders of primary allocation must be made on one volume. It may be either A, B, or C. If all 50 cylinders cannot be allocated on a single volume, the DEFINE fails.

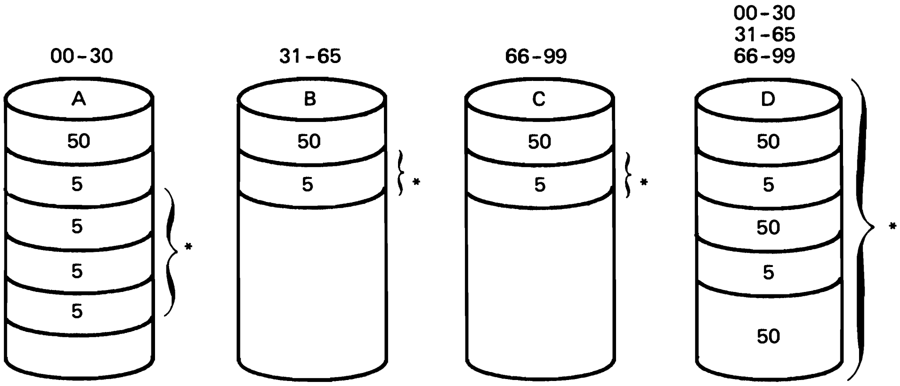
Volumes are searched in the order specified. If both A and B have 50 cylinders available, allocation is made on A because it was specified first.

When the file is extended, VSAM attempts to make the 5-cylinder secondary allocations on the same volume the primary allocation was made on. This continues until all data space of the required class is used.

To further extend the file, VSAM searches the volumes for space in the same order specified for primary allocation. If VSAM cannot acquire the primary amount of space (50 cylinders), an error code is issued.

Example 3:

```
VOLUMES(A B C)
KEYRANGES((00 30) (31 65) (66 99))
ORDERED
CYLINDERS(50 5)
SUBALLOCATION
```



*-extended at execution time

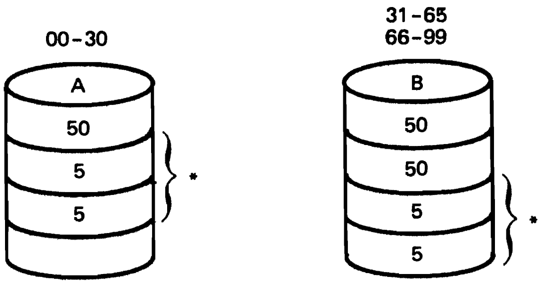
Figure 9-8. Example 3.

A primary allocation of 50 cylinders is made for each key range. The first key range is on volume A, the second on volume B, the third on volume C. If 50 cylinders cannot be allocated on each volume, the DEFINE fails. The 5-cylinder secondary allocations are made as needed.

A key range can be extended only on the volume it occupies or on an overflow volume. If volume D were added to the VOLUMES list, all key ranges would be extended on volume D (first a primary allocation amount of 50 cylinders for a key range on volume D, then secondary allocations of 5 cylinders) if the appropriate volume initially assigned to the key range is full.

Example 4:

```
VOLUMES(A B)
KEYRANGES((00 30) (31 65) (66 99))
ORDERED
CYLINDERS(50 5)
SUBALLOCATION
```



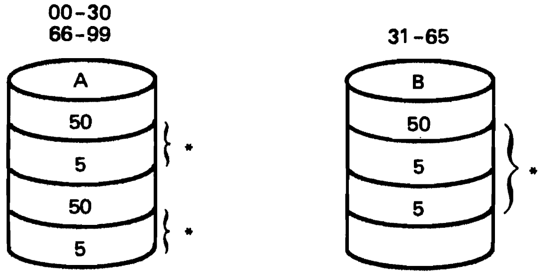
*-extended at execution time

Figure 9-9. Example 4.

If only volumes A and B are specified, the first key range is allocated on volume A, and the second and third key ranges are allocated on volume B. Volume A has one 50-cylinder primary allocation, and volume B has two 50-cylinder primary allocations. This can occur only for a file with the SUBALLOCATION attribute specified. If both UNIQUE and KEYRANGES are specified, each key range must reside on a separate volume.

Example 5:

```
VOLUMES(A B A)
KEYRANGES((00 30) (31 65) (66 99))
ORDERED
CYLINDERS(50 5)
SUBALLOCATION
```



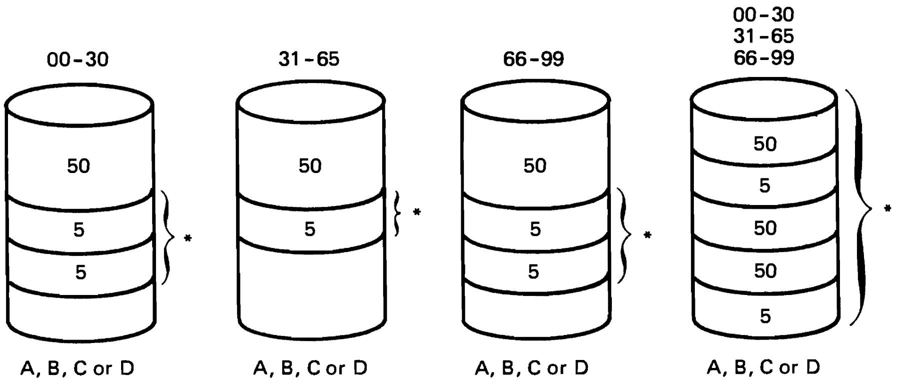
*-extended at execution time

Figure 9-10. Example 5.

A primary allocation of 50 cylinders is made for each key range. The second key range is on volume B; the first and third key ranges are on volume A. This can occur only for a file with the SUBALLOCATION attributed specified. If both UNIQUE and KEYRANGES are specified, each key range must reside on a separate volume.

Example 6:

```
VOLUMES(A B C)
KEYRANGES((00 30) (31 65) (66 99))
UNORDERED
CYLINDERS(50 5)
SUBALLOCATION
```



*-extended at execution time

Figure 9-11. Example 6.

A primary allocation of 50 cylinders is made for each key range. VSAM attempts to put one key range on each volume. If volume A does not have 50 cylinders available, the first key range is put on volume B, and the second and third key ranges are put on volume C. If neither A nor B has 50 cylinders, all three key ranges are placed on volume C.

VSAM first extends a key range on the volume it is on before trying to extend it on any overflow volume. If volume D were added to the VOLUMES list, each key range would be extended on volume D, if no more spaces were available on the volume of its primary allocation.

If volume D were listed in the VOLUMES parameter, it would not necessarily be an overflow volume. If 50 cylinders of primary allocation were available on A, B, and C, then D would be an overflow volume. If A doesn't have 50 cylinders available, but B, C, and D have 50 cylinders each, the first key range is put on volume B, the second on volume C, and the third on volume D. Volume A becomes the overflow volume.

Exercise:
You have a 600-cylinder file that you want to reside on two 3330 volumes, with 400 cylinders on volume A and 200 cylinders on volume B. How do you specify this allocation requirement in the DEFINE command?

Do *not* specify `VOL(A B)`
`CYL(600)`

This request would be rejected because the amount of primary space to be allocated on each volume is greater than that available on one volume (404 cylinders on a 3330).

Do *not* specify `VOL(A B)`
`CYL(400,200)`

This request would obtain 400 cylinders of primary allocation on volume A and 400 cylinders of primary allocation on volume B.

Do specify `VOL(A B)`
`CYL(200,200)`

This request obtains 200 cylinders primary allocation on volume A,
200 cylinders secondary allocation on volume A,
200 cylinders primary allocation on volume B.

The mounting requirements with multiple volumes are simple. All volumes must be mounted (except with sequential KSDS, ESDS, and RRDS). A primary allocation amount will be acquired on *every* volume.

Allocation

The CYLINDERS|RECORDS|TRACKS|BLOCKS parameters of the DEFINE command determine how VSAM allocates space. Considerations in choosing allocation parameters are:

- You may specify allocation at the CLUSTER/AIX level, DATA level, DATA and INDEX levels, and CLUSTER/AIX and DATA levels.
- If you specify allocation at the CLUSTER/AIX level only, the amount needed for the index is subtracted from the specified amount. The remainder of the specified amount is assigned to data.
- If you specify allocation at the DATA level only, the specified amount is assigned to data. The amount needed for the index is in addition to the specified amount.
- If you specify allocation at both the DATA and INDEX levels, the specified data amount is assigned to data, and the specified index amount is assigned to the index.
- If you specify *secondary* allocation at the DATA level, secondary allocation must be specified at the INDEX level unless you specify allocation at the CLUSTER level.
- A control area can never cross an extent boundary. A cluster extent consists of a whole number of CAs.
- A CA is never larger than one cylinder (CKD) or one max-CA (FBA). Optimum performance is obtained when an integral number of CAs occupy a cylinder (or max-CA).
- Access Method Services checks the smaller of primary and secondary space allocation values against the specified device's cylinder (or max-CA for FBA devices) size. If the smaller quantity is greater than the device's cylinder (or max-CA) size, the CA is set equal to the cylinder (or max-CA) size. If the smaller quantity is less than or equal to the device's cylinder (or max-CA) size, the size of the CA is set equal to the smaller space quantity. For FBA, this value is then rounded up to a multiple of min-CA size.

For example:

```
CYL(5 10)   results in a 1-cylinder CA;
TRK(100 3)  results in a 3-track CA;
REC(2000 5) results in a 1-track CA (assuming 10 records
            per track--minimum CA is 1 track);
TRK(3 100)  results in a 3-track CA;
```

For a device with 32 blocks per min-CA and 352 blocks per max-CA:

```
BLK(444 365)  results in a 352-block CA;
BLK(350 210)  results in a 224-block CA;
BLK(96 20)    results in a 32-block CA.
```

For CKD to force Access Method Services to select cylinder CAs, specify CYLINDERS or UNIQUE. When defining using the RECORDS|TRACKS parameters, specify the smaller of primary or secondary allocation as a value of at least one cylinder.

- If you specify IMBED, the data allocation includes the sequence set. More room must be given for data allocation than if you specify (or default to) NOIMBED.
- If you specify secondary allocation, space for a component can be expanded to a maximum of 123 extents (if there is sufficient data space) with a limit of 16 extents per volume if REUSE is specified.
- A UNIQUE file can have a maximum of 16 extents per volume, but it cannot be extended; no secondary allocations are permitted for UNIQUE files.
- A spanned record cannot be longer than a CA minus the control information (10 bytes per CI). Don't specify large spanned records with small primary or secondary allocation.
- VSAM acquires space in increments of control areas. For example, if the allocation amount is 20 tracks and the device is a 3330, the CA size is one cylinder; two cylinders of space (two CAs) are allocated (a 3330 has 19 tracks per cylinder).
- LISTCAT gives information in increments of control area size. If you specify either TRACKS or RECORDS and the allocation is less than one cylinder, LISTCAT reflects the allocation as TRACKS. If the specification results in a one-cylinder CA, LISTCAT reflects the allocation as CYLINDERS. If you specify BLOCKS, the allocation is given in multiples of blocks.

NOALLOCATION

NOALLOCATION allows you to define a file into a catalog without suballocating any space to it. This parameter can be useful in two different ways:

- Creating default models. (See Chapter 6 for a discussion of default models.)
- Creating "dynamic" files for which space is not actually suballocated until the file is opened.

Formerly, files that were used for brief periods of time (for example, workfiles) occupied disk space from the time they were defined until they were deleted. If they were required again, they had to be redefined.

Using the DEFINE CLUSTER command with NOALLOCATION and REUSE parameters makes it possible to define a file for which no space is suballocated until the file is to be opened; this file is called a "dynamic" file. The catalog entry for a dynamic file contains only the allocation *size* specified at define. Information about the suballocated space is added to the catalog when the file is opened.

When you try to delete a dynamic file, VSAM determines if space is currently allocated to it. If it is, VSAM deletes it as if it were a normal VSAM cluster. If space is not allocated, only the catalog entry of the file is removed.

Dynamic files may be entry-sequenced (including SAM ESDS supported by the VSE/VSAM Space Management for SAM Feature), key-sequenced, or relative-record files.

Restrictions

The following restrictions apply to dynamic files:

- A path (but not an alternate index) may be built over a dynamic file, except for a SAM ESDS.
- A dynamic file that does not have space allocated to it cannot be printed (PRINT), copied (REPRO), or exported via EXPORT. EXPORT only supports non-empty dynamic files, but EXPORTRA is valid for an empty dynamic file. To allow catalog recovery, you can EXPORTRA an empty dynamic file or model. Dynamic files are recorded in the CRA of the first volume of the volume list associated with the index component (KSDS, AIX) or data component (non-indexed clusters).
- A default model cannot be opened. If you specify NOALLOCATION, you must also specify REUSE if you plan to open the file.
- The CYLINDERS, TRACKS, USECLASS, etc. parameters normally control space allocation, but for noallocation models (other than reusable files) these attributes are recorded only for modeling purposes.
- If you specify the VOLUMES parameter when you define a file as NOALLOCATION, VSAM records those volumes in the catalog as candidate volumes.
- The NOALLOCATION attribute exists in the catalog entry, but it cannot be implicitly modeled. It can be explicitly modeled (MODEL parameter of DEFINE).
- You cannot specify NOALLOCATION on the ALTER command.
- You cannot ALTER REMOVEVOLUMES for the recovery (CRA) volume or the last existing volume on the candidate list for NOALLOCATION files.

Data Security and Integrity Options

When considering performance, you must also consider the Data Security and Integrity options you are using. VSAM performance is affected by the following:

- Share options—See "Protecting Shared Data" for more information.
- Write check. If you specify WRITECHECK on your DEFINE command, it means you wish to have your records checked as they are written. After a record is written, it is then read without data transfer to test for a data check condition. If NOWRITECHECK is specified (and this is the default), a record is written but no checking occurs. It follows, therefore, that you will get better performance with the NOWRITECHECK option.
- Speed vs Recovery. If you specify RECOVERY on your DEFINE command, it means that space allocated to the data component is preformatted during initial loading. SPEED means space will not be preformatted. Performance is better during initial loading if you specify SPEED. However, specifying RECOVERY enables you to recover from certain system failures during initial load.

If you specify SPEED in a file's DEFINE command, and a system failure occurs, the file must be deleted, redefined, and reloaded. RECOVERY is only useful if you have a recovery procedure that allows you to resume loading the file after a system failure. RECOVERY formats each CA before loading records into it. It allows you to find the software end of file if an abnormal termination occurs during initial creation. After the initial creation of the file, RECOVERY is always in effect.

RECOVERY works in conjunction with the Access Method Services VERIFY command. If a system failure occurs before a file is closed (CLOSE or TCLOSE), VERIFY can prevent your having to reload the file by updating the catalog with the current high RBA. This ensures that your data will not be overwritten inadvertently at a later time, and that you may continue the load at the point of interruption (load-extend). If the SPEED option was in effect while the file was being loaded, VERIFY cannot help because no preformatting was done and no high RBA exists until the file is closed.

Distributed Freespace

Freespace can occur in files either as a result of the DEFINE specification, or due to a CI/CA split. Later in this section are examples of CI and CA splits resulting from record inserts during both sequential and direct processing.

