GC26-3841-0
File No. S370-30

Systems

OS/VS2 Access Method Services

Release 3

IBM

First Edition (February 1975)

This is the first edition of a new publication that applies to Release 3 of OS/VS2 and to all subsequent releases of that system unless otherwise indicated in new editions or technical newsletters. Another new publication, *OS/VS1 Access Method Services,* GC26-3840, contains corresponding OS/VS1 information. OS/VS2 and OS/VS1 information was previously intermingled in *OS/VS Access Method Services,* GC26-3836; this previous publication is now out of date for both OS/VS2 and OS/VS1. (Information on the Mass Storage System is only for planning purposes until the availability of that product.)

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

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *Virtual Storage Supplement (to IBM System/360 and System/370 Bibliography),* GC20-0001, and the technical newsletters that amend the bibliography, to learn which editions and technical newsletters are applicable and current.

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

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, System Development Division, LDF Publishing—Department J04, 1501 California Avenue, Palo Alto, California 94304. All comments and suggestions become the property of IBM.

ABOUT THIS BOOK

This book describes the use of Access Method Services commands for the Virtual Storage Access Method (VSAM). This publication provides all the VSAM information required to use Access Method Services to establish and maintain data sets. For information on the use of VSAM macro instructions, see *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*

Readers of this book are presumed to have a background in programming.

This book has the following major divisions:

- "Guide to Access Method Services," which lists functions and the Access Method Services commands used to perform them.
- "Introduction," which provides an overview of Access Method Services, including general language considerations, the structure of VSAM data sets, and the use of the 3850 Mass Storage System and the Time Sharing Option (TSO) with VSAM and Access Method Services. Everyone should read this chapter.
- "Optimizing VSAM's Performance," which describes the factors that affect VSAM's performance that can be specified or influenced by values specified in Access Method Services commands.
- "Data Security and Integrity," which describes the security and integrity features of VSAM that are specified as options in Access Method Services commands.
- "Defining Entries," which describes the use of the DEFINE command to define catalogs, data spaces, VSAM data sets, and cataloging nonVSAM data sets.
- "Altering Entries," which describes the use of the ALTER command to modify attributes in catalog entries.
- "Listing Catalog Entries," which describes the use of the LISTCAT command to list catalog entries.
- "Deleting Catalog Entries," which describes the use of the DELETE command to delete catalog entries.
- "Moving Entries," which describes the use of the IMPORT and EXPORT commands to create a backup copy of a data set or to move a data set or user catalog between systems.
- "Converting Catalog Entries," which describes the use of the CNVTCAT command to convert entries in an OS catalog to entries in a VSAM catalog.
- "Verifying End-of-File," which describes the use of the VERIFY command to verify that the true end-of-file is reflected in the catalog.
- "Copying and Printing ," which describes the use of the REPRO and PRINT commands to copy and print data, to reorganize data, to convert data from indexed-sequential or sequential organization to VSAM organization and from VSAM organization to sequential organization, and to copy catalogs.
- "Listing Tape Volumes Mounted at Checkpoint," which describes the use of the CHKLIST command to list tape data sets that were open during a checkpoint.

- "Controlling Command Execution," which describes the use of the IF-DO-END command sequence, the SET command, and the PARM command to control command execution and to specify diagnostic aids and printed-output options.
- "Records Written to the SMF Data Set," which describes the contents of record types 63 and 67; these records conform to the SMF header format and are written to the SMF data set to provide for VSAM catalog recovery, but they are not to be treated as SMF records.
- "Appendix A: Sample Output from Print," which shows output in each of the various formats provided by the PRINT command.
- "Appendix B: Interpreting LISTCAT Output Listings," which provides information on the structure of LISTCAT output and shows sample output.
- "Appendix C: Sample Output from CHKLIST," which shows a CHKLIST listing and explains its contents.
- "Appendix D: JDL DD Parameters to Take Care With," which describes JCL parameters that have either no effect in a VSAM environment or that have a negative effect.
- "Glossary," which defines terms relevant to Access Method Services and VSAM.
- "Index," which is a subject index to the book.

Required Publications

The reader should be familiar with information presented in the following publications:

- *Operator's Library: OS/VS2 Reference (JES2),* GC38-0210, which describes the initial program load (IPL) procedure; an alternate master catalog can be selected at IPL time in a VS2 system.
- *OS/VS Data Management Services Guide,* GC26-3783, which presents basic concepts such as access method, direct-access storage, and the distinction between data-set organization and data-set processing.
- *OS/VS Message Library: VS2 System Messages,* GC38-1002, which provides a complete listing of the messages issued by Access Method Services in a VS2 system.
- *OS/VS Utilities,* GC35-0005, which describes the utility programs available for use with nonVSAM data sets in OS/VS2.
- *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide,* GC26-3838, which describes the macro instructions used to process VSAM data sets and describes how to use an ISAM processing program to process a VSAM data set or an indexed-sequential data set that has been converted to VSAM format.
- *OS/VS2 JCL,* GC28-0692, which describes the JCL parameters for VS2 described in this publication.
- *OS/VS2 Planning Guide for Release 3,* GC28-0667, which describes the special features added to OS/VS2 with Release 3.

- *OS/VS2 System Programming Library: System Generation Reference,* GC26-3792, which describes the use of the DATASET macro to create a master catalog in a VS2 system.

Related Publications

The following publications contain information that is related to this publication:

- *Introduction to the IBM 3850 Mass Storage System (MSS),* GA32-0028, which introduces the storage devices and programming of the Mass Storage System.
- *OS/VS Checkpoint/Restart,* GC26-3784, which explains how to take checkpoints and to restart processing.
- *OS/VS Mass Storage System (MSS) Planning Guide,* GC35-0011, which gives a general description of the programming of the Mass Storage System.
- *OS/VS Mass Storage System (MSS) Services for Space Management,* GC35-0012, which describes the Mass Storage System Access Method Services commands for space management in the Mass Storage System. This publication also describes the responsibilities of a person assigned to manage the space in the Mass Storage System.
- *OS/VS System Management Facilities (SMF),* GC35-0004, which provides a complete description of the SMF records that describe VSAM.
- *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications,* GC26-3819, which provides information about advanced applications of VSAM, which the reader doesn't need to know about to make normal use of VSAM. The topics covered include: gaining access to control intervals; I/O buffering; constructing parameter lists for the macros that generate, modify, and examine control blocks at execution; processing an index as data; and processing a VSAM catalog as a key-sequenced data set.
- *OS/VS2 Access Method Services Logic,* SY35-0010, which describes the functional operation of Access Method Services programs in an OS/VS2 system.
- *OS/VS2 Catalog Management Logic,* SY26-3826, which describes the functional operation of VSAM Catalog Management in an OS/VS2 system.
- *OS/VS2 System Programming Library: Data Management,* GC26-3830, which describes using the PROTECT macro.
- *OS/VS2 System Programming Library: Supervisor,* GC28-0628, which describes the authorized program facility.
- *OS/VS2 TSO Command Language Reference,* GC28-0646, which describes TSO commands.
- *OS/VS2 TSO Terminal User's Guide,* GC28-0645, which describes commands available to the TSO terminal user.
- *OS/VS2 Using OS Catalog Management with the Master Catalog: CVOL Processor,* GC35-0010, which describes the use of control volumes in a VS2 system.