Freespace can be specified for a KSDS or alternate index only. CI freespace should be as large as the design insertion level. Determine the freespace required by estimating the percentage of additions to be made between file reorganizations. If there are to be no additions, or if records will not be lengthened, there is no need for freespace.

Loading a File

- You specify free space for both the control interval and the control area as a percentage of the total space for the respective unit. For example, FREESPACE (20 10) indicates that 20% of the space in each control interval is to be initially empty and 10% of each control area is to be initially empty. If you specify the minimum free space (1 1), you will be given enough free space for one logical record in each control interval and free space for one control interval in each control area. The system default for free space is (0 0).
- Freespace may be altered after the file is loaded. To take full advantage of mass insertion, ALTER freespace to (0 0) after the load.
- If additions will occur only in a specific part of the file, load those parts, which will not be added to, with a freespace specification of (0 0). Alter the freespace to (n n) to load those parts of the file that will receive the additions. If SPEED is specified, it is in effect for loading the initial portion only. Any subsequent portions are loaded with RECOVERY, regardless of the DEFINE specification.
- If additions will occur throughout the file, but will be unevenly distributed, specify a small amount of freespace when you define the file. Then increase the percentage after loading the file. As new CIs and CAs are required, they will be created with the increased freespace specification. Additional splits (after the first split) in the part of the file with the most growth will be minimized. CIs that have little or no growth will contain only a small amount of unneeded freespace.
- If there will be few additions to the file, consider a freespace specification of (0 0) for loading the file and subsequent processing. When records are added, new CAs will be created to provide room for additional insertions. In this case, unused freespace will not be provided.
- Specify freespace that is at least as large as one record and preferably large enough to hold multiple full records.

- For *direct* insertions, make the CI freespace larger than the CA freespace.

- The greater the freespace specification, the more DASD space is required. For *sequential* processing, more I/O operations (with more system overhead) are required to process the same number of records. A bad combination of CI size/record size/freespace can cause poor sequential performance if much of the freespace is unusable.

- Too much freespace could increase the number of index levels, which could increase run time for *direct* processing.

- Too little freespace can cause an excess of (time-consuming) CI/CA splits. After a split, extra time is required for *sequential* processing because the records are not in physical sequence. For *direct* processing, CA splits can increase seek time. Another factor is the additional VSAM overhead required to do the split. Use LISTCAT or the ACB JRNAD exit to monitor CA splits; reorganize the file when they become prevalent. If insertions are truly random, ideally all CAs would split at approximately the same time.

- Records are loaded or mass inserted at the end of a CI until the freespace threshold is passed. The freespace threshold is the point at which freespace becomes less than the amount specified in the DEFINE command.

- VSAM ensures that at least one record (or one segment of a spanned record) is placed into a CI. Also if the CA freespace specified in the DEFINE command is not zero, but less than one CI, the result is one free CI in the CA.

- If a CI can contain four logical records and (25 0) freespace is specified, the CI would contain three logical records and 25% freespace. If (20 0) freespace is specified, the result is three logical records and 25% freespace. If (33 0) freespace is specified, the result is two logical records and 50% freespace. If (80 0) freespace is specified, the result is one logical record and 75% freespace.

- Remember that a CI contains logical records, freespace, and control information (CIDF and RDFs). A 4K CI cannot contain four 1K logical records. A 4K CI with (25 0) freespace specified will contain at least 1K of freespace; only two 1K fixed length logical records could be loaded into the CI. Only one more 1K logical record could be added before a CI and/or CA split would be required.

- If ten CIs fit into one CA and (0 5) freespace is specified, the CA will have one free CI.

- When using the ERASE macro, the deleted record's space is returned to the freespace. Sequence set entries are not changed at the time of the erase. If a CI is emptied by ERASE, it can be reclaimed later as a free CI if it is needed.

Note: Space that becomes free within a control interval because of some (but not all) records being deleted or shortened can remain unused even though the space is available. This situation occurs in the cases where new records to be added to the file do not have key-field values that match the range of the freed area within the control interval. For example, a record with key-field value 250 cannot be inserted between records with key-field values 22 and 70. You may (depending on the amount of unuseable space) have to reorganize the file to make the available free space useable.

CI/CA Splits

The rules for CI and CA splits are as follows.

Sequential processing

CI split: If the insert is in the middle of the CI, the records with higher keys are moved to the free CI. The insert and the records with lower keys remain in the old CI. If the insert is at the logical end of the CI, the inserted record goes to the free CI.

CA split: If the insert is *not* in the last logical CI, all CIs after the split CI are moved to the new CA. If the insert is within the last logical CI, that CI is moved to the new CA. If the insert is at the end of the last logical CI, the inserted record is placed into the new CA.

Direct processing

CI split: Half the records (those with the higher keys) in the CI are moved into the new CI. The new record is inserted (in key sequence) into the CI to which it belongs.

CA split: Half the CIs (those with the higher keys) are moved to the new CA. Insertion then occurs through regular CI split processing, using the newly-created freespace CIs.

Updates can cause CI/CA splits when:

- The record length is increased, and there is not enough freespace in the CI; or
- The record length is decreased and additional RDFs are required. If the space required for the RDFs is more than the amount by which the record is shortened, and there is insufficient free space, the CI must be split.

Following are some examples of CI/CA splits.

Figure 9-12 shows the CA after direct insertion of records 025 and 101.

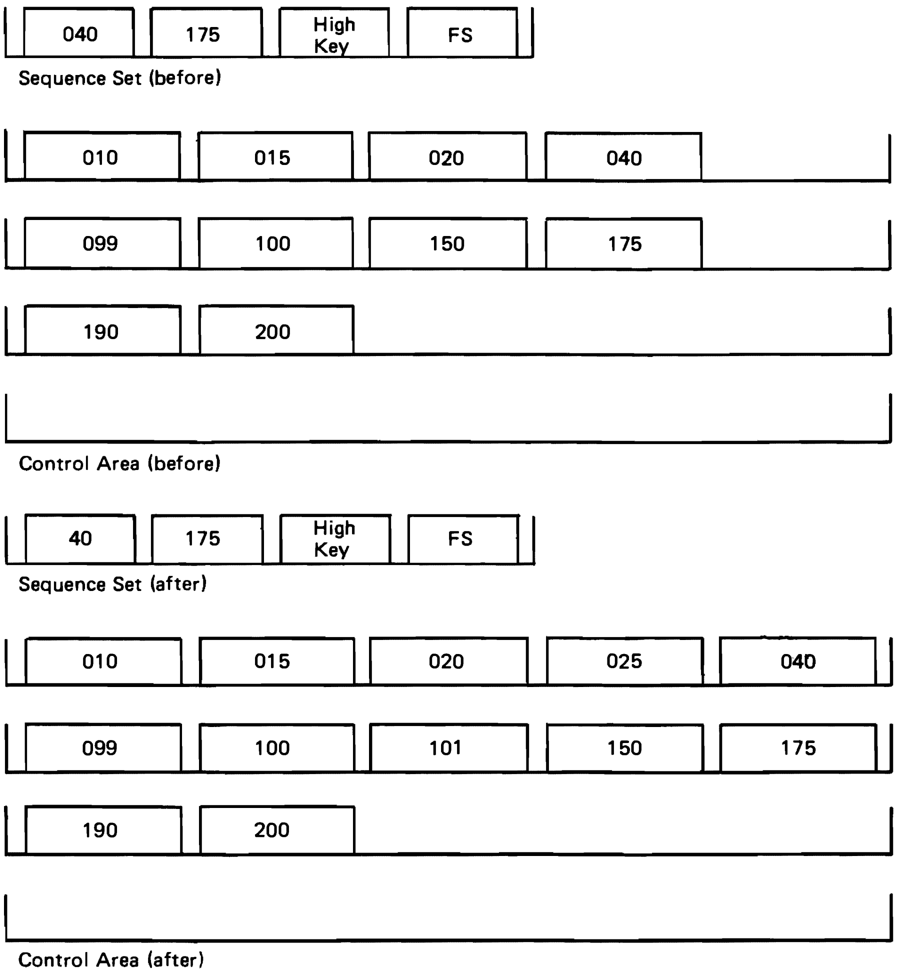


Figure 9-12. CA After Direct Insertion of Records 25 and 101.

Figure 9-13 shows the CA after direct insertion of record 026, causing a CI split.

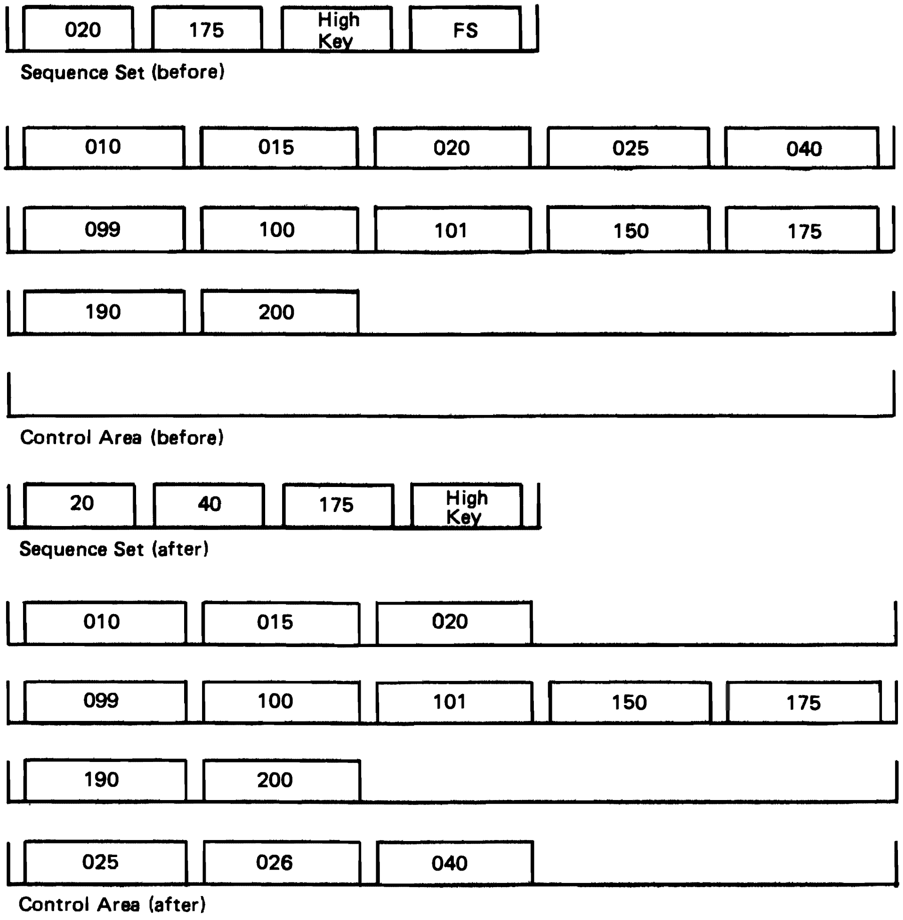


Figure 9-13. CA After Direct Insertion of Record 26.

Figure 9-14 shows a CA split and CI split caused by the direct insertion of record 168.

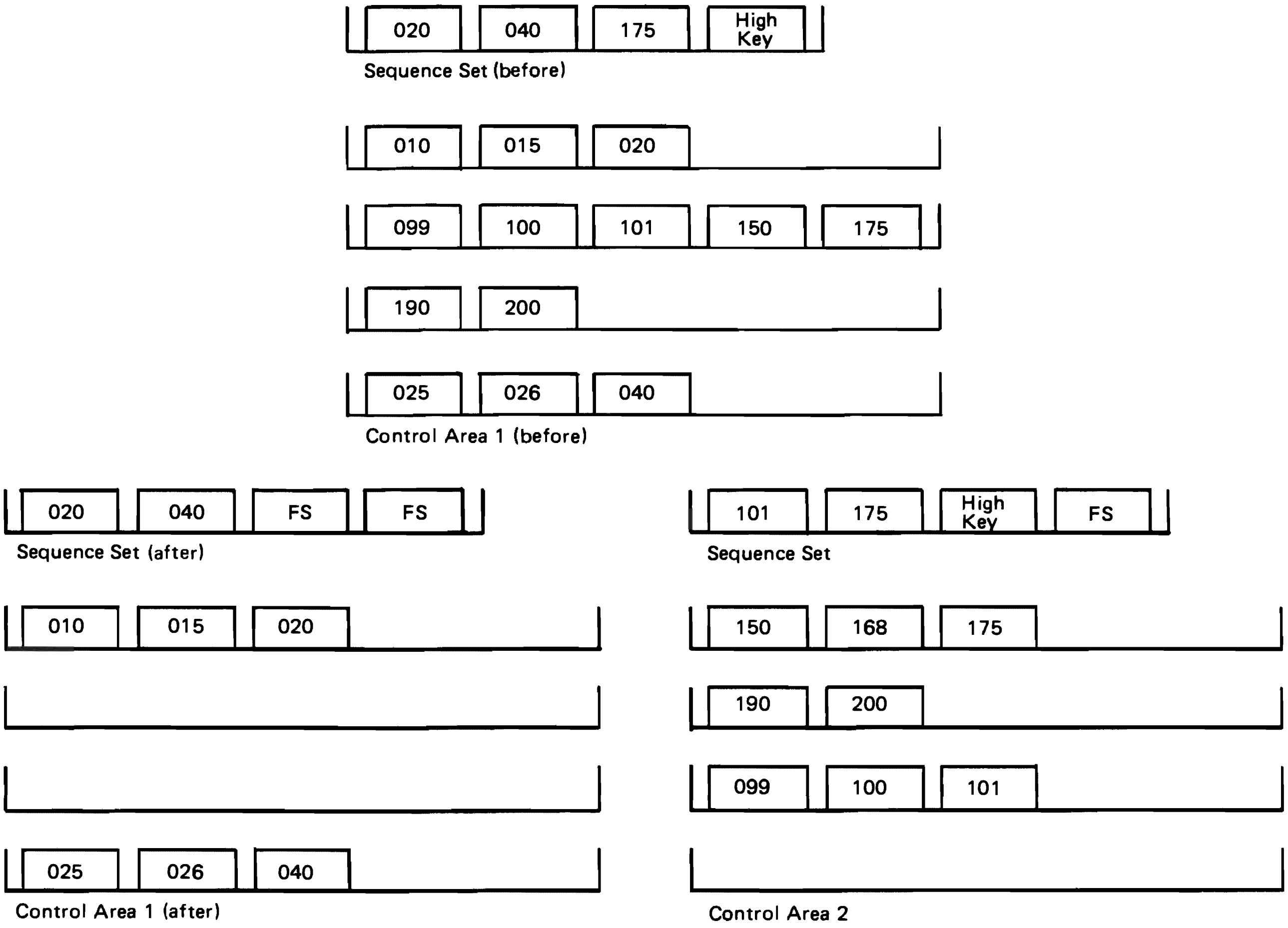


Figure 9-14. CA Split and CI Split After Direct Insertion of Record 168.

Figure 9-15 shows the CA after sequential insertion of records 20 and 101.

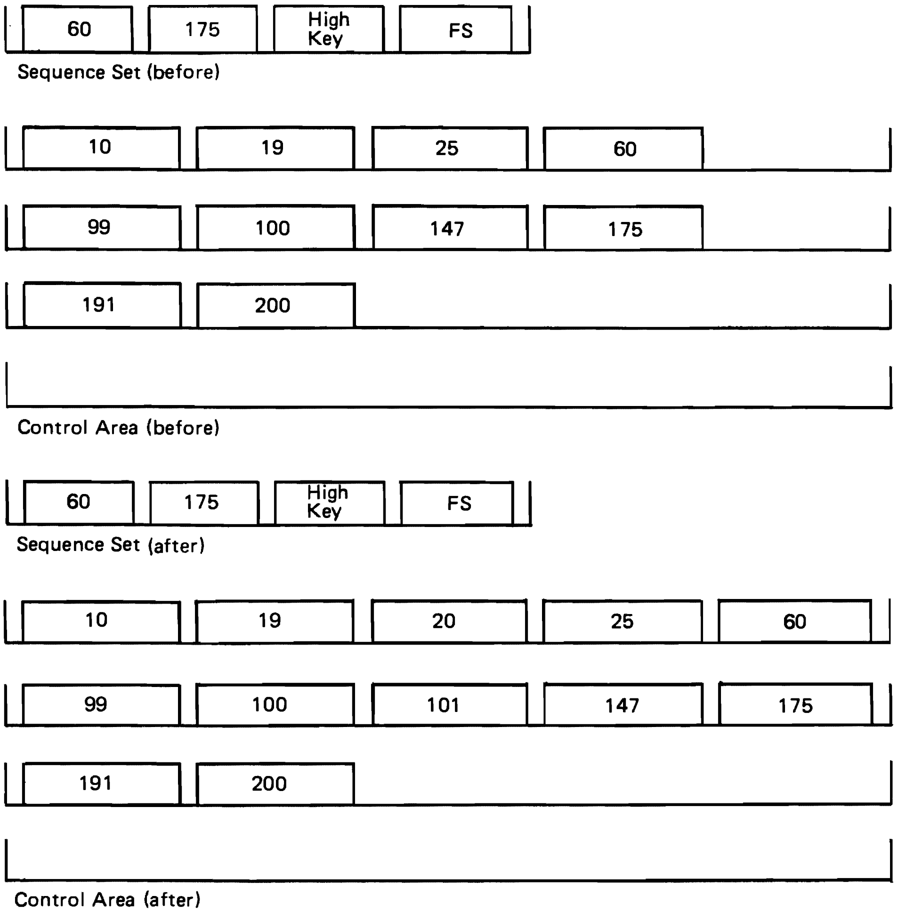


Figure 9-15. CA After Sequential Insertion of Records 20 and 101.

Figure 9-16 shows the CA after sequential insertion of records 12, 13, and 14. Record 12 causes a new CI split. Note that the key associated with the old CI is one number less than the low key in the new CI. This permits mass insertion to take advantage of the newly-created freespace.

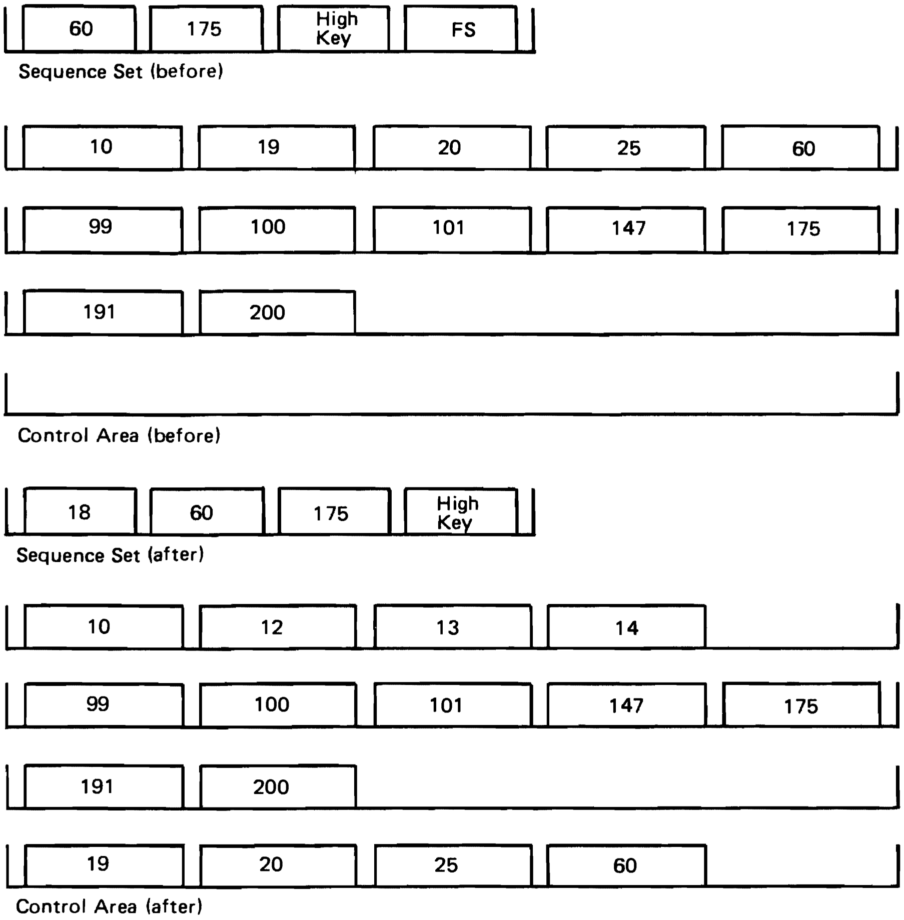


Figure 9-16. CA After Sequential Insertion of Records 12, 13, and 14.

Figure 9-17 shows a CA split and CI split caused by the sequential insertion of record 144. Note that the key associated with the old CI is one less than the low key in the new CI. This permits mass insertion into the newly-created freespace.

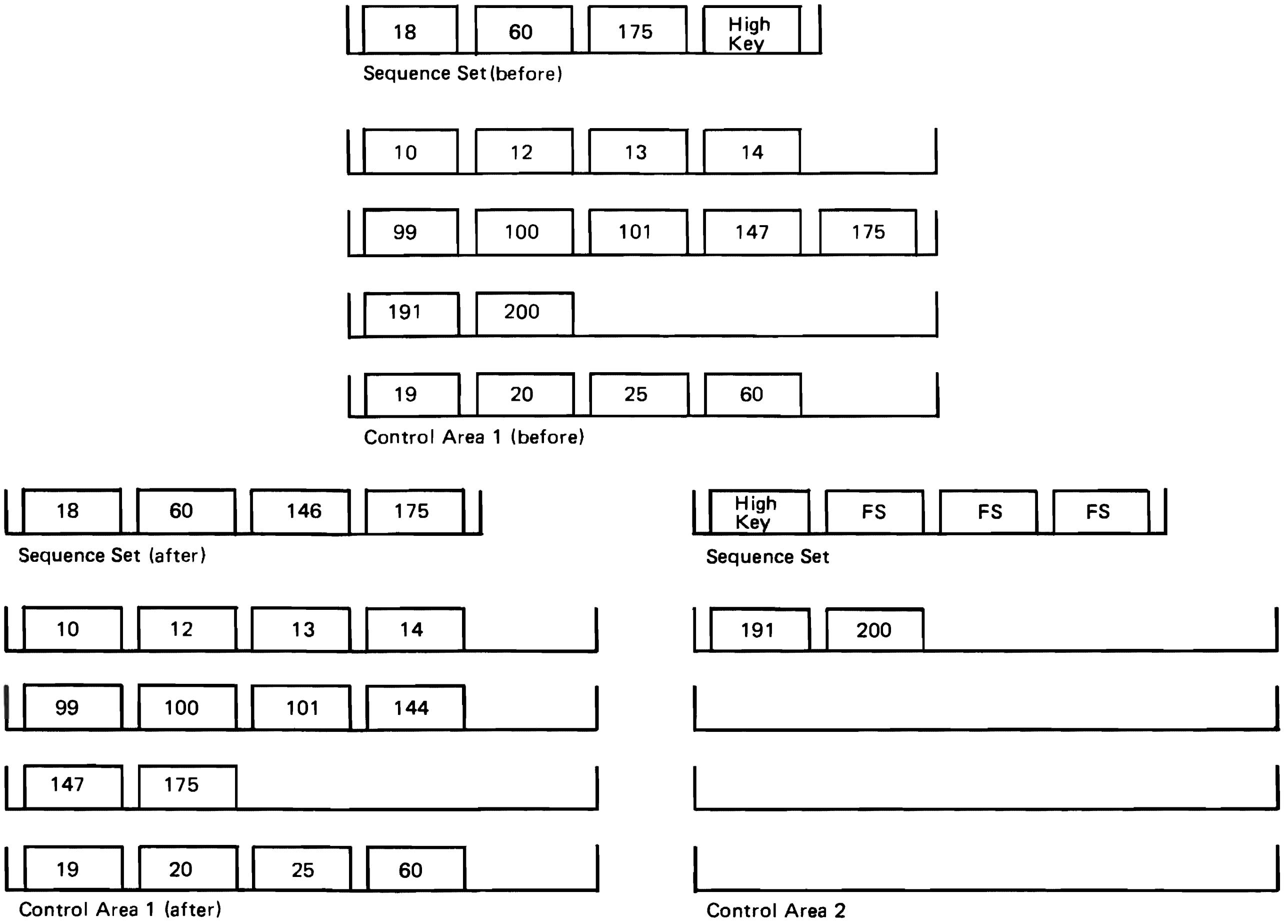


Figure 9-17. CA Split and CI Split After Sequential Insertion of Record 144.

Figure 9-18 shows a CA after a sequential insertion of records 205, 210, 223, and 228, during load extend processing. Note that the freespace is preserved.

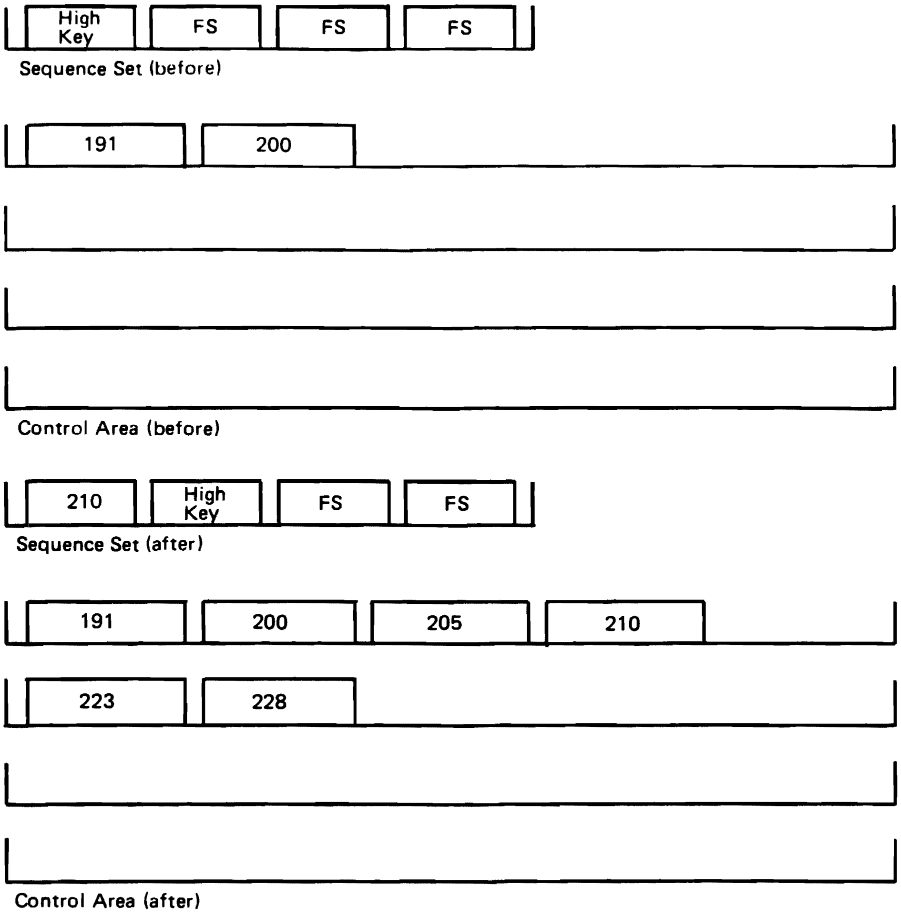


Figure 9-18. CA After Sequential Insertion of Records 205, 210, 223, 228.

Index Options

The following options influence performance and storage requirements for the use of the index of a key-sequenced file or alternate index:

- Number of index records in virtual storage
- Index and data on separate volumes
- Sequence set records adjacent to control areas
- Replication of index records

Number of Index Records in Virtual Storage

For keyed access, VSAM needs to examine the index of a file, and it is obvious that performance improves if a large number of index records can be held in virtual storage.

Before the processing program begins to process the file, it must specify, either explicitly or by default, the amount of space VSAM can use to buffer index records. Enough space for one index record is the minimum; but, when the space is large enough for only one or two index records, an index record may be continually deleted from virutal storage to make room for another and then retrieved again later when it is required anew. Ample space in which to buffer index records can improve performance by preventing this situation, provided that the buffer allocation does not cause excessive paging by VSE. Remember that VSAM searches only the sequence set for sequential access but every index level for direct access.

You can ensure that an acceptable number of index records will be in virtual storage by specifying enough space for I/O buffers for index records through the BUFFERSPACE parameter of the DEFINE CLUSTER command for a file, or through the BUFNI and BUFSP parameters of the ACB, or through the BUFSP parameter of the DLBL statement when you begin to process the file. VSAM keeps as many index set records in virtual storage as the space will hold.

Whenever an index record must be retrieved to locate a record, VSAM makes room for it by deleting another index record from the space.

Index and Data on Separate Volumes

When you define a key-sequenced file or alternate index, you can place the entire index or the index set alone on a separate volume from the data, either on the same or on a different type of storage device. Data and index of a cluster (file) are defined separately and the volume that is to contain each is specified in the VOLUMES parameter of the DEFINE command. Only the index set is placed on the separate volume if the sequence set is imbedded with the data as described below.

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

Sequence Set Records Adjacent to Control Areas

When you define a key-sequenced file, alternate index, master catalog, or user catalog, you can specify that the sequence set index record for each control area is to be on the first min-CA (track) of the control area. This reduces disk-arm movement because it is not necessary to do separate seeks to locate both the sequence set index record and the data record. One arm movement enables VSAM to retrieve or store both the index record and the contents of the control interval in which the data record is stored. When this option is used, sequence set records are *replicated* to reduce rotational delay (that is, as many copies of the sequence set as will fit on the min-CA (track) are written on the min-CA (track)).

Figure 9-19 illustrates replication of a sequence set record that has been placed adjacent to its control area; the advantages of this option are further discussed below.