- *OS/VS2 Virtual Storage Access Method (VSAM) Logic,* SY26-3825, which describes the functional operation of VSAM (Record Management, Open, Close, End-of-Volume, and Control Block Manipulation commands) in an OS/VS2 system.

Notational Conventions

A uniform system of notation describes the format of Access Method Services commands. This notation is not part of the language; it simply provides a basis for describing the structure of the commands.

The command-format illustrations in this book use the following conventions:

- Brackets [] indicate an optional field or parameter.
- Braces { } indicate a choice of entry; unless a default is indicated, you must choose one of the entries.
- Items separated by a vertical bar (|) represent alternative items. No more than one of the items may be selected.
- An ellipsis (...) indicates that multiple entries of the type immediately preceding the ellipsis are allowed.
- Other punctuation (parentheses, commas, spaces. etc.) must be entered as shown. A space is indicated by ƀ.
- **Boldface** type indicates the exact characters to be entered. Such items must be entered exactly as illustrated (in upper case, if applicable).
- *Italic* type specifies fields to be supplied by the user.
- Underscored type indicates a default option. If the parameter is omitted, the underscored value is assumed.

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

Release 3

Following is a summary of major changes to Access Method Services for VS2 Release 3.

IBM 3850 Mass Storage System

Parameters in the DEFINE and ALTER commands enable you to specify options for a VSAM user catalog or cluster that is stored on a mass storage volume. See "Mass Storage System" in the "Introduction" for general information about the use of Access Method Services with the Mass Storage System.

Backing up Catalogs

The REPRO command enables you to make a backup copy of the master catalog or of a user catalog. "Backing up a Catalog" in the chapter "Copying and Printing" describes how to unload and reload a catalog. "Backing up the Master Catalog" in the chapter "Data Security and Integrity" further discusses reloading the master catalog. The sections "Updating a Backup Catalog," "VSAM Volume Cleanup," and "VSAM Catalog Cleanup" in the chapter "Data Security and Integrity" discuss subjects related to backing up a catalog.

Backing up Data

You can use the REPRO or EXPORT command to make a backup copy of a data set, as in previous releases. The sections "Backing up Data" and "Ensuring the Accessibility of Secondary Extents" in the chapter "Data Security and Integrity" discuss making and using backup copies of data sets.

LISTCAT Output Options

Parameters in the LISTCAT command enable you to select entries for listing by creation and expiration dates and to list a subset of historical information for each entry.

TSO LISTCAT Output

LISTCAT under TSO prints specific messages for certain errors indicated by IDC0007I with return code 8. LISTCAT under TSO doesn't print blank lines or IDC0001I FUNCTION COMPLETED.

Listing Tape Volumes Mounted at Checkpoint

The CHKLIST command enables you to process the checkpoint data set to identify the tape volumes mounted at the time a checkpoint was taken. See the chapter "Listing Tape Volumes Mounted at Checkpoint" for information about the use of the CHKLIST command to identify tapes mounted at checkpoint.

Miscellaneous Topics

The following topics are now discussed in this book:

In the "Introduction":
"Invoking Access Method Services"
"VSAM Volume Ownership"
"Key Ranges"
"The Master Catalog"
"OS Control Volumes (CVOLs)"
"Transporting User Catalogs"
"Allocating User Catalogs"
"VSAM Catalog Structure"

In the Chapter "Data Security and Integrity":
"Passwords for NonVSAM Data Sets"
"Preformatting Control Areas"

Release 2

Dynamic Allocation

Job control language DD statements are no longer required to cause a data set or volume to be allocated. See "JCL and Dynamic Allocation" in "Introduction" for information on dynamic allocation.

Copying a Catalog

The REPRO command has been expanded to allow you to copy a VSAM catalog for conversion from a volume of one device to another—for example, from 2314 to 3330. See "Copying a Catalog" in the chapter "Copying and Printing" for information about copying a catalog.

Converting a Catalog

A new command, CNVTCAT, allows you to convert OS/VS catalog entries into entries in an existing VSAM catalog. See the chapter "Converting Catalog Entries" for information about the use of the CNVTCAT command to convert OS/VS catalog entries.

System Catalog

In VS2, Release 2, the VSAM master catalog is the system catalog.

Alias Names

The DEFINE command has been expanded to allow you to define aliases for a user catalog or for a nonVSAM data set. See "Defining an Alternate Name" in the chapter "Defining Entries" for information on defining aliases. The DELETE and LISTCAT have been expanded to support aliases.

Generation Data Groups

The DEFINE command has been expanded to allow you to define a generation data group to which generation (NonVSAM) data sets can be attached. See "Defining a Generation Data Group" in the chapter "Defining Entries" for information about defining a generation data group and attaching generation data sets to it. The ALTER, DELETE, and LISTCAT commands have been expanded to support generation data sets; see "Altering Entries," "Deleting Catalog Entries," and " Listing Catalog Entries" for information on changes to these commands.

Page Space

The DEFINE command has been expanded to allow you to define system data sets, called page spaces, that are required by the Auxiliary Storage Manager. See "Defining a Page Space" in the chapter "Defining Entries" for information about defining a page space. The ALTER, DELETE, and LISTCAT commands have been expanded to support page spaces; see "Altering Entries," "Deleting Catalog Entries," and "Listing Catalog Entries" for information on changes on changes to these commands.

Partitioned Data Sets

The ALTER command is the only method by which members of partitioned data sets can be named in VS2, Release 2. See "Altering Entries" for information about renaming PDS members.

Catalog Use

The order in which catalogs are searched has been modified to allow you to use the data-set name as a means of selecting the catalog to be used. If a user catalog that has as its name or alias the same name as the first qualifier of a data-set name, that user catalog is searched. See "Order of Catalog Use: DEFINE" for information about the order in which catalogs are searched.