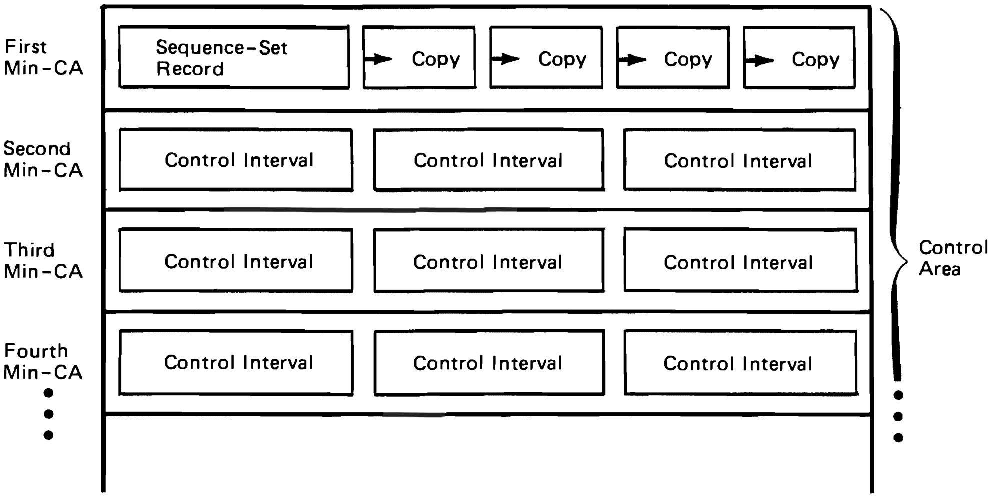


Figure 9-19. Sequence Set Record Imbedded (For CKD devices, a min-CA is equal to a track.)

Replication of Index Records

You can specify that each index record be replicated (written on a) min-CA (track) of a direct-access volume as many times as it will fit). Replication reduces the time lost waiting for the index record to come around to be read (rotational delay). Average rotational delay for a nonreplicated index record is half the time it takes for the volume to complete one revolution. Replication of a record reduces this time; for example, if ten copies of an index record fit on a min-CA (track) of a 3330, average rotational delay is only one-twentieth of the time it takes for the volume to complete one revolution.

This option costs direct access storage space; it requires a full min-CA (track) of storage for each replicated index record. If the entire index set is not being held in storage and there is significant direct processing, replication is a good choice. To decide whether to replicate, you have to weigh the relative values of direct access storage space and processing speed. You should also consider the DASD device involved. Some devices such as the 3340 have more than one logical min-CA (track) to a physical min-CA (track). Consequently, replication will not reduce rotational delay as much on a 3340 as it will on a 3330.

After you have defined a file with replication, you should ensure that the index is not allocated on any defective tracks or blocks, in which case alternate tracks or blocks would be assigned. The assignment of alternate tracks or blocks defeats the gain of replication.

The possible combinations of sequence set and index set are:

- Sequence set records separated from index set records and only sequence set records replicated (IMBED, NOREPLICATE).
- Sequence set records separated from index set records, and both sequence set records and index set records replicated (IMBED, REPLICATE).
- Sequence set records and index set records together, and all index set records replicated (NOIMBED, REPLICATE).

- Sequence set records and index set records together, and no index set records replicated (NOIMBED, NOREPLICATE)

In summary, IMBED reduces seek time when processing both *directly* and *sequentially*. REPLICATE has its greatest effect when insufficient buffer space is available to hold the index set during *direct* processing.

Imbedded Index Records

The IMBED option places the lowest level of index physically adjacent to the data it references. This resuts in less seek time than would result when not using the IMBED option. The imbedded index CI is also replicated on the first min-CA of the control area it references. That is one min-CA of each control area is used for the index. In some situations, this may be judged as too much space for index in relation to the data. For example, assuming a control area size of one cylinder on a 3340, this would be 1/12 of the data space, whereas on a 3330, it is only 1/19 of the data space.

On a 3340, it may be better to separate the index from the data, placing them on different volumes, and specifying NOIMBED. In this case, also specify NOREPLICATE because the space saved by not imbedding would not be realized if REPLICATE were chosen. Because of the physical characteristics of the 3340, replication is less helpful than for a 2314, 3330, 3350, 3310, or 3370.

Specifying IMBED reduces seek arm movement during *direct* or *sequential* processing.

Performance Measurement

VSAM keeps statistical information about a file in its 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.

You can list the entire catalog record, the statistics, and the parameters selected when the file was defined, by using the LISTCAT command. (See "Appendix B: Interpreting LISTCAT Output" in *Using VSE/VSAM Commands and Macros* for an explanation of the output produced by the LISTCAT command.) You can use the SHOWCB and TESTCB macros in a processing program to display or test one or more file statistics and parameters. These statistics and parameters include:

- Control interval size
- Percentage of free control intervals per control area
- Number of bytes of available space at the end of the file.
- Length and displacement of the key
- Maximum record length
- Number of buffers
- Number of records*
- Password
- A timestamp that indicates if either the data or the index has been processed separately
- Number of levels in the index
- Number of extents
- Number of records retrieved, inserted, deleted, and updated*

* VSAM does not update these statistics when a file is processed with user-supplied buffers (MACRF=UBF in the ACB). See the ACB macro for more information.

- Number of control-interval splits in the data and in the sequence set of the index
- Number of EXCPs that VSAM has issued for access to a file

Notes: When a cluster or alternate index is exported, that is, named in an EXPORT command, the statistics are reset in the exported catalog record due to the redefinition of the imported cluster or alternate index in another catalog.

SAM ESDS statistics are not updated when the file is accessed via DTF.

Chapter 10: Data Interchange Considerations

VSE/VSAM Release 2

VSE/VSAM Release 2 provides the following enhancements to previous releases.

DASD Sharing

Compatibility with Previous Releases:
The sharing of VSAM files across 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 devuces 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 SHAREOPTIONS parameter; the first value specified is now used to indicate the way a file can be shared across both partitions and systems in VSE. The second SHAREOPTIONS value is the same as before; it is allowed for compatibility with OS/VS.

The sharing of VSAM files across VSE systems does not affect VSAM file or catalog format and therefore has no impact on OS/VS data compatibility. Note that 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

Compatibility with Previous Releases:
The new SHAREOPTIONS(4) support prevents a SHAREOPTIONS(4) key-sequenced file from being opened for output (MACRF=OUT) 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 SHAREOPTIONS(4) support does not impact file or volume portability to OS/VS.

Catalog Management Performance Improvement

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

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

Access Method Services CANCEL Command

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

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

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, but a volume allocated to VSAM with the DEDICATE parameter can be processed by OS/VS.

Additional Classes of Space

Compatibility with Previous Releases:
Files specifying new space classes may be moved (along with their catalog records) to a previous release of 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

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

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

Disposition Parameters for a File

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

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

Default Models

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

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

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

Compatibility with Previous Releases:
Unpredictable results.

Compatibility with OS/VS VSAM:
Not applicable.

Partition/Processor Independence

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

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

VSE/VSAM Space Management for SAM Feature

Compatibility with Previous Releases:
SAM ESDSs and their catalog entries cannot be processed.

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

VSE/VSAM Release 1

NOIMBED For Catalogs

Compatibility with DOS/VS VSAM: Catalogs defined with the disk-space optimization enhancements (NOIMBED) cannot be processed on DOS/VS VSAM. A conversion process (described below) can be used, however, to allow processing on DOS/VS VSAM.

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.

Converting an Imbedded Catalog

Master or user catalogs defined under DOS/VS Release 34 or earlier releases are constructed with the sequence set imbedded in the data component. To convert an imbedded catalog to a non-imbedded catalog, perform the following steps:

1. Export all files that are to be carried over to the new catalog. Use the EXPORT PERMANENT option to delete the files from the catalog.
2. Delete any other files using the DELETE command.
3. Delete all VSAM data space owned by the catalog by issuing the DELETE SPACE command for all volumes owned by the catalog.
4. EXPORT DISCONNECT any user catalog entries from the master catalog.
5. Clean up the catalog volume by issuing the DELETE MASTERCATALOG|USERCATALOG command.
6. Redefine the catalog, specifying the NOIMBED attribute (in addition to the other options you desire).
7. Redefine all necessary data space with DEFINE SPACE for all applicable volumes.
8. IMPORT all previously exported files.
9. Redefine any nonVSAM and/or user catalog entries.
10. For the master catalog, IMPORT CONNECT any user catalogs that were EXPORT DISCONNECTed.

Control Interval Split Integrity

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

Compatibility with DOS/VS VSAM:
Files on a Fixed Block Architecture device cannot be processed by DOS/VS VSAM. 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 (via EXPORT and IMPORT) from 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 from a non-Fixed Block Architecture device.

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

Data Space Classification Support

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

Conversion of Catalog to Fixed Head Data Space

All data space defined under DOS/VS Release 34 or earlier is considered class-0, with no additional specification necessary. Release 34 or earlier job streams define class-0 space and request class-0.

To assign files from previous releases to classes other than class-0:

- Define the new space with the desired class specification.
- Rename the old file using the ALTER command.
- Define a new file, using the old file name and specifying the desired USECLASS.
- REPRO the old file to the new location (new space class).

You can also use EXPORT/IMPORT to assign new data space classes:

- EXPORT the old file to a portable file (permanent).
- Redefine the space of the required classes.

- IMPORT the file specifying the USECLASS desired as an OBJECTS subparameter.

To change the classification of a previously defined space, you must delete the space and redefine it. If this space contains files, the files may be preserved using EXPORT|EXPORTRA and IMPORT|IMPORTRA.

New Physical Record Size

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

Compatibility with OS/VS VSAM:
A file created by VSE/VSAM with a new physical record size 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, as long as the index CI size is equal to .5, 1, 2, or 4K.

New SHOWCAT Operand

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

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.

DOS/VS VSAM does not support CIMODE processing. Specify RECORDMODE if the portable file will be processed by DOS/VS.

Compatibility with OS/VS VSAM:
Files from OS/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 OS/VS VSAM, automatically take advantage of the EXPORT or EXPORTRA command's I/O buffering performance enhancement.

OS/VS VSAM does not support CIMODE processing. Specify RECORDMODE if the portable file will be processed by OS/VS.

VSAM-VTAM Similarities

The Virtual Telecommunications Access Method (VTAM) is an access method for teleprocessing. There is considerable similarity between the two access methods with regard to control block names and fields, control block manipulation, and general approach to request handling.

Both access methods use an ACB. The VSAM ACB represents the file. In VTAM, however, the ACB essentially represents an application program. Both types of ACBs are objects of the OPEN macro instruction, and VSAM and VTAM ACBs can be opened with one macro instruction.

Both types of ACBs can contain pointers to an exit list. Both VSAM and VTAM exit lists contain addresses of routines to be entered when error conditions occur (LERAD and SYNAD exit routines) and addresses of routines to be entered when special situations occur.

Both access methods follow the same general I/O-request procedure. An I/O macro instruction is issued that indicates an RPL. The RPL in turn contains information about the request, such as the location of the I/O work area or whether the request is to be handled synchronously or asynchronously.

Finally, both access methods use the same macro instructions (GENCB, MODCB, TESTCB, and SHOWCB) to generate and manipulate their respective ACB, EXLST, and RPL control blocks.

To make control blocks unique, a special operand is used when the control block is generated. By specifying AM=VTAM on the ACB, EXLST, or RPL macro instruction, the control block is generated in VTAM form. Omitting this operand causes a VSAM control block to be built. A VSAM control block will also be built if AM=VSAM is specified. If an installation uses both of these access methods, it may be desirable to have AM=VSAM specified in VSAM programs for documentation purposes.

Chapter 11: Data Protection

The protection of data includes *data security*, the safety of data from theft or intentional destruction, and *data integrity*, the safety of data from accidental loss or destruction. All security options and some integrity options are specified for a file and its components in the DEFINE command. Some integrity provisions can be implemented through commands rather than through command options. The protection of data also includes recovery commands and backup tools. Methods and procedures about the use of these tools are included in this chapter.

Data Security

VSAM provides options to protect files against unauthorized use and loss of data. These options, specified when a file or catalog is defined using Access Method Services, include passwords that can be altered, a user-written security verification routine, and controls over file sharing among partitions or among subtasks in a partition.

Passwords to Authorize Access

You can optionally define passwords for clusters, alternate indexes, components (data and index), paths, and catalogs, which a program or operator must provide to gain access to the protected objects. Password levels differ for various degrees of security. These levels are (from low to high):

- *Read access* (READPW parameter). This is the read-only password, which allows you to retrieve data records and catalog entries, but not to add, update, or delete them, nor to see password information in a catalog entry.
- *Update access* (UPDATEPW parameter). This password authorizes you to retrieve, update, add, or delete records in a file. Specifying a catalog's update password authorizes you to define files in it.
- *Control-interval access* (CONTROLPW parameter). This password authorizes you to retrieve and store the entire contents of a control interval (rather than a logical record).
- *Full access* (MASTERPW parameter). This is the master password, which allows you to perform all operations (retrieving, updating, adding, and deleting) on a file and on the catalog entry or any index associated with it. Using this password to gain access to a catalog entry allows you to delete an entire file and to alter any catalog information (including passwords) about data, index, or catalog. The master password allows all operations and bypasses any additional verification checking by the user security verification routine.

Each higher-level password allows all operations permitted by lower levels. Any level may be null (not specified), but if a low-level password is specified, the master level password must also exist. The DEFINE and ALTER commands accomplish this by propagating the value of the highest password specified to all the higher password levels. For example, if you specify only a read-level password, that password becomes the update, control-interval, and master level passwords as well. If you specify a read password and a control-interval password, the control-interval password becomes the master level password as well. However, the update level password is not affected. (It remains null.)

A password, if required, is normally supplied by the processing program in a field pointed to by the ACB or through Access Method Services parameters. If neither of these are supplied, the password must be supplied by the operator.

Two options can be specified in the DEFINE command for use when the operator supplies a password: the ATTEMPTS option and the CODE option.

- The ATTEMPTS option specifies how many times, 0 through 7, the operator can attempt to supply the correct password. If 0 is specified, passwords cannot be supplied by the operator. If ATTEMPTS is not specified in the DEFINE command, the default (2) allows the operator to attempt to supply the password twice.
- The CODE option specifies a one-to-eight character name, other than the name (file-ID) of the file, to which the operator responds with a password. This *prompting code* helps keep data secure by not allowing the operator to know both the name of the file and its password. If the CODE option is not specified, the name of the job and the name (file-ID) of the file are supplied to the operator.

If the processing program omits the password or supplies the wrong password, and the operator cannot supply the correct password in the allowed number of attempts, OPEN is terminated. An error code is set in the ACB indicating that the file cannot be opened because the correct password was not supplied.

Catalogs are, themselves, VSAM files and may have passwords. *If you define passwords for any files in a catalog, you must also define passwords for the catalog in order for the file passwords to have effect.* If you do not define passwords for the catalog, no password checking takes place during operations on the file's catalog entry. For some operations (for example, listing all of a catalog's entries with their passwords or deleting catalog entries), the catalog's passwords may be used instead of the entry's passwords. Thus, if the master catalog is protected, its update or higher-level password is required when defining a user catalog because all user catalogs have an entry in the master catalog. When deleting a protected user catalog, the user catalog's master password must be specified.

Operations on a catalog may be authorized by the catalog's appropriate password or, in some cases, by the appropriate password of the file whose definition in the catalog is being operated on. For example:

- If you want to define a file in a password-protected catalog, you must specify the catalog's update (or higher) password.
- If you want to delete a protected file from a password-protected catalog, you must specify the catalog's or file's master password.
- If you want to alter a file definition in a password-protected catalog, you must specify the catalog's or file's master password.
- If you want to list a file's catalog definition in a password-protected catalog (and the file is password-protected also), you must specify the catalog's or file's read (or higher) password. If you want to list the passwords themselves, you must provide the master password.
- If you want to list a file's catalog definition in a password-protected catalog (and the file is not password-protected), you do not have to specify a password.

Because a user catalog defines itself, it may be password-protected without the master catalog being password-protected. To delete an empty user catalog, you must give its master password, whether the master catalog is password-protected or not.

Some Access Method Services operations may involve more than one password authorization. For example, importing a file involves defining the file and loading records into it. If the catalog into which the file is being imported is password-protected, its update (or higher) password is required for the definition; if the file is password-protected, its update (or higher) password is required for the load. In these cases, the master password of the catalog satisfies both requirements.

Every VSAM file is represented in a catalog by two or more entries: a cluster entry and a data entry or, if the file is a key-sequenced file, a cluster entry, a data entry, and an index entry. Of the two or three entries, the cluster entry is the controlling entry. Each of the two or three entries can have its own set of four passwords; the passwords you assign need have no relationship to one another. One reason for this separate password protection is to prevent access to the index of a key-sequenced file, since an index can be opened independently of the cluster. For example, if you password-protect a cluster but do not password-protect the cluster's data component, another user could issue LISTCAT to determine the name of your cluster's data component, then open the data component and access records in it even though the cluster itself is password-protected.

The following protection considerations and precautions should be observed when using commands that refer to a catalog without using the files defined by the catalog:

- To gain access to passwords in a catalog (for example to list or change passwords), you must specify the master password of either the entry or the catalog. If both the password of the entry and the password of the catalog are supplied, the password of the catalog is used. Similarly, a master password must be specified with the DEFINE command if you want to model the entry's passwords (with the MODEL parameter).
- To delete a protected file entry from a catalog requires the master password of the entry or the master password of the catalog containing the entry. To delete a nonempty VSAM data space, the master password of the catalog is required, if the catalog is password protected; to delete an empty VSAM data space, the update password of the catalog is sufficient. When a catalog entry is created (with a DEFINE command), the catalog's update or higher-level password is required.
- You can list catalog entries that are password protected by specifying the read passwords of the entries or the catalog's read password. You can list unprotected entries without specifying the catalog's read password. If you wish to list the passwords associated with a catalog entry, you must specify either the master password of the entry or the master password of the catalog.
- If the proper password is not specified with an Access Method Services command, a password prompt occurs. Unless you have specified the CODE parameter on either the DEFINE or ALTER command, the prompt includes the *file-ID* of the file; if you specify CODE, the prompt includes the code name you specified. In some circumstances, more than one prompt occurs. For example, when an ALTER or DELETE request is processed, the catalog must be referred to twice, once to locate the information, and again to perform the requested function. Again, incorrect password specification when you want to list catalog entries may cause numerous prompts. To avoid unnecessary prompts, specify

the catalog's master password, which allows access to all entries contained in that catalog.

- Specification of a password where none is required is always ignored.

The following protection considerations and precautions should be observed when using commands that cause access to a VSAM file:

- To access a VSAM file by using its cluster name instead of a data or index name, you must specify the proper level password for the cluster. The proper level password for the cluster is required even if the data or index passwords are null (that is, no password was assigned).
- To access a VSAM file by using its data or index name instead of its cluster name, you must specify the proper level data or index password. If cluster passwords are defined, however, the master password of the cluster may be specified instead of the proper data or index password.
- If a cluster is not password protected, you can access the file using the cluster name without specifying passwords. This is true even if the data and index entries of the cluster have passwords defined. This allows unrestricted access to the VSAM file as a whole, but protects against unauthorized modification of the data or index as separate components.
- An update password is required at open for MACRF=IN files when DLBL DISP or ACB CLOSDSP is DELETE or DATE.

User Security Verification Routine

If you specify password protection when you define a file (or catalog), you can also supply your own routine to double-check the authority of a processing program to access the file. To use this routine, specify the name of your USVR (user security verification routine) in the AUTHORIZATION parameter of the DEFINE or ALTER command.

The verification routine must reside in the core image library (either system of private). VSAM transfers control to the verification routine only after the program trying to open the file gives a correct password other than the master password. (The verification routine is always bypassed whenever a correct master password is specified.) The authorization option can also include a maximum of 255 bytes of information which will be passed to the authorization routine when it is called. When the authorization routine gets control from VSAM, the registers are set as follows:

Register	Contents
0	Unpredictable
1	Address of a parameter list
2-13	Unpredictable
14	Return address to VSAM
15	Entry point to verification routine

The parameter list has a 44-byte *file-ID*, an 8-byte prompting code (from the CODE option) or zeros, an 8-byte file owner ID (from the OWNER parameter of DEFINE), and an authorization string of up to 255 bytes.

When the authorization routine returns to VSAM, register 15 should be set to zero if the processing program is authorized to access the file or catalog. If register 15 is not zero, VSAM does not allow the processing program to open the file.

Protecting Shared Data

Files can be shared among partitions, among tasks in a partition, or among VSE systems. File sharing is controlled by (1) the use of the SHAREOPTIONS parameter in the DEFINE command and (2) the type of processing (input or output) for which the file was opened.

For sharing among systems, you must establish the DASD sharing environment via the correct system generation and IPL commands. You are also responsible for ensuring that the volume containing the file is mounted on a *shared* device.

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

The degree of sharing to be allowed for the file is specified, when the file is defined, in the SHAREOPTIONS parameter of the DEFINE command. The SHAREOPTIONS parameter can be changed by the ALTER command (if the file is not concurrently open for another program). A file cannot be deleted or reset if it is currently open for another program, regardless of the sharing option specified.

During the initial load of a file (regardless of the share option values specified), VSAM treats the share option specification as if it were (1). After the file is loaded and sucessfully closed, VSAM uses the original share option value.

One of the following file sharing options can be specified:

- Sharing option 1: The file may 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 may be opened by more than one program for input processing and, at the same time, it may 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. VSAM does nothing to ensure either the integrity of information written in the file or the integrity of the data retrieved from the file. VSAM does ensure, however, that an open file is not deleted or reset.
- Sharing option 4: A key-sequenced or relative-record file can be opened by any number of programs for both input and output processing by users in the first system requesting the file. Once a file has been opened for output by one system, VSAM accepts only open for input requests from another system.

  VSAM ensures write integrity by using the VSE LOCK facility. Read integrity is ensured by VSAM only when records are being retrieved for update. If records are not being retrieved for update, some records in control intervals being updated concurrently by more than one program may be missed or skipped by VSAM because each program might retrieve a different copy of the control interval. If one task makes multiple requests (through two or more ACBs) to the same file, VSAM cannot resolve the integrity conflict and issues an error code. The requestor must resolve the conflict and retry the request.

Note:
If you specify sharing option 4 for an ESDS, VSAM treats the specification as share option 2.

If a file cannot be shared for the type of processing you specify, your request to open a file is denied.

If a file is fully sharable (sharing options 3 and 4), more than one program can open it at the same time to update or add records. If the file is not sharable, only one program at a time can open it to update or add records. With sharing options 2, 3, or 4, any number of programs can retrieve records from the file regardless of whether it is sharable or not. With sharing option 1, data retrieval is prevented by the OPEN macro if the file is already opened for output.

If an alternate index is defined with the UPGRADE attribute and sharing option 1 or 2, keep in mind the restrictions on the number of users who can open it for input and/or output processing. For example, if sharing option 2 is specified for an alternate index that is a member of an upgrade set, once an update path over the base cluster is opened for output you cannot open another update path over the base cluster, or the base cluster itself, for output because this option does not allow a file to be opened twice for output.

Cross-Systems Sharing

Cross-systems sharing permits files and catalogs to be shared among several VSE systems. You do not need to invoke cross-systems sharing support when opening VSAM files and catalogs. Catalogs are automatically shared if they reside on shared devices (as defined to the supervisor at IPL). File sharing is supported for all files owned by a shared catalog. Both the catalog and the file must reside on shared devices.

VSE does not support cross-systems SHAREOPTIONS (4) (concurrent output processing by two processors). Specifying SHAREOPTIONS (4[,n]) results in sharing option 4 support across partitions and sharing option 2 support across systems.

You may wish to have a nonshared master catalog on each system with shared user catalog(s) connected to each master catalog. To do this, define the user catalog under one master catalog, then IMPORT CONNECT the user catalog to another master catalog. The shared (user) catalog(s) must contain entries for all shared files.

Data Integrity

Access Method Services has both commands and command options for VSAM data integrity. Also, VSAM and VSE utility programs will aid you in protecting data. The following list indicates what the integrity and protection tools are:

Integrity Commands and Command Options

DEFINE CLUSTER allocation option
DEFINE CLUSTER RECOVERY|SPEED option
DEFINE CLUSTER DATA WRITECHECK option
DEFINE CLUSTER WRITECHECK option
DEFINE USERCATALOG command
DEFINE SPACE command
DELETE SPACE FORCE option
EXPORT/IMPORT commands
EXPORTRA/IMPORTRA commands
LISTCAT command
LISTCRA command
REPRO command
RESETCAT command
VERIFY command

VSAM and VSE Utility Programs

FAST COPY DISK (CKD)
FAST COPY DISK (FBA)
VTOC Utility (IKQVDU) VSAM

These commands, options, and programs are also listed later in this chapter in the section "Guide to VSAM Recovery" with a brief explanation of how each is used. The list shows where to find a more detailed explanation and backup and recovery procedures that use integrity and protection tools.

You can use the commands DEFINE SPACE, DEFINE USERCATALOG, and DEFINE CLUSTER to your advantage for data integrity. Although these commands and options are not designed specifically for integrity, they can be used to improve data integrity.

Using the DEFINE SPACE Command

The DEFINE SPACE command DEDICATE parameter can be used to easily dedicate an entire volume or group of volumes to VSAM by defining a space that occupies the whole volume. Other volumes can be used exclusively for nonVSAM files. This allows recovery on a volume basis to be strictly VSAM or nonVSAM. If the volumes are mixed, two different approaches are needed for integrity. For example, a copy of the data on tape is needed to back up the nonVSAM data, but several exports may be all that is necessary for VSAM files. Both the COPY and EXPORT commands are necessary on the mixed volumes. If the volumes are segregated, only one of the integrity measures is necessary.

Using the DEFINE CLUSTER Allocation Subparameter

Secondary allocation that occurs after the last catalog backup results in new catalog records that are not available to the backup catalog. The allocation subparameter of the DEFINE CLUSTER command can be used to improve file integrity and reduce this exposure by eliminating or minimizing secondary allocation. An entry-sequenced file is extended only by adding new control areas to the end of the file. Thus, the effect of addition is predictable and the problem is eased. If it is impractical to allocate enough primary space to accommodate additions, the secondary allocation quantity should be large enough so that extension is infrequent. When secondary allocation does occur, a new backup of the catalog or file (or both) can be made. By monitoring the file statistics in the catalog, either by way of a LISTCAT command or by way of a SHOWCB macro against an open ACB (to inspect the number of bytes of available space), you can predict when secondary allocation will occur. You can determine if a secondary allocation took place with a SHOWCB or TESTCB for the RPL feedback information after each PUT request.

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

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

For these reasons it is advisable to design free-space percentages to minimize the probability of a split for a given insert level, rather than to allow extra primary allocation for expansion. The file should be reloaded (reorganized) when its insert level approaches the design point. For further information, refer to "Distributed Free Space" in Chapter 9.

Using the DEFINE USERCATALOG Command

The DEFINE USERCATALOG command can be used to create many user catalogs (as many as one per volume) and 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 those ten files has to be recovered. Further, if the data is segregated, with a particular catalog having only VSAM entries or only nonVSAM entries, the recovery can be accomplished using only one method:

Catalogs with only VSAM entries can have the RECOVERABLE attribute and the data recovered using EXPORTRA/IMPORTRA. This would recover current data, or access to current data could be regained using RESETCAT.

Once a catalog defined without the RECOVERABLE attribute has been destroyed, the data it controls can no longer be accessed. Thus, if a system contains only one (master) catalog and that catalog is destroyed, the resources of the whole system are lost and must be restored by the use of backup copies. The catalogs with nonVSAM entries can be backed up with the Fast Copy Disk Volume utility program. After the volume is restored only those jobs that updated the files since the backup was made would have to be rerun.

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

Protecting VSAM Files and Volumes

You must plan in advance how much and what kind of protection you need. You must 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 is it sufficient to use the Fast Copy Disk Volume utility program plus the file update reruns necessary to make the file current? The next section, "VSAM Data Backup Considerations," will help you answer these questions. The "Guide to VSE/VSAM Recovery" section (later in this chapter) lists and describes the protection and integrity functions available.

VSAM Data Backup Considerations

In choosing methods of backup and recovery, several factors you must consider, other than the physical methods of accomplishing the job, are: the need for backup, operational characteristics, and security and integrity of the backup medium.

- *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 regular backup procedures and the relative 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:* Frequency of backup and the frequency of possible recovery usually interact with the time required in your consideration of best method of backup and recovery. You may find some methods 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 of backing up might be unacceptable for another file that must be backed up frequently.

- *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. A method that takes longer may have other characteristics that are more desirable. Time required for recovery may also necessitate that a backup technique be used that takes longer.

- *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. If recovery is infrequent, a difficult method may require more time to reason out than another method would require to do the actual recovery.

- *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. Security and integrity factors may also need to be reviewed as the nature of data changes in an installation.

Relationship of Catalog Entries to VSAM Files and Volumes

The VSAM catalog contains information essential to accessing and controlling its files and volumes.

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

If a catalog is recoverable (created with the RECOVERABLE option), a catalog recovery area (CRA) is created on each volume owned by the catalog. The CRA duplicates the file and volume information contained in the catalog about that volume. Figure 11-1 identifies by object type the volume whose CRA contains the duplicate information. However, accessing the volume via the CRA can be done as an integrity function only by:

- Using EXPORTRA/IMPORTRA to recover the data by moving it physically and defining the moved file in a catalog.
- Using LISTCRA to list the CRA's contents or with the COMPARE option to match CRA entries to catalog entries. Mismatch messages indicate volumes out of synchronization with the owning catalog.
- Using RESETCAT to make the information in the CRA and the catalog identical. This is done by setting the catalog information equal to the CRA information. If the CRA has no information for an entry in the catalog, the catalog entry is deleted. If there is no catalog information for a CRA entry, the CRA entry will be added to the catalog. The volume information in the catalog will be compared with the VTOC labels. If the space header in the volume record refers to a non-existent format-1 VTOC label, the space header is deleted. If the format-1 label does not match a space header, the label is scratched. No data is moved by using RESETCAT.

All other types of data access must use catalog information.