Generic Names

The LISTCAT, ALTER, and DELETE commands have been expanded to allow the use of a generic name—a qualified name in which one qualifier is represented by an asterisk—to specify a set of entries to be listed, altered, or deleted. See "Listing Catalog Entries," "Altering Entries," and "Deleting Catalog Entries" for the use of generic names in LISTCAT, ALTER, and DELETE commands.

Level Names

The LISTCAT command has been expanded to allow the use of a qualified name in which one or more qualifiers are represented by a single asterisk. See "Listing Catalog Entries" for the use of level names in the LISTCAT command.

GUIDE TO ACCESS METHOD SERVICES

Figure 1 shows a list of tasks that Access Method Services commands can be used to perform. The left-hand column shows tasks that you might want to perform. The middle column more specifically defines the tasks. The right-hand column shows the commands that can be used to perform each task.

Operation		Command
Attach	a user catalog to the master catalog	DEFINE, IMPORT
Catalog	a VSAM data set	DEFINE
	a nonVSAM data set	DEFINE
Change	a data set's description in the catalog	ALTER
	the device type of the volume on which the catalog resides	REPRO
Connect	a user catalog to a master catalog	IMPORT
	an OS/VS CVOL catalog to a master catalog	DEFINE
Convert	a data set to VSAM format	REPRO
	a VSAM data set to sequential format	REPRO
	OS/VS catalog entries to VSAM Catalog entries	CNVTCAT
Copy	a catalog	REPRO
	a data set	REPRO
Create	a backup copy of a data set	REPRO, EXPORT
	a catalog	DEFINE
	an alias	DEFINE
	a VSAM data set	DEFINE
	a generation data group	DEFINE
	a page space	DEFINE
Define	a catalog	DEFINE
	a VSAM data set	DEFINE
	a data space	DEFINE
	a generation data group	DEFINE
	a nonVSAM data set	DEFINE
	a page space	DEFINE
Delete	a catalog	DELETE
	an alias	DELETE
	a VSAM data set	DELETE
	a data space	DELETE
	a generation data group	DELETE
	a nonVSAM data set	DELETE
	a page space	DELETE

Figure 1 (Part 1 of 2). Tasks and Commands

Operation		Command
Disconnect	a user catalog	EXPORT
Enter	a data set in the catalog	DEFINE
List	a password	LISTCAT
	a data set	PRINT
	contents of the catalog	LISTCAT
	tapes mounted at a checkpoint	CHKLIST
Load	records into a data set	REPRO
Modify	a data set's description in the catalog	ALTER
Move	a catalog to another system	EXPORT, IMPORT
	a VSAM data set to another system	EXPORT, IMPORT
	a nonVSAM data set to another system	REPRO
Password Protect	give or change VSAM data set or catalog	DEFINE
	add a password to an existing VSAM data set or catalog	ALTER
	delete a password	ALTER
	list passwords	LISTCAT
	replace a password	ALTER
Print	a data set	PRINT
Recover	from possible loss of data	VERIFY
Release	a user catalog from the master catalog	EXPORT
Rename	a data set	ALTER
Uncatalog	a data set	DELETE
Unload	a data set	REPRO
Verify	end-of-file	VERIFY

Figure 1 (Part 2 of 2). Tasks and Commands

INTRODUCTION

Access Method Services is a service program that is used with VSAM (Virtual Storage Access Method) to establish and maintain data sets. If you use VSAM, you must use the commands provided by Access Method Services. If you are responsible for maintaining the system catalog, you must use Access Method Services commands.

This chapter introduces the commands available through Access Method Services, provides the background information about VSAM that is required to establish and maintain data sets and catalogs, and briefly describes the use of the Time Sharing Option (TSO) with VSAM and Access Method Services.

Access Method Services Commands

Access Method Services enables you to define a VSAM data set and load records into it, convert a sequential or an indexed-sequential data set to the VSAM format, list VSAM catalog information or data-set records, copy a data set for reorganization, create a backup copy of a data set, convert an OS catalog, and make a data set portable from one operating system to another.

You invoke Access Method Services functions by issuing a command and specifying its parameters. You can execute the IDCAMS program and include the command and its parameters as input to the program. You can also call the IDCAMS program from within another program and pass the command and its parameters to the IDCAMS program. TSO (Time Sharing Option) users can execute Access Method Services functional commands from the TSO terminal.

There are Access Method Services commands for:

- Defining, altering, and deleting data sets.
- Listing catalog entries.
- Copying and printing data sets.
- Moving catalogs and data sets from one operating system to another.
- Aiding in recovery from damage to data.
- Converting an OS catalog's entries to entries in a VSAM catalog.
- Listing tape volumes that were mounted at the time of a checkpoint.
- Controlling command execution by testing or setting condition codes.
- Establishing diagnostic-aids and printed-output options.

There are two types of Access Method Services commands: functional commands that are used to request the actual work—for example, defining a data set or listing a catalog—and modal commands that allow the conditional execution of functional commands. TSO users are allowed to use only the functional commands of Access Method Services.

Functional Commands

The functional commands are:

- DEFINE, which is used to create catalog entries for data sets, catalogs, and space that VSAM is to allocate from.
- ALTER, which is used to alter previously defined catalog entries.
- DELETE, which is used to delete catalog entries.
- LISTCAT, which is used to list catalog entries.
- EXPORT, which is used to create a copy of a VSAM data set for backup or to make a data set or user catalog portable so that it can be used on another system.
- IMPORT, which is used to read a backup copy of a VSAM data set or to make a data set or catalog that was previously exported from one system available for use in another system.
- VERIFY, which is used to cause a catalog to correctly reflect the end of a data set after an error occurred in closing a VSAM data set that may have caused the catalog to be incorrect.
- REPRO, which is used to copy data sets, to convert sequential and indexed-sequential data sets to VSAM format, to convert VSAM and indexed-sequential data sets to sequential format, and to copy VSAM catalogs.
- PRINT, which is used to print VSAM, ISAM, or SAM data sets.
- CNVTCAT, which is used to convert the contents of an OS catalog into entries in a VSAM catalog.
- CHKLIST, which is used to identify tape volumes mounted when a checkpoint was taken.

The functional commands that can be used on nonVSAM data sets include DEFINE, ALTER, DELETE, LISTCAT, REPRO, PRINT, CNVTCAT, and CHKLIST.

The functional commands that can be used on page spaces include DEFINE, DELETE, ALTER, and LISTCAT.