Type of entry	**Volume whose catalog recovery area contains a copy of the catalog entry**
Volume entry	Its own volume
Key-sequenced cluster entry and its data and index entries	The volume that contains the (first part of the) cluster's index component
Alternate index entry and its data and index entries, when the alternate index is for a key-sequenced cluster	The volume that contains the (first part of the) alternate index's base cluster's index component
Path entry, when the path is related to a key-sequenced cluster	The volume that contains the (first part of the) path's base cluster's index component
Entry-sequenced cluster's entry and its data entry	The volume that contains the (first part of the) cluster's data component
Alternate index entry and its data and index entries, when the alternate index is for an entry-sequenced cluster	The volume that contains the (first part of the) alternate index's base cluster's data component
Path entry, when the path is related to an entry-sequenced cluster	The volume that contains the (first part of the) path's base cluster's data component
Relative-record cluster entry and its data entry	The volume that contains the (first part of the) cluster's data component
NonVSAM file	The volume that contains the nonVSAM entry's catalog
User-catalog connector entry in the master catalog	The volume that contains the master catalog
Catalog's self-describing entries	These entries are not duplicated in any catalog recovery area

Figure 11-1. Catalog Recovery Area Contents

Creating Backup Copies of VSAM Files

Several methods of backup and recovery can be used for VSAM files. It is usually not possible to use one method for all files in an installation. You should consider individual data sets or groups of data sets and determine the most suitable method for each.

- The EXPORT command is used to create an unloaded, portable copy of the file. The operation is simple, there are options that offer protection, and most catalog information is exported along with the data, easing the problem of redefinition. You can prevent the exported file from being updated until the IMPORT command reestablishes its accessibility. (See "Using EXPORT/IMPORT: Transporting or Backing Up Files" in *Using VSE/VSAM Commands and Macros* for more information and examples.)

- The REPRO command is used to create either a SAM or a duplicate VSAM file for backup. An advantage over EXPORT is the accessibility of the backup copy. A DEFINE command is required before reloading, but this is a relatively minor inconvenience, particularly if the original DEFINE statements can be used. (See "Loading Records into a File" in *Using VSE/VSAM Commands and Macros* for more information and examples.)

- User-written programs for backup are usually most attractive when there is some characteristic of the data that makes backup or recovery easier for you, but which cannot be taken advantage of by a generalized backup method. Files for which not all records have to be saved for backup might fit into this category. Also, keyed sequential files which have to be processed sequentially on a regular basis, could be backed up by creating a sequential file as a by-product.

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

Creating Backup Copies of Volumes

You can use the Fast Copy Disk Volume system utilities to create a backup copy of an entire volume and to restore that copy on a volume. The use of these utilities in a VSAM environment requires special considerations because both the volume VTOC and the catalog contain space mapping information about the volume that has to be synchronized to insure accessibility and to avoid damage to data. If the volume being restored is a recoverable volume (a volume owned by a recoverable catalog), use RESETCAT to synchronize the catalog with the volume (See the chapter "Using RESETCAT: Resetting Catalog Entries" in *Using VSE/VSAM Commands and Macros*). See *VSE/Advanced Functions System Utilities,* for details on how to use Fast Copy Disk Volume. The section "VSAM Recovery Techniques" later in this chapter outlines how to solve out-of-synchronization catalog and volume problems.

Protecting VSAM Catalogs

Because of the importance of the VSAM catalog, you should consider backup for the catalog as well as for files and volumes. If all of the files owned by a catalog are backed up individually, it is possible to recover from destruction of the catalog by carrying out recovery procedures for each of the files. This may be reasonable. The probability of losing an entire catalog is very low. However, to speed recovery or minimize exposure in the case of catalog damage or destruction, three backup methods are available:

- REPRO can be used to unload the catalog to a nonVSAM, or a VSAM file. It can be used to create a backup copy of either a master or user catalog and to re-establish that backup copy as a catalog. This set of functions is referred to as catalog unload and reload. The REPRO command requires no special operands to perform the function. The unload function is triggered when the REPRO source is a catalog and reload is triggered when the REPRO target is a catalog. When a new catalog is defined an unloaded catalog file may be reloaded into the newly defined catalog, or the unloaded catalog can be reloaded into a version or the original catalog. The section "Reloading a Catalog" in "Using REPRO: For Catalog Backup and File Reorganization" in *Using VSE/VSAM Commands and Macros* should be studied carefully before using REPRO unload/reload as your catalog recovery method.
- The entire catalog volume may be backed up by using the VSE Fast Copy Disk Volume utility program.
- Defining your catalog(s) with the RECOVERABLE attribute allows recovery through the catalog recovery areas that reside on each volume owned by the recoverable catalog. Catalog entries may be recovered using the EXPORTRA/IMPORTRA, and RESETCAT commands.

REPRO and the VSE utilities can be used to backup both nonrecoverable and recoverable catalogs. File, volume, and catalog recovery will be discussed from nonrecoverable and recoverable standpoints.

Creating Backup Copies of Nonrecoverable Catalogs

VSAM nonrecoverable catalogs should be protected by backup procedures from the following two types of accidental loss:

1. The data is lost. The actual volume or file information can no longer be read.

   The only way to safeguard yourself from loss of the data is to have a copy of the data in another form or place. The usual method for doing this is to use the VSE Fast Copy Disk Volume utility program to copy the volume to tape or another disk volume.

   When you restore a volume several problems must be overcome:

   - First, the information on the restored volume is downlevel. That is, if your original volume has been updated since the backup was made, these updates must be applied to the restored level of the volume to bring it to the level of the original volume.

   - Second, if the volume is not the catalog volume, you have information about the volume in the catalog that may not match what is actually on the volume. It would be helpful to have a LISTCAT listing of the catalog at the time you created the backup copy to compare with a present listing. The data spaces and file extents may be different if any file updates have been done since the backup was made. See the section "Inaccessible Volume" later in this chapter for complete recovery procedure.

   - Third, if the volume is the catalog volume, all of the volumes owned by the catalog may have file and data space extents that do not match the catalog information. Again, LISTCAT listings of the backup copy and the original catalog help. Each volume must be handled as if the volume was just restored. See the section "Inaccessible Volume" later in this chapter.

   If you have lost a file, an IMPORT command would be the usual way to recover. If you do not have a copy created by the EXPORT Command to use with the IMPORT command, either redefine the file and rerun the initial load and all updates to recreate the file or restore a backup level of the volume that contains a downlevel copy of the file, use the EXPORT command to create a portable copy, use the IMPORT command to restore the copy to your present volume and rerun the updates run since the backup was taken.

2. If no data has been lost, but the catalog is partially or totally unusable, use a backup copy of the catalog.

   The REPRO Command can be used to take periodical backups of the nonrecoverable catalog(s). The REPRO commands unloaded version (copy in backup form) can be reloaded directly into the inaccessible catalog (if the damage was not a physical problem). Then all the files could be removed from the volume using the EXPORT command (if there were entries for them in the unloaded copy). Then the volumes cleared of data spaces, data spaces redefined and files redefined in the catalog by the IMPORT command. This procedure can be used to recover the current files.

   If a REPRO unload copy is not available, then a volume backup must be restored and volume recovery procedures followed. See the section "Inaccessible Volume" later in this chapter.

Creating Backup Copies of Recoverable Catalogs

VSAM catalogs can be defined with the RECOVERABLE attribute, which makes it possible to recover VSAM files and their catalog entries if the catalog is damaged or destroyed. Recovery is made possible by recording catalog information about an owned volume on that volume in a catalog recovery area (CRA). Recovery information is recorded on each volume owned by the catalog. Space for the CRA that contains this information is automatically set aside when you define the first data space on a new volume and also when you define the catalog, itself. There is no separate catalog entry for the CRA: VSAM records its physical disk address in the volume's format-4 VTOC label.

The recovery information in the CRA is updated immediately when parallel information in the catalog is changed by VSAM catalog management. The affected volume(s), as determined by the operation to be performed (data space, cluster, path), must be mounted.

Note: If the catalog is changed by the REPRO reload command or restored using a VSE utility, the CRAs owned by the catalog on volumes other than the catalog volume are unchanged. The catalog volumes CRA is restored using the VSE utility and is unchanged by REPRO. A LISTCRA using the COMPARE option of the restored catalog reflects CRA-catalog mismatches.

Recoverable catalogs cannot be used with nonVSAM entries, so if you are planning to put nonVSAM entries into a VSAM catalog, read the section "Creating Backup Copies of Nonrecoverable Catalogs" in this chapter. You might consider putting just the nonVSAM file entries into one catalog and VSAM entries in another. Then recovery can be used and nonVSAM files can be cataloged as well.

If you should discover that your recoverable catalog is damanged and its entries are inaccessible, downlevel, or contain erroneous information, you have the option of chosing one of two methods to restore your catalog to a usable condition.

- *EXPORTRA/IMPORTRA*: You can use this method to selectively recover specific catalog entries. Reorganization of your catalog and data is a by-product of this method since it involves the movement of data. You may use the following procedure to restore accessibility to your catalog:

  Issue the EXPORTRA command. EXPORTRA uses the information in the CRA, rather than the catalog, to gain access to the VSAM files and produce a copy. The copy can be introduced into the system by means of the IMPORTRA command. (See the section "Using EXPORTRA/IMPORTRA: Recovering Catalog Entries and Data" in *Using VSE/VSAM Commands and Macros.*

- *RESETCAT*: If you don't want your data to be moved and if you wish to confine all updating to the catalog (and CRAs), you can use RESETCAT. RESETCAT does not permit selective reset of specific catalog entries. An entire volume's worth of catalog entries are reset. You would use RESETCAT if a catalog or one or more of its owned volumes becomes inaccessible:

  First, if a catalog is inaccessible, the REPRO command can be used to reload the catalog or a volume restore may be done. Then the RESETCAT command can be used to synchronize the catalog's entries to the level of its owned volumes. If the inaccessible volume is not the catalog volume the volume can be restored and the RESETCAT command used. (Also see the sections "Inaccessible Volume" and "Unusable Catalog" later in this chapter.)

Guide to VSE/VSAM Recovery

VSAM Recovery is the process of regaining access to lost VSAM data. If access to VSAM data is lost, Access Method Services, VSAM and VSE utility programs can be used in combination to regain access to the data. Some of the utility programs can only recover data that is not current *downlevel* and further processing must be done to make the file, volume, or catalog current.

Other tools are used to organize the VSAM data so it can be more easily recovered. These tools are used mostly to back up data if files are damaged. Protecting data in this way must be done before data is lost. (See "Protecting VSAM Files and Volumes" and "Protecting VSAM Catalogs" in this chapter.)

Levels of Recovery

Two types of VSAM data recovery in terms of the currency of the recovered data are: *current* and *downlevel.*

The *current* type of data recovery operation restores addressibility and access to the most recent version of the data. Operations that recover current data are generally used to correct problems such as read and write errors associated with the data itself or with the data description.

The *downlevel* type of data recovery operation restores addressibility and access to a version of the data other than the most recent. Operations that recover downlevel data are generally used to correct logical problems such as a programming error or faulty transactions. This is the most common type of recovery, probably because of the types of problems encountered and the level of data available for recovery. An example of a downlevel recovery is the restore of a volume.

VSE/VSAM Recovery Tools

The following chart contains Access Method Services recovery tools and other integrity options as well as backup programs and commands. In the chart, several of the column headings are not self-explanatory and are explained here:

TOOL TYPE indicates where the tool is supported: Access Method Services, VSAM program, or VSE utility program.

LEVEL indicates what level of data the command or program recovers or helps recover: CUR is current data, no further processing is required after the data is recovered. DOWN is downlevel data, usually means a backup copy is needed to recover data and then further processing to make the data current. ANLY is used to indicate this is an analysis tool and is used to determine level and synchronizationof file or volume, with its catalog.

FILE TYPE indicates what is being recovered: FILE is VSAM file, VOL is volume, and CAT is VSAM catalog.

TOOL CLASS indicates VSAM the command or program class: REC is command designed for recoverable catalogs, BKP is a backup command or program other than recovery, and INT is any tool that is a VSAM integrity option, other than recovery-type tools.

Recovery Tool Name	Tool Type	Level	File Type	Tool Class	Application	Where Discussed
FAST COPY DISK VOLUME	VSE Utility	DOWN	VOL	BKP	Use the Fast Copy Disk Volume system utility to create a backup copy of an entire volume and to restore that copy on a volume. The use of these utilities in a VSAM environment requires special considerations due to the fact that both the volume VTOC and the catalog contain space mapping information about the volume which has to be synchronized to insure accessibility and to avoid damage to data.	• *VSE/Advanced Functions System Utilities*
EXPORT/IMPORT	Access Method Services	DOWN	FILE	BKP	Use the EXPORT command to create backup copies of data and associated catalog entries. The catalog entries can be reestablished in the catalog from which they were extracted or into a different catalog using IMPORT command. The data file is reestablished by IMPORT without redefining it.	• "Using EXPORT/IMPORT: Transporting or Backing Up Files" in *Using VSE/VSAM Commands and Macros* • "Creating Backup Copies of VSAM Files" and "Creating Backup Copies of Nonrecoverable Catalogs" in this volume
EXPORTRA/IMPORTRA	Access Method Services	CUR	FILE	REC	Use the EXPORTRA command to recover data independent of the status of the catalog. The recovered data can then be imported into the system and catalog using IMPORTRA.	• "Using EXPORTRA/IMPORTRA: Recovering Catalog Entries and Data" in *Using VSE/VSAM Commands and Macros* • "Creating Backup Copies of Recoverable Catalogs" in this volume
LISTCAT	Access Method Services	ANLY	FILE,VOL, CAT	REC	Use the LISTCAT command to list the contents of a nonrecoverable catalog after a recovery operation. Visually compare this list with a copy of the LISTCAT list most recently done before the recovery.See the section "Catalog Entry Mismatches" which describes the out-of-synchronization condition you may find.	• "Catalog Entry Mismatches" in this volume • "Using LISTCAT: Listing Catalog Entries" in *Using VSE/VSAM Commands and Macros*
LISTCRA	Access Method Services	ANLY	FILE,CAT, VOL	REC	Use the LISTCRA command to list the contents of a catalog recovery area (CRA) and, if desired, to compare the contents of the CRA with the catalog. The COMPARE option is useful in detecting potential catalog problems and in checking the validity of a catalog that was reestablished by means of the REPRO command or a volume restore. You can use the LISTCRA command with the COMPARE option to identify the level of VSAM recovery required by inaccessible files, unusable catalogs, and inaccessible volumes. The chart in the section "LISTCRA Mismatch Messages" aids in analysis of LISTCRA output.	• "LISTCRA: Analysis of Recoverable Catalogs" in this volume
RECOVERABLE attribute for catalogs	Access Method Services	CUR	VOL,CAT	REC	The recoverable catalog is created by including the optional RECOVERABLE attribute when you define the catalog. When this is done, a catalog recovery area (CRA) is created on each volume owned by the catalog. All catalog entries are duplicated in the CRA of the volume where the entry is pertinent. Loss of any or all catalog entries can be recovered via the CRA and the CRA processing tools: EXPORTRA, IMPORTRA, LISTCRA, and RESETCAT.	• "Relationship of Catalog Entries to VSAM Files and Volumes" in this volume • "Creating Backup Copies of Recoverable Catalogs" in this volume