Modal Commands

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

- IF, which tests a condition code and executes according to the results of the test. IF is followed by THEN and ELSE clauses which specify alternative actions.
- DO-END, which specify the beginning and ending instructions of an instruction sequence.
- SET, which changes condition codes.
- PARM, which specifies diagnostic-aids and printed-output options.

Language Considerations

All Access Method Services commands have this general structure:

COMMAND *parameters terminator*

COMMAND specifies the type of service requested. The *parameters* further describe the service requested. *Terminator* indicates the end of the command statement.

Commands can begin at or to the right of the left margin. The default margins are 2 and 72 for batch processing jobs. Commands are separated from their parameters by one or more *separators,* that is, one or more blanks, commas, or comments. Comments are strings of characters surrounded by a /* and an */. Comments can contain any characters you desire except an "*/".

Many of the commands can be abbreviated. The abbreviations that are allowed are listed in the discussion of each command.

Parameter Set

A parameter can be either a positional parameter or a keyword parameter. A *positional parameter* is characterized by its position in relation to other parameters. Omission of a positional parameter cannot be indicated by a comma. A *keyword parameter* is a value preceded by a specific character string. For example, in:

VOLUME (25DATA)

VOLUME is a keyword that indicates that the value 25DATA is a volume serial number.

A keyword parameter might have a set of subparameters. The subparameters must follow the same rules as parameter sets in general.

Many of the keywords may be abbreviated. The abbreviations permitted are listed with each parameter. Some keywords are plural in form; they can be coded in the singular form also.

As a group, positional parameters must always appear first in a parameter set. Keyword parameters always follow any positional parameters. The order of keyword parameters is not important.

A parameter or subparameter can consist of a list of similar items. Lists must be enclosed in parentheses unless the list contains only one item. The parentheses can be preceded and followed by blanks, commas, or comments. For example:

DELETE(*entryname* [...] **)**

indicates that if more than one entry is to be deleted, the list of entry names must be enclosed in parentheses. However, if only one entry name is specified, the parentheses are not required.

An item in a list can be a parameter set itself. Each such item, as well as the list of items, is enclosed in parentheses. Given:

OBJECTS ((*entryname* NEWNAME (*newname*))]ъ ...])

the following are valid:

```
OBJECTS( –
  ENTRY1 NEWNAME(NEWNAME1))
```

```
OBJECTS( –
  (ENTRY1 NEWNAME(NEWNAME1)) –
  (ENTRY2 NEWNAME(NEWNAME2)))
```

In the first case, only one entry is to be renamed. The entryname and its new name are enclosed in parentheses. In the second case, each entryname and its new name are enclosed in parentheses and the entire list is enclosed in parentheses.

All parameters and subparameters must be separated from one another by one or more separators (commas, blanks, or comments). The only exception is that parameters that immediately follow a subparameter set that is enclosed in parentheses do not need to be separated from the closing parenthesis.

The values you specify in the parameters can be surrounded by separators. Some values can be longer than a single record. When a value is longer than a single record, you indicate that it is continued by coding a plus sign (+) followed by zero or more blanks. The first non-separator character found in a record following the plus sign is treated as a continuation of the value.

A value cannot contain commas, semicolons, blanks, parentheses, or slashes unless the entire value is enclosed in single quotation marks. A single quotation mark in a field enclosed in single quotation marks must be coded as two single quotation marks.

In some parameters it is necessary to specify a password following the name of a catalog entry or the name of a JCL statement. You do this by coding the name, a slash, and the password. The slash can be surrounded by separators. For example:

```
DELETE PAYROLL/CTLGPAY
```

specifies the password CTLGPAY for the data set PAYROLL.

When you specify a password following a name, you must remember the convention for entering commands. If you code:

```
MYNAME/*WORD*/
```

MYNAME is treated as a name followed by the comment WORD. If you wanted ***WORD*/** to be the password, you must code either:

```
MYNAME/'*WORD*/'
```

or

```
MYNAME/  *WORD*/
```

Notice that the second example contains a blank after the first slash that separates the name from the password.

Continuing Commands

Commands can be continued on several records or lines. Each record or line except the last must have a hyphen or plus sign as the last non-blank character before or at the right margin. The hyphen, which should always be preceded by a blank, indicates continuation of the command. A plus sign indicates continuation of the command and continuation of a value within the command.

Blank records or records ending with complete comments must end with a continuation mark when they appear in the middle of a command and when they appear between the THEN and ELSE clauses of an IF command. Records ending with partial comments must always end with a continuation mark. Also, remember that only blank characters may appear between a continuation mark and the end of the record.

Continuation Cautions

The continuation rules must be used cautiously when modal commands, comments, or blank records appear in the input stream. Blank records or records ending with complete comments must end with a continuation mark when these types of records appear in the middle of a command or when they appear between the THEN and ELSE clauses of an IF command. Records ending with partial comments must always end with a continuation mark.

You must be careful when continuing modal commands so that you don't inadvertently specify a null command. For information on null commands, see "Null Commands."

Remember that only blank characters can appear between a continuation mark and the end of the record.

The following examples show common continuation errors:

- ```
  IF  LASTCC  =  0  -
  THEN
  LISTCAT
  ```

  A continuation mark (hyphen) is missing after the THEN keyword. A null command is assumed after the THEN keyword, and the LISTCAT command is unconditionally executed.

- ```
  IF  LASTCC  =  0  -
  THEN-
  REPRO ...
  /*ALTERNATE PATH*/
  ELSE-
  PRINT ...
  ```

  Because no continuation mark (hyphen) follows the comment, a null command is assumed. The ELSE keyword will not match up with the THEN keyword. Note the correct use of the continuation marks on the other records.

- ```
  PARM  TEXT  (  -  /*COMMENT*/
  TRACE )
  ```

  The PARM command will not be continued onto the second record because characters other than blanks appear between the continuation mark (hyphen) and the end of the record.
  ```

- PARM TEST (TRA+
  /*FIELD CONTINUATION*/
  CE)

  The end of the PARM command is found after the second record because no continuation was indicated. The command is rejected.

Terminator

The terminator indicates the end of the command. The terminator can be either an end of command condition (that is, no continuation mark) or a semicolon. If you use the semicolon as the terminator, the semicolon cannot be enclosed in quotation marks or embedded in a comment. Everything to the right of the semicolon is ignored. If there is information to the right of the semicolon that is continued to another record, all of the information including the continued information is ignored.

For example, if you coded:

```
PARM TEST (TRACE); PARM GRAPHICS (CHAIN(TN))/*COMMENT*/
PRINT ...
REPRO ...
```

Characters following the semicolon terminator are ignored. The continuation mark (hyphen) at the end of the first record causes the PRINT command to also be ignored. The first PARM command and the REPRO command are the only commands that are recognized.

JCL and Dynamic Allocation

When a VSAM data set or volume is to be used, it must be identified. A data set or volume can be identified through JCL or by the data-set name or volume serial number within the command that requires the data set or volume for its execution. If JCL is not used, the data set or volume is dynamically allocated, as required. The data-set name or volume serial number exists in a catalog; the catalog that contains the entry (1) is a user catalog specified in a JOBCAT or STEPCAT DD statement, (2) is the master catalog, or (3) has as its name or alias the first qualifier of the data-set name.

In order to dynamically allocate a volume, it must be mounted as permanently RESIDENT or RESERVED. The PRIVATE and PUBLIC use attributes should be carefully considered when mounting volumes.

When JCL is used to identify a data set, the following information must be included:

- The data-set name.
- The AMP='AMORG' parameter if the JCL also supplies unit and volume serial number information.

Concatenated DD statements are supported for JOBCAT and STEPCAT DD statements and under other circumstances as explicitly specified in the remainder of this publication.

Output Data Sets

The normal output data set for listing is SYSPRINT. The default parameters of this data set are:

- Record format: VBA
- Logical record length: 125, that is, 121+4
- Block size: 629, that is, 5(121+4)+4

Print lines are 121 bytes in length. The first byte is the ANSI control character. The minimum specifiable LRECL is 121 (U-format records only). If a smaller size is specified, it is overridden to 121.

It is possible to alter the above defaults through specification of the desired values in the DCB parameter of the SYSPRINT statement. The record format, however, cannot be specified as F or FB. If you do specify either one, it is changed to VBA.

In several commands you have the option of specifying an alternate output data set for listing. If you do specify an alternate, you must specify DCB parameters in the referenced DD statement. The only exception is when the referenced DD statement contains the parameter SYSOUT. When the statement contains SYSOUT, the defaults described earlier are used. When specifying an alternate output data set, you cannot specify F or FB record formats.

Invoking Access Method Services

When you want to use an Access Method Services function, you invoke the Access Method Services processor. The processor decodes your request (that is, the list of commands and parameters you supply), then calls the routine(s) required to carry out your request.

There are three ways you can invoke Access Method Services:

- As a job or job step
- From a TSO terminal
- From a processing program

As a Job or Job Step

You can use job control language (JCL) statements to invoke Access Method Services. The examples of each command illustrate invocation by JCL. You identify Access Method Services with PGM=IDCAMS or AMS:

```
//YOURJOB     JOB    your installation's job-accounting information
//JOBCAT      DD     DSNAME=YOUR.CATALOG,DISP=SHR
//STEP1       EXEC   PGM=IDCAMS
//STEPCAT     DD     DSNAME=ANOTHER.CATALOG,DISP=SHR
//SYSPRINT    DD     SYSOUT=A
//SYSIN       DD     *
Access Method Services
commands and
their parameters
/*
```

//YOURJOB is required.
The JOB statement describes your job to the OS/VS2 system. Your installation might require you to supply user identification, accounting, and authorization information with the JOB statement's parameters.

//JOBCAT is optional.
The JOBCAT DD statement identifies a user catalog or catalogs that can be used by each of the job's steps. Because the master catalog is always open and available to all jobs on the system, you should not identify it with a JOBCAT DD statement.

//STEP1 is required (that is, an EXEC statement is required).
The EXEC statement invokes Access Method Services to decode and process the input statements (that is, the list of Access Method Services commands and parameters).

//STEPCAT is optional.
The STEPCAT DD statement identifies a user catalog or catalogs that can be used by the job step. If user catalogs are identified with JOBCAT and STEPCAT DD statements, only the catalog(s) identified with the STEPCAT DD statement are used in the job step. Because the master catalog is always open and available to all jobs on the system, you should not identify it with a STEPCAT DD statement.

//SYSPRINT is required.
The SYSPRINT DD statement identifies the output device (usually a printer, terminal, or console) that Access Method Services sends messages and output information to. (See "Output Data Sets" for more details on how to describe an output device other than SYSOUT=A.)

//SYSIN is required.
The SYSIN DD statement identifies the source of the input statements. An input statement, to Access Method Services, is a functional or modal command and its parameters. When you code SYSIN DD *, you identify the immediately following statements as input.

The last input statement is followed by a delimiter statement, which has "/*" in the first two columns. The restrictions that apply to all Access Method Services commands are described in the section "Language Considerations."

From a TSO Terminal

When you use the Time Sharing Option (TSO) to process your data, you can invoke Access Method Services from your TSO terminal. Each time you issue an Access Method Services command as a TSO command, TSO invokes Access Method Services.

You can issue one command at a time, which Access Method Services processes completely before TSO allows you to continue processing. You are allowed to issue the following Access Method Services commands:

ALTER
CNVTCAT
DEFINE
DELETE
EXPORT
IMPORT
LISTCAT
PRINT
REPRO
VERIFY

You identify a VSAM object (that is, a cluster, catalog, page space, or volume) or a nonVSAM data set with its entryname. You cannot use a parameter that identifies a DD statement to describe the VSAM or nonVSAM object, or to identify an alternate output device.

The command you issue is processed immediately. You cannot issue modal commands (IF, THEN, ELSE, DO, END, PARM, or SET) to modify or make conditional the processing of subsequent functional commands.

The restrictions that apply to the coding of Access Method Services commands are described in the "Language Considerations" section. Other TSO restrictions are noted with the descriptions of applicable parameters.

From a Processing Program

A processing program can invoke Access Method Services with the CALL macro. Before the program issues CALL, however, it must initialize appropriate register and parameter list contents.

The register contents follow standard linkage conventions; that is, register 1 contains the address of the argument list, register 13 contains the address of a save area, register 14 contains the address of the return point, and register 15 contains the address of the entry point IDCAMS in Access Method Services.

The contents of the argument list are described in *OS/VS2 Access Method Services Logic.*

VSAM Data Structure

You need to know about the structure and treatment of VSAM data sets to use Access Method Services. VSAM data can be stored in key sequence or in entry sequence. Records in a *key-sequenced data set* are stored in the order defined by the collating sequence of the contents of the key field in each record. Each record has a unique value, such as employee number or invoice number, in the key field. To determine where to insert a new record, VSAM uses an index that pairs the key of a record with the record's location.

Records in an *entry-sequenced data set* are stored without respect to the contents of the records. Their sequence is determined by the order in which they are stored: their entry sequence. A new record is stored after the last record in the data set.

When a data set is created, it is defined, along with its index, if any, in a *cluster.* A key-sequenced data component and its index component make up a cluster. An entry-sequenced data component is also defined as a cluster, even though it does not have an index.

In addition to key-sequenced and entry-sequenced data sets, VSAM provides catalogs. All data—ordinary user data, indexes, and catalogs—however, is physically stored and manipulated in the same way.

VSAM Volume Ownership

VSAM data spaces are defined in a VSAM catalog. That catalog controls (owns) the volumes that contain the data spaces defined in the catalog, including the volume that contains the catalog itself. The catalog also owns candidate volumes for objects defined in the catalog.

All VSAM clusters, page spaces, and data spaces stored on a volume must be cataloged in the catalog that owns the volume. On the other hand, nonVSAM data sets, including OS control volumes (CVOLs), can be cataloged in any VSAM catalog.