Figure 11-2. Recovery Tools and Integrity Options (Part 1 of 3)

Recovery Tool Name	Tool Type	Level	File Type	Tool Class	Application	Where Discussed
REPRO	Access Method Services	DOWN	CAT	BKP	The REPRO command is used to create a backup copy of the catalog. The unloaded or backup copy can be reloaded into a newly defined catalog or a version of the original if the backed up catalog becomes unusable.	• "Using REPRO: Catalog Backup and File Reorganization" in *Using VSE/VSAM Commands and Macros* • "Creating Backup Copies of Nonrecoverable Catalogs" in this volume
RESETCAT	Access Method Services	CUR	FILE,VOL, CAT	REC	Use the RESETCAT command to synchronize the catalog with its owned volumes. The CRAs are merged with the catalog entries on a volume basis (a single volume may be selected) and the catalog entry is reset to the level of the volume entry.	• "Using RESETCAT: Resetting Catalog Entries" in *Using VSE/VSAM Commands and Macros* • "Creating Backup Copies of Recoverable Catalogs" in this volume
VERIFY	Access Method Services	CUR	FILE	INT	Use the VERIFY command to correct the catalog information in an attempt to regain access to a file that was improperly closed.	• "Verifying a File's Accessibility" in *Using VSE/VSAM Commands and Macros* • "File Not Properly Closed" in this volume
VTOC Utility (IKQVDU)	VSAM Utility Program	CUR or DOWN	VOL	BKP	Use the VSAM VTOC utility program IKQVDU to initialize a VSAM-owned volume when the owning catalog is not available. VSAM volume ownership can be given up and VSAM space can be returned to the VTOC as available space. All data in that space is lost. Caution: The owning catalog is not modified.	• *VSE/VSAM VSAM Logic, Volume 1* or *2*
DEFINE SPACE	Access Method Services		VOL,CAT	INT	Use the DEFINE SPACE command to dedicate use of volumes for VSAM files in order to segregate VSAM and nonVSAM recovery. You can dedicate a volume by defining a VSAM data space that occupies the whole volume, or by specifying the DEDICATE parameter.	• "Data Integrity " in this volume • "Defining a VSAM Data Space" in *Using VSE/VSAM Commands and Macros*
DEFINE USERCATALOG	Access Method Services		VOL,CAT	INT	Use DEFINE USERCATALOG command to maximize the use of user catalogs and to limit the use of the master catalog. Compare the effect of the loss of a catalog when 10 files are cataloged in each of 10 catalogs, and 50 files are cataloged in each of two catalogs. The fewer the catalogs the greater the disruption of daily operations in the event of loss of a catalog.	• "Using DEFINE: Defining Objects in a Catalog" in *Using VSE/VSAM Commands and Macros* • "Defining a Catalog" in Chapter 5 of this volume
DEFINE option WRITECHECK	Access Method Services	CUR	FILE	INT	Use the optional WRITECHECK parameter of the DEFINE command to verify each write operation when writing data to auxiliary storage. (Refer to the WRITECHECK parameter for an explanation.)	• "Defining a VSAM File (Cluster)" in *Using VSE/VSAM Commands and Macros* • "DEFINE CLUSTER" in *Using VSE/VSAM Commands and Macros*

Figure 11-2. Recovery Tools and Integrity Options (Part 2 of 3)

Recovery Tool Name	Tool Type	Level	File Type	Tool Class	Application	Where Discussed
DELETE SPACE FORCE	Access Method Services		VOL	INT	Use the DELETE SPACE FORCE command to remove information from both the VTOC and the catalog. When space is deleted by using FORCE option, the VTOC's VSAM volume ownership is given up, VSAM space is returned to the VTOC, the space definition in the catalog for that volume is deleted, and VSAM files on that volume are marked as unusable in the catalog. If you want to redefine the files, you must first delete them.	• "Specifying Information That Deletes an Entry" in *Using VSE/VSAM Commands and Macros*
DEFINE CLUSTER RECOVERY|SPEED	Access Method Services	CUR	FILE	INT	When you define a cluster, you can indicate that VSAM is to preformat each control area as records are loaded into the cluster (RECOVERY) or is not to preformat them in interest of performance (SPEED). As records are loaded into a preformatted area, there is always a following end-of-file indicator that indicates how far loading has progressed. If an error occurs that prevents loading from continuing, you can readily identify the last successfully loaded record and resume loading from that point.	• "Defining a VSAM File (Cluster)" in *Using VSE/VSAM Commands and Macros*
DEFINE CLUSTER Allocation	Access Method Services		FILE	INT	Minimize or eliminate secondary allocations for files to overcome the difficulty in catalog recovery stemming from secondary extents.	• "Using the DEFINE CLUSTER Allocation Subparameter." in this volume

Figure 11-2. Recovery Tools and Integrity Options (Part 3 of 3)

LISTCRA: Analysis of Recoverable Catalogs

You can use the LISTCRA command either to list the contents of the CRA before you do selective recovery or to list the entries in the recovery area that are different from those in its associated catalog. (See "EXPORTRA/IMPORTRA: Recovering Catalog Entries and Data" in *Using VSE/VSAM Commands and Macros* for more information and examples.)

See Figure 11-1 for the contents of CRAs and information about how catalog entries are placed in CRAs.

The LISTCRA command with the COMPARE option can be used to analyze the following conditions:

- File not properly closed
- Inaccessible file
- Unusable catalog
- Inaccessible volume

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

LISTCRA Mismatch Messages

CATALOG VOLUME RECORDS

MISCOMPARES

Message Type		**Cause**	**Action**
DATA SPACE EXTENTS	Mismatched data space group	A difference in the number, size, and/or location of VSAM data spaces.	Requires volume recovery
		A difference in the number and/or location of extents for one or more files.	Requires volume recovery
		Space has been extended or deleted.	Requires volume recovery
DATA SET DIRECTORY	Mismatched data set directory	A difference in the names and/or number of files associated with this volume.	Requires volume recovery
VSAM OBJECT RECORDS			
CATALOG ENTRY HAS DIFFERENT NAME	Mismatched name	The catalog was restored, or the volume containing the file was restored. As a result the record in the catalog pointed to by the CRA record is no longer for the same object.	Requires file recovery.
VOLUME OR KEY-RANGE	Mismatched volume or key range.	The catalog was restored, or the volume containing the file was restored. As a result the object's volume locations or key-ranges in the CRA do not match those in the catalog.	Requires file recovery.
EXTENTS	Mismatched extents	File was not properly closed, the catalog was restored, or the volume containing the file was restored.	Requires file recovery.
HIGH USED RBA	Mismatched high used relative byte address	Same as for mismatched extents.	Requires use of the VERIFY command to correct the high RBA.
STATISTICS	Mismatched statistics	Same as for mismatched extents.	No recovery action is required; mismatched statistics do not affect the accessibility of data.
OTHER	Mismatch of something other than the above fields, e.g. passwords.	Same as for mismatched extents.	Same as for mismatched statistics

Catalog Entry Mismatches

Whenever a catalog is used out of synchronization with the volumes it owns, there is the possibility that the information in the catalog does not match the physical characteristics of the volumes or files that it describes. Catalog entry mismatches may indicate that the data is inaccessible, partially accessible or completely accessible.

The descriptions of catalog-CRA mismatches are meant to help guide you through a visual comparison of two LISTCAT listings. One listing taken of a catalog prior to the catalog being restored and the other taken after the catalog is restored. If an entry difference is noted, consult "Appendix B: Interpreting LISTCAT Output Listings" in *Using VSE/VSAM Commands and Macros* for the LISTCAT field descriptions, then use this section to determine what mismatch has occurred. A recommended course of action is located for the mismatch in the section "LISTCRA Mismatch Messages" in this chapter.

This method is for the analysis of nonrecoverable catalogs for out of synchronization conditions that may occur when the nonrecoverable catalog is restored to a previous level.

Determination of Catalog Mismatch

Recoverable Catalogs: The COMPARE option of the LISTCRA command can be used to determine whether a backup catalog mismatches a volume that it owns. The COMPARE option compares the catalog recovery area on the volume with the entries in the catalog and lists the fields that don't match.

Non-recoverable catalogs: There is no information on the volume about VSAM objects in the catalog; however, by comparing the LISTCRA runs that were made when the catalog was backed up with runs made when the catalog is restored, critical changes may be detected.

Volume Mismatches

Mismatched Space Map: This mismatch indicates that the catalog does not correctly reflect the tracks (min-CAs) on the volume occupied by VSAM files. Files wholly contained in space correctly indicated as allocated can be accessed if their data set descriptions in the catalog are correct.

Mismatched Data Space Group: This mismatch indicates that the catalog does not correctly reflect the VSAM files on the volume. Files wholly contained within data spaces that are accurately described are accessible if their data set descriptions in the catalog are correct.

Mismatched Data Set Directory: This mismatch indicates that the catalog does not correctly reflect the data sets on the volume. Data sets that are known from the data set directory are accessible if their data set descriptions are correct.

Data Set Mismatches

Mismatched Statistics: These mismatches do not affect accessing of a file.

Mismatched High RBA: This mismatch indicates that the catalog does not correctly reflect the end of data in a file. This condition can be corrected by running VERIFY against the file.

Mismatched Extents: This mismatch indicates that the file has acquired additional extents that are not reflected in the catalog. The data contained in the extents that are correctly identified may be accessed; for a key-sequenced file it may be necessary to treat the data portion as an entry-sequenced file in order to access the data.

Mismatched Volume or Key Range: This mismatch indicates that either the file was extended to a volume that the file record in the catalog does not know about or that the file on the volume has the same name as but is not the same file that is described in the catalog. If the file was extended to a volume not known in its catalog record the extents of the file on that volume are not accessible; the extents of the file on known volumes may be accessible.

Actions that Cause Catalog Mismatches

There are several actions that cause mismatches from a backup catalog. Some of these are overt actions such as the use of the DEFINE and DELETE commands to create files or data spaces; others are automatic system actions, such as acquiring additional extents.

Define/Delete/Extend Data Space: Any of these actions cause the data space group set of fields for a data space to be invalid in a backup catalog.

Define/Delete Files: Either of these actions cause the data set directory in the volume record and some of the file entries to be invalid in a backup catalog. The use of the EXPORT command may cause a deletion. The use of the IMPORT command always causes an entry definition.

ALTER ADDVOLUMES/REMOVEVOLUMES: The ALTER command is used to add a volume to a file as a candidate or remove a volume from a file as a candidate.

File Extension via Suballocation: Extension causes the volume space map in the backup catalog to be invalid as well as the entry for the file.

Minimization of Catalog Mismatches

The possible catalog mismatches described above, which cause files to be wholly or partially inaccessible, are all caused by the DEFINE, DELETE, and ALTER commands, or by the extension of VSAM files or data spaces. Since DEFINE, DELETE, and ALTER are always known to you, a backup copy of the catalog can be made each time one of these commands is used. The only thing, then, that invalidates a backup catalog without you being aware is the extension of space. Thus, the minimization of space extension tends to minimize critical catalog changes. Further, you have the ability to define VSAM objects with no secondary extent value and thus prevent any VSAM object from extending. As long as a VSAM object does not extend, it remains totally accessible from a backup copy of the catalog.

VSAM Recovery Techniques

These step-by-step procedures are used to analyze and recover from the following conditions:

- File not properly closed
- Inaccessible file
- Unusable catalog
- Inaccessible volume

The explanations given in the sections on creating backup copies of files, volumes, and catalogs should be read as well because the two areas, backup and recovery, overlap.

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

- The volume being restored does not contain multivolume files.
- If a volume does contain a portion of a multivolume file, all volumes that contain portions of those multivolume files are treated as a single unit; that is, if a volume is required, the entire set is restored.

File Not Properly Closed

Cause of Failure: VSAM files are not properly closed when they are opened for output and a system failure occurred, or abnormal termination was entered. This condition is reflected in the catalog and is communicated to the next program that does an OPEN of the file. There is a possibility that the failure occurred after the load or update of the file was complete, then the file itself and the file's catalog entry are correct.

Error Conditions:

- Incorrect high RBA in catalog
- Incomplete write to direct access device
- Duplicate data

Recovery for Incorrect High RBA in Catalog: This is the error most likely to occur. If you are running in RECOVERY mode, correct the error by using the VERIFY command to scan a given file starting from the indicated high RBA in the catalog to the end of the file. The resultant high RBA is then used to update the catalog.

Recovery for Incomplete Write to a Direct Access Device: The file must be restored from a backup copy. You can use either an exported or sequential backup copy created by the REPRO command.

Use the IMPORT command to put a previously exported copy into the catalog, or:

1. Delete the file that failed.
2. Redefine the file with the DEFINE command.
3. Load the new file with the sequential backup file by using the REPRO command.

The restored file is downlevel and all updates since the backup was taken must be reapplied to make the file current.

Recovery for Duplicate Data in a Key-Sequenced File, Alternate Index, or Catalog: This can result from a failure during a control interval or control area split. One of two possible situations can exist for a duplicate data error conditions, depending on the type of processing being done.

For addressed or control interval (CNV) processing, you correct the error condition by using the REPRO command to copy the current version of data to a temporary file and then copy it back into the original file. This gives you a reorganized file without duplicate data.

For keyed or sequential processing, VSAM automatically detects and corrects the duplicate data condition (VSAM erases the original versions of the copied records.) Duplicate records caused by a failure during a control interval split in VSE/VSAM may cause an error if the file is processed by DOS/VS.

Summary: The warning "data set not properly closed" may indicate an error in a VSAM file. This condition can generally be corrected by using the VERIFY command.

If other errors are encountered or suspected, they can generally be corrected by using either the IMPORT command or the REPRO command.

Inaccessible File

Cause of Failure: A VSAM file may become inaccessible due to damage to the file itself, damage to related information in the catalog, or both. Depending on the extent of damage and prior actions, it may be possible to gain access to either the current or a downlevel version of the data.

Error Conditions:

- The file cannot be opened
- The file is partially unreadable (but can be opened)
- The file is totally unreadable (but can be opened)

Recovery for the File That Cannot be Opened: This is probably due to catalog damage. Determine the extent of this damage. If the damage is relatively minor (that is, relatively few catalog file entries are affected):

1. Use an analysis tool to determine the extent of damage.
   a. Use LISTCRA (with the COMPARE option) to determine how many file entries are affected and if there is damage to the volume information. If mismatches occur see the "LISTCRA Mismatch Messages" section for suggested action. LISTCRA can only be used for recoverable catalogs.
   b. Use LISTCAT if the catalog is not recoverable. This can be done by comparing a previous list of LISTCAT with a list of the damaged catalog. See the section "Catalog Entry Mismatches" in this chapter if entry mismatches are detected.
2. In a nonrecoverable catalog, if an exported copy of the file is available, you can import it to gain access to a downlevel copy of the file.
3. In a recoverable catalog, you can use RESETCAT or EXPORTRA/IMPORTRA depending on whether you want to reset the catalog entry to the level of the volume or move and reorganize the file(s).
   a. If you desire to gain access to current data and not move the data, the RESETCAT command can accomplish this. The volume's CRA information replaces non-matching information.

   or

   a. The EXPORTRA command uses the volume's CRA to gain access to the data and move it into a temporary sequential file.

b. The IMPORTRA command deletes the entry of the damaged file in the catalog and redefine the file(s) it moves to the receiving volume. This move also reorganizes the moved files.