VSAM's ownership and usage of space on a volume is indicated in the volume's table of contents (that is, in DSCBs in the volume's VTOC). Two *data-set security bits* in the Format 1 (Identifier) DSCB of each VSAM data space on the volume are set to indicate that a password is required to read or write data in the data space. (They are set whether the object or objects in the data space are actually password-protected or not.) The *ownership bit* in the volume's Format 4 (VTOC) DSCB is set to 1. The ownership bit indicates that the volume is owned by a VSAM catalog, but does not identify the owning catalog. Each VSAM catalog contains a volume entry for each volume it owns. The volume entry describes the volume's characteristics, each extent of the volume's VSAM data space(s), and each VSAM object that uses the volume's space.

In order to take away the ownership of a volume from a VSAM catalog, you must first delete all VSAM objects and then all data spaces on the volume. The DELETE SPACE command deletes the VSAM data spaces on the volume, removes the volume entry from the catalog, and revises the Format 4 DSCB in the volume's VTOC. When you are unable to use the DELETE command because Access Method Services can no longer access the volume (because of damage that resulted from a system or hardware failure), you can reset the ownership bit by using the SUPERZAP program. You may also use ALTER with the REMOVEVOLUMES parameter to take away the

ownership of a candidate volume from a VSAM catalog (whether or not the catalog is accessible).

The VTOC of a volume owned by a VSAM catalog contains the name of each VSAM data space on the volume and of each unique data and index component on the volume. (The Format 1 DSCB contains the name of the object it describes.) For an object whose name is generated by VSAM, the name recorded in the VTOC has one of the following formats:

- For a data space containing suballocated VSAM objects:

  Z999999n.VSAMDSPC.Taaaaaaa.Tbbbbbbb

  where

  n=2 indicates that no catalog resides in the data space

  n=4 indicates that a catalog resides in the data space

  (In a Format 1 DSCB established under VS1 or VS2 Release 1, n=6 also indicates that a catalog resides in the data space.)

  aaaaaaabbbbbbb is the System/370 clock timestamp value

- For a unique data space and the data or index component that occupies it (they have the same name):

  VSAMDSET.DFDyyddd.Taaaaaaa.Tbbbbbbb

  where

  yyddd is the creation date (year and Julian day)

  aaaaaaabbbbbbb is the System/370 clock timestamp value

When a volume owned by a VSAM catalog is first cataloged, a timestamp is written in its Format 4 DSCB and in its catalog entry. Each time the volume is dumped by way of IEHDASDR DUMP/RESTORE, the timestamp in the Format 4 DSCB in the dumped copy is updated. When VSAM mounts a volume, it compares the timestamps. If the timestamp in the VTOC is earlier than the timestamp in the catalog, the volume is considered down-level. Access Method Services does not open a data set on a down-level volume.

How Data Is Physically Stored

VSAM data is stored on direct-access storage devices that are controlled by VSAM. A VSAM data set is independent of the type of device on which it is stored. The records of a data set need not be stored in a continuous area of storage. From the user's point of view, however, the area is continuous, starting at address 0. VSAM addresses a point in the area by its displacement, in bytes, from 0, called its *RBA (relative byte address).* For example, the first record in a data set has RBA 0. The second record has an RBA equal to the length of the first record, and so on, except for intervening control information and free space, space that is left free for future expansion when data is first loaded. RBAs are independent of a data set's being stored in nonadjacent areas on a volume or on several volumes.

The total space of a data set is considered to be divided into a continuous set of areas called *control intervals.* When you retrieve a record, the contents of the control interval in which it is stored are read in by VSAM. A control interval is thus the unit of data transmission between virtual and auxiliary storage.

Each control interval contains control information describing the records that are stored in it. Figure 2 shows how data records and control information are stored in a control interval.

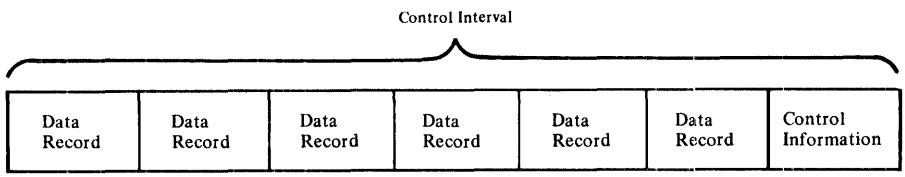


Figure 2. Control-Interval Format

Each control interval in a data set has an RBA: it is the same as the RBA of the first record in the control interval. The RBA of the first control interval in a data set (and of the first record in the control interval) is 0. The RBA of the second control interval in the data set (and of the first record in the control interval) is equal to the length of the first control interval.

The length of control intervals in a given data set is fixed. All of them are the same length, and the length cannot be changed without defining a new data set and copying the old into it. From data set to data set, the optimum length of a control interval varies with the type of storage device (number of bytes of storage per track), the size of your records, and the amount of virtual storage that you provide for buffers.

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

When space for a data set is allocated by tracks, a control interval may not span tracks. The maximum size of a control interval is the largest multiple of 512 that will fit on a track of the device used to store the data set, and a whole number of control intervals must fit on a track. The reason for this limitation is that, with track allocation, VSAM would have to use more than one channel program to gain access to the contents of a control interval that spanned tracks, and VSAM could not ensure data integrity.