Recovery for the File That is Partially Unreadable: The problem is either confined to the file itself or an entire physical extent of the file is not readable.

1. Use an analysis tool as indicated in "Recovery for the File That Cannot Be Opened" (above) to determine if there is a mismatch in the number of extents. If the catalog indicates one or more extents than are on the volume, it may be caused by a volume being restored independent of the catalog.
2. If there is a mismatch between the catalog and the CRA, you can use EXPORTRA/IMPORTRA or RESETCAT, or, for a nonrecoverable catalog, you can import a previously exported copy. See "Recovery for the File That Cannot Be Opened" (above) for use of these tools.
3. If no catalog mismatch, a backup copy of the file must be restored, using EXPORT/IMPORT or REPRO.

Recovery For the File That is Completely Unreadable: Either the file has been destroyed or the catalog and volume are not synchronized. Using the procedures outlined for other types of file recovery:

- Analyze the catalog with LISTCRA or LISTCAT to determine if the damage is in the file or in the catalog.
- If the damage is in the catalog and the catalog is recoverable, you can use EXPORTRA/IMPORTRA or RESETCAT to gain access to current data.
- If the damage is to the file or if the catalog damage is to a nonrecoverable catalog use IMPORT or REPRO to restore the file. This gives you access to a downlevel copy of the data.

Summary: The inaccessibility of a VSAM file can be analyzed, using the LISTCAT or LISTCRA commands and the extent of file damage determined. Recovery of data can be affected, based on the analysis, using EXPORTRA/IMPORTRA, RESETCAT, IMPORT, and REPRO.

Unusable Catalog

Cause of Failure: A catalog may become unusable because of physical damage to the catalog. Depending on the extent of the damage and prior actions, it may be possible to gain access to current level catalog entries or to downlevel catalog entries.

Error Conditions:

- Catalog can be opened, but many VSAM files cannot be opened.
- The catalog cannot be opened.
- The catalog volume is unusable

Recovery for the Catalog that Can be Opened, But Many VSAM Files Cannot: A problem with the catalog probably exits. This can be determined by using an analysis tool. If I/O errors are encountered or mismatches are detected, some form of catalog recovery is required. If not, the problem is confined to the files themselves and the procedures given for "Recovery for the File Cannot Be Opened" (above) can be used. (See "Inaccessible File".)

1. Use an analysis tool to determine if a problem exists in the catalog.

   a. Use LISTCRA (with the COMPARE option) to determine how many file entries are affected and if there is damage to the volume information. If mismatches occur see the "LISTCRA Mismatch Messages" section in this chapter for suggested action. LISTCRA can only be used for recoverable catalogs.

   b. Use LISTCAT if the catalog is not recoverable. This can be done by comparing a previous list of LISTCAT with a list of the suspect catalog. See the section "Catalog Entry Mismatches" in this chapter if entry mismatches are detected.

2. If the problem is with the catalog, recovery depends on the availablility of backup copies of the catalog, volumes, and files, and whether the catalog has been defined with the RECOVERABLE attribute. Nonrecoverable catalogs are discussed first:

   a. Delete each of the VSAM files that cannot be opened.

   b. Redefine these files in the catalog, or use the IMPORT command to load backup copies created by the EXPORT command.

   c. If backup copies created by the EXPORT command are not available, load the files with backup REPRO copies, if available.

   or

   a. Restore all affected volumes with backups (that were created at the same time), to volumes other than the current level volumes.

   b. Use REPRO or EXPORT to move downlevel copies of affected files to temporary files.

   c. Use IMPORT command or the REPRO command to put these temporary files into the current catalogs volumes. The current files are unaffected.

3. *For Recoverable Catalogs:* If an unloaded copy of the catalog built by using the REPRO Command or a backup copy of the catalog volume is available, you can do the following:

   a. Restore the catalog volume or reload the catalog into the previous version.

   b. Use LISTCRA (with the COMPARE option) to identify mismatched volumes and files.

   c. Use RESETCAT to reset the catalog entries to the level of its owned volumes.

4. If you prefer to move and reorganize files rather than reset catalog entries with RESETCAT, you can use the following EXPORTRA/IMPORTRA procedure:

   a. Restore the catalog volume or reload the catalog.

   b. Use LISTCRA (with the COMPARE option) to identify volume and file mismatches.
   The three mismatch situations are:

      - The catalog volume entry is mismatched.
      - A volume entry other than the catalog volume is mismatched.
      - Files are mismatched on volumes that are not mismatched.

The Catalog Volume Entry is Mismatched: If the catalog volume entry is included as a mismatched volume (that is, a data set directory or a data space group mismatch), the catalog and its owned volume are out of synchronization. (This condition does not occur if the catalog volume was restored.)

You can do the following steps to reestablish the catalog and its files:

1. Recover all files on all owned volumes, using the EXPORTRA command.
2. Use the VSAM VTOC utility (IKQVDU) to clean up all owned volumes.
3. If the unusable catalog is a user catalog, use the EXPORT command with the DISCONNECT subparameter to delete the user catalog pointer from the master catalog and avoid a conflict in catalog names during redefine.
4. Define a new catalog and space on all owned volumes, using the DEFINE command.
5. Use the IMPORTRA command to reestablish the files.

A Volume Entry Other Than the Catalog Volume Mismatched: This means a data set directory or a data space group mismatched. You can recover all files on the mismatched volumes using the following procedure:

1. Recover all files on the mismatched volumes with the EXPORTRA command.
2. Use the DELETE command (with the FORCE option) to clear the volume of VSAM data spaces. This may cause file entries to be marked unusable in the catalog, these should be deleted.
3. Define space on the volumes.
4. Use the IMPORTRA command to reestablish the files recovered with the EXPORTRA command.

Files Are Mismatched On Volumes That Are Not Mismatched: If the mismatches are mismatched RBAs, use the VERIFY command to update the RBAs. Other mismatches should use this procedure:

1. Recover the mismatched files with EXPORTRA.
2. Use IMPORTRA to reestablish the files recovered by EXPORTRA.

Recovery for the Catalog That Cannot Be Opened: You must have a backup copy of the volume created by the REPRO command or a VSE utility to regain access to a nonrecoverable catalog. The recoverable catalog can be rebuilt with current data, if all volumes and files can be accessed by EXPORTRA. First, nonrecoverable procedures:

1. Restore the volume or reload the backup copy into the previous version of the catalog.
2. Use LISTCAT listings of backup files and current files to determine if there are mismatches. See the section "Catalog Entry Mismatches" in this chapter if entry mismatches are detected.
3. For those files with other than a RBA or general mismatch, delete the file and reestablish with a backup copy of the file created by the IMPORT command or the REPRO command.

To recover with a recoverable catalog:

1. Use EXPORTRA to recover all files on all volumes.
2. Use IKQVDU to clear all volumes.

3. Redefine the catalog and all data spaces.
4. Use IMPORTRA to reestablish all files.

or

1. Use EXPORTRA to recover all files on the catalog volume.
2. Restore the catalog volume or reload into a previous version of the catalog.
3. Use RESETCAT to reset the catalog to the level of its volumes.
4. Use the IMPORTRA command to reestablish the files.

Recovery For the Catalog Volume That is Unusable: For nonrecoverable catalogs see the procedure for "Recovery for the Catalog That Cannot Be Opened" (above). The recoverable catalog can be recovered as above, except: 1) The catalog volume must be restored. 2) The EXPORTRA/IMPORTRA commands used to recover those downlevel files on the catalog volume or the files must be recovered from backup copies created by the EXPORT command or the REPRO command.

Summary: An unusable catalog can be reestablished provided certain backup procedures made possible by the system copy utility and the REPRO command are followed. This provides a downlevel version recovery when a file or volume is damaged or unusable. Current level recovery is possible when a recoverable catalog is damaged, but the volumes and files are not damaged. The amount of work required to recover to the current level base depends on how recent the backup data was taken and how easily intermediate updates can be reapplied. Factors which affect the timeliness of backup data are activities such as altering the amount of space controlled by the catalog, defining and deleting files, and suballocating space to a VSAM file.

Inaccessible Volume

Cause of Failure: A given 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. If the problem is because of a catalog restore operation, the procedure outlined under "Unusable Catalog" (above) can be used to correct the condition. If the problem is because of physical damage to the volume, recovery depends on the availability of backup copies of the catalogs, volumes, and files, and whether the catalog to which the volume belongs was defined with the RECOVERABLE attribute.

If the catalog is recoverable, then a catalog recovery area (CRA) on the volume contains duplicate catalog information for each file on it. Within this context, this section first discusses recovery of volumes without CRAs, then volumes with CRAs.

Error Conditions:

- Nonrecoverable volume is totally unusable.
- Nonrecoverable volume is partially unusable.
- Recoverable volume is totally unusable.
- Recoverable volume is partially unusable.

Recovery for Nonrecoverable Volume That is Totally Unusable: Recovery can only be effected if you have a volume backup and a catalog volume backup created at the same time (that is, at the same level), or copies of the files created by the REPRO command or by the EXPORT Command. If you have backup volumes:

1. Restore the damaged volume(s).
2. Using a backup level LISTCAT listing, compare it with a current level listing for possible mismatches. See the section "Catalog Entry Mismatches" in this chapter if entry mismatches are detected.
3. Use the EXPORT command or the REPRO command to move the downlevel copies recovered to temporary files.
4. Initialize the volume and reestablish nonVSAM files.
5. If there is a volume mismatch which requires the use of EXPORT command or the REPRO command, use the DELETE command FORCE with the option to clean up the volume and remove the volume entry from the catalog (file entries are marked unusable), then define space on the volume.
6. Use the IMPORT command or the REPRO command to reestablish the files. These files are downlevel and any update applied after the backup was taken has to be reapplied to make the file current.
7. If reestablished nonVSAM files are cataloged, delete and redefine the nonVSAM entries.

If you do not have volume backup, but do have file backup:

1. Initialize the volume and reestablish the nonVSAM files.
2. Use DELETE FORCE to clean up the volume of VSAM ownership and data spaces. This will also remove the volume entry from the catalog and mark file entries unusable.
3. If copies created by the EXPORT command of VSAM files are available, use the IMPORT command to reestablish them.
4. If backup files created by the REPRO command exist, delete the unusable files and redefine them using the DEFINE command and then load the backup copies into the newly created files with the REPRO Command.
5. If reestablished nonVSAM files were cataloged, delete and redefine the nonVSAM entries.

Recovery for Nonrecoverable Volume That is Partially Unusable: If the VSAM files are partially or totally unusable, but the nonVSAM files are accessible, use the above procedure. If the VSAM files are accessible, but the nonVSAM files are not:

1. Recover the VSAM files on the volume using the EXPORT command.
2. Initialize the volume and reestablish the nonVSAM files.
3. Using the IMPORT command, reestablish the VSAM files.
4. If the reestablished nonVSAM files were cataloged, delete and redefine the nonVSAM entries.

Recovery for Recoverable Volume That is Totally Unusable: You can recover if a copy of the volume is available. There are two ways to recover, by using the EXPORTRA/IMPORTRA commands to move and reorganize data, or by using the RESETCAT command to reset the catalog entries equal to the entry information in the volume CRA.

Using RESETCAT:

1. Restore the damaged volume(s).
2. Use the LISTCRA command (with the COMPARE option) to indicate if there are mismatches which may require further processing.
3. Use RESETCAT to reset the catalog entries of the affected volume entry and its files entries to the level of the CRA information on the volume.

Using EXPORTRA/IMPORTRA:

1. Restore the damaged volume.
2. Use LISTCRA (with the COMPARE option) to see if the volume entry is mismatched, or if there are file mismatches.
3. If there are file mismatches only, use the VERIFY command for those files which have mismatched RBAs and EXPORTRA for those with more serious mismatches.
4. If there is a volume mismatch other than a general mismatch, all files on the volume must be recovered using the EXPORTRA command.
5. If there was a volume mismatch which required the use of EXPORTRA, use DELETE (with the FORCE option) to clean up the volume and then use a DEFINE SPACE on the volume. Keep in mind that a DELETE (FORCE) results in the loss of all VSAM data on that volume.
6. Use the IMPORTRA command to reestablish the files recovered by means of the EXPORTRA command.

Recovery for Recoverable Volume That is Partially Unusable: You must have volume or file backup for the inaccessible files. Again, you may use EXPORTRA/IMPORTRA or RESETCAT to recover access to the downlevel data from the backup copy, first using RESETCAT for the *downlevel* portion:

1. Recover the accessible VSAM files on the volume by using the EXPORTRA command.
2. Restore the volume.
3. Use RESETCAT to reset the catalog volume and file entries to the level in the volume CRA.
4. Use IMPORTRA to reestablish the current files.

Using EXPORTRA/IMPORTRA:

1. Recover the accessible VSAM files on the volume by using the EXPORTRA command.
2. Restore the volume
3. Use an EXPORTRA command to recover the previously inaccessible VSAM files that were restored.
4. Use a DELETE command with the FORCE option to clean up the volume and then use a DEFINE SPACE on the volume.
5. Reestablish the recovered files using the IMPORTRA command.

Summary: A given volume that is wholly or partially unusable can be reestablished if backup copies of the data are available. In certain cases, the current version of the data can be extracted from the unusable volume and reestablished in the system.

Quick Recovery

There are some applications, such as online teleprocessing systems which require that file recovery be done as quickly as possible. In this type of situation, normal 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.

The procedure is as follows:

1. Define all files in such a manner that they can acquire *no* additional extents.
2. Use the REPRO command to back up the catalog whenever any file is defined, deleted, or altered.
3. If the catalog controlling these files is lost, do the following:
   a. REPRO the backup catalog into the existing catalog.
   b. Run VERIFY against all files controlled by the catalog.
4. If a volume is lost, do the following:
   a. Restore the lost volume to the backup copy.
   b. REPRO the corresponding backup catalog into the existing catalog, if the volume is the catalog volume.
   c. Run VERIFY against all files on the volume or all files if the volume is the catalog volume.
   d. Update restored files from journalled records.

The restriction that files can acquire no additional extents does not mean that control area splits will not be allowed. As long as there is sufficient unused space in the current extent, control area splits can still occur. For any VSAM data set an overallocation of, for example, 20 cylinders will allow at least 20 control area splits to occur. Be sure to overallocate for the index, also, because at least one new index record will be created whenever a control area split occurs. This requirement is, of course, lessened if the file was defined with the IMBED subparameter of the DEFINE CLUSTER command. It should also be noted that the above procedures minimizes the need for the catalog to be recoverable.

Index

SC24-5145-1

VSE/VSAM Programmer's Reference (File No. S370-30) Printed in U.S.A. SC24-5145-1

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
Programmer's Reference
SC24-5145-1

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.)

SC24-5145-1

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

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 ______________________________

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