VSAM uses track allocation when you define a data set, if you specify track allocation or specify record allocation for a very small number of records.

VSAM gets and manages space on a storage volume in units called *data spaces.* One or more data sets are stored in a data space. Conversely, a data set may be stored in one or more data spaces, on one or more storage volumes. Data spaces and data sets may be extended beyond their original size. Data spaces are extended by whole numbers of tracks or cylinders; VSAM extends them automatically as more space is needed. A data space can be as large as a volume; it can be extended up to 15 times. A data set can

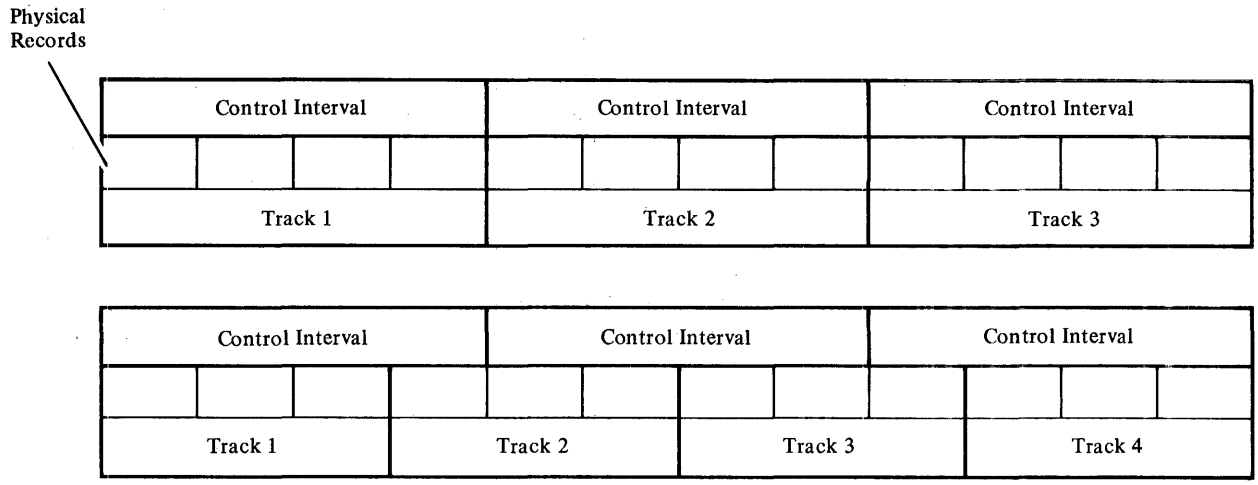


Figure 3. Control Intervals and Physical Records

contain up to 2^{32} (approximately 4,290,000,000) bytes of information (data records and control information); it can be extended up to 127 times.

The collection of control intervals that make up the area of a data set is divided into groups of control intervals called *control areas.* A control area is the unit of space that VSAM preformats for data integrity as records are added to a data set. In a key-sequenced data set, VSAM distributes unused control intervals throughout a data set as a percentage of free control intervals per control area.

The number of control intervals per control area of a data set is fixed by VSAM, with a minimum of 3. If 50 were the number chosen, for example, the first 50 control intervals would be the first control area; the next 50 would be the second control area, and so on. Whenever the space for a data set is extended, it is extended by a whole number of control areas.

Key-Sequenced Data

A key-sequenced data set includes an index component, which is a mechanism for keeping track of records. An index component consists of records (one per control interval) that relate key values to the relative locations of the records in the data component. A key is put into the index from a record's key field, whose size and position are the same for every record in the data set, and whose value cannot be altered. VSAM uses the index to locate a record for retrieval and to locate the collating position for insertion.

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

An entry in an index-set record consists of the highest key that an index record in the next lower level contains, paired with a *vertical pointer* to that index record. An entry in a sequence-set record consists of the highest key in

a control interval, paired with a vertical pointer to that control interval. Not all data records have sequence-set entries, for there is only one entry for each control interval in the data set.

For direct access by key, VSAM follows vertical pointers from the highest level down to the sequence set to find a vertical pointer to data. For sequential access by key, VSAM refers only to the sequence set; it uses a *horizontal pointer* in a sequence-set record to get from that sequence-set record to the one containing the next key in collating sequence so it can find a vertical pointer to data. Figure 4 shows vertical and horizontal pointers and the relationship between the levels of an index and the control areas of the key-sequenced data set. The highest-level index record (A) controls the entire next level (records B through Z); each sequence-set index record controls a control area.

VSAM uses free space to insert or lengthen key-sequenced records. When you define a data set, you can cause free space to be distributed in two ways: as entirely free control intervals in each control area and as unused space within a control interval containing records. Figure 5 shows how free space can be distributed.

Each entirely free control interval has an index entry giving its location. When free space in a used control interval is insufficient to insert or lengthen a record, a free control interval is used, and its index entry changed from a free-space indicator to a high-key indicator. This action is called a *control-interval split.* The control interval without sufficient space for an insertion or a lengthening is relieved of some of its records also, so that the old and the new control intervals share the total of their free space.

Figure 6 illustrates a control-interval split and shows the resulting free space available in the two affected control intervals. Because the number of records in the first control interval is reduced, subsequent insertions revert to the simpler case, instead of becoming more complex. The figure shows that some of the records in the control interval that is too full for insertion are moved to

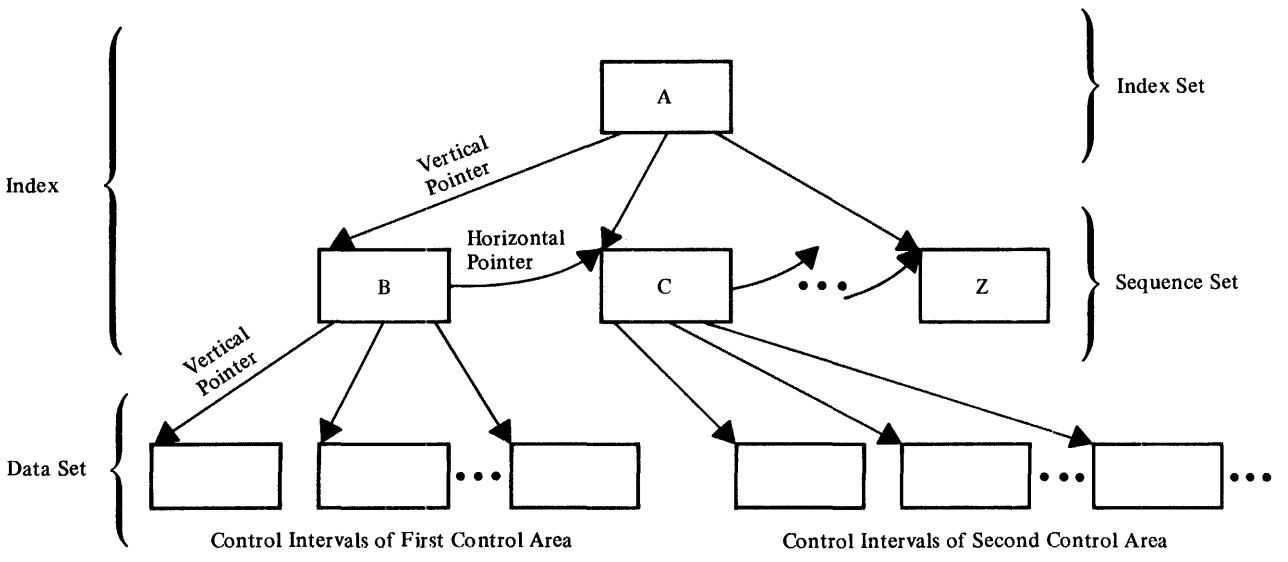


Figure 4. Index Levels and Data

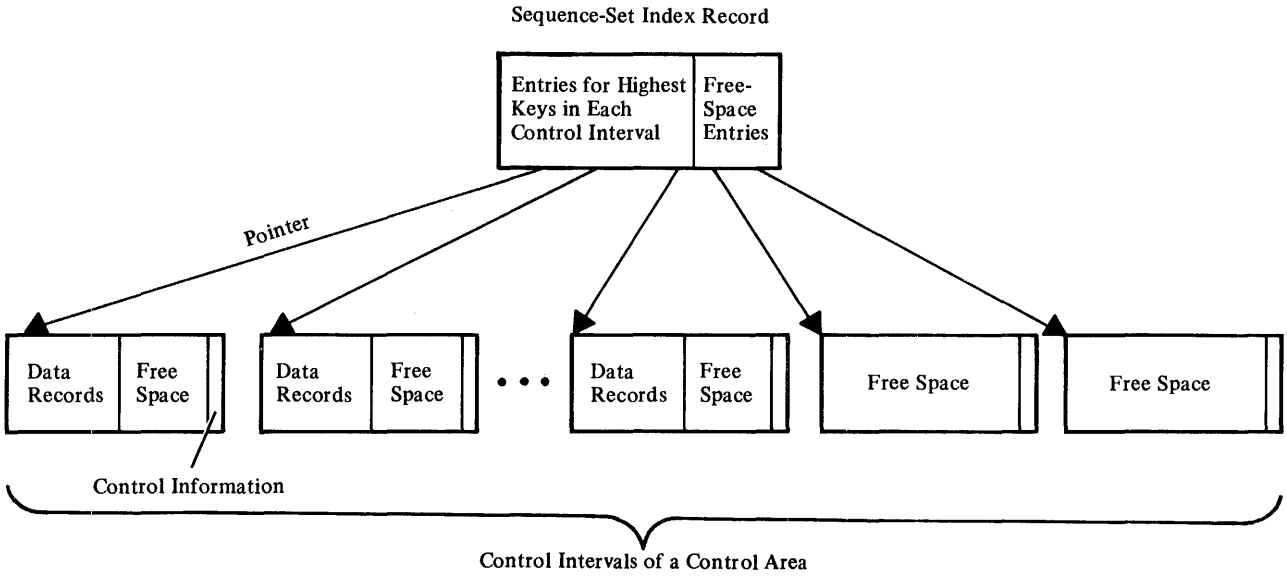


Figure 5. Free-Space Distribution

a free control interval, and the new record is inserted into the control interval dictated by the key sequence.

If the control intervals involved in a split are not adjacent, the entry sequence of data records (their sequence by ascending RBAs) is no longer the same as their key sequence. In Figure 6, the entry sequence of the records in the last three control intervals on the right is: 55, 56, 57, 60, 61, 58, 59. But the sequence-set index record reflects the key sequence, so that, for keyed sequential requests, the data records are retrieved in the order: 55, 56, 57, 58, 59, 60, 61.

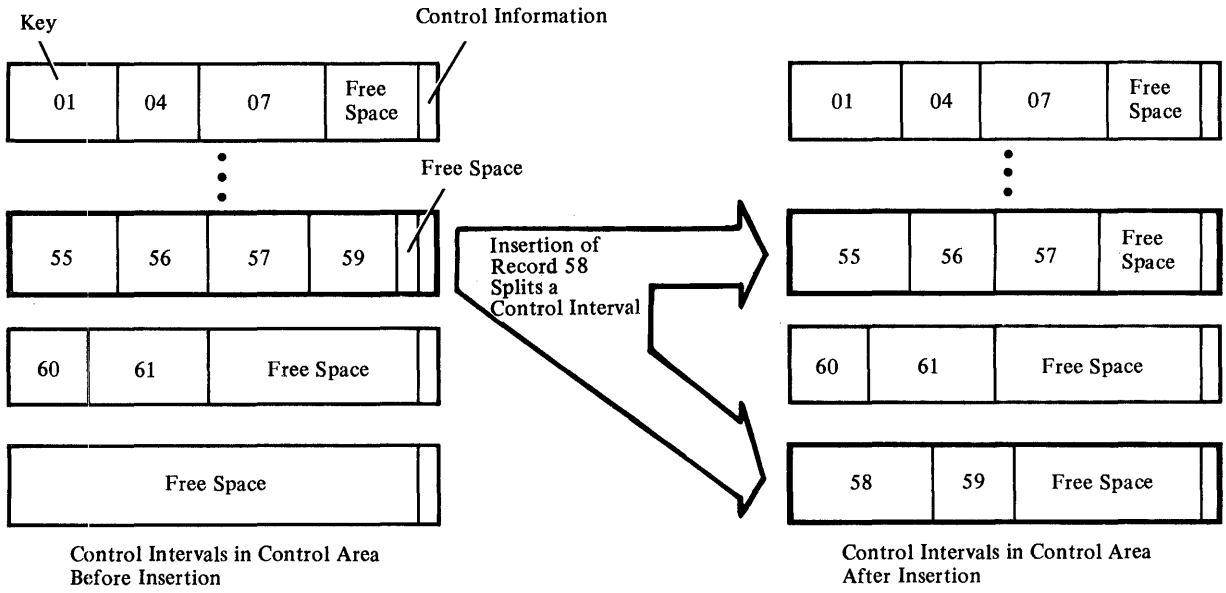


Figure 6. Control-Interval Split

If no free control intervals remain in a control area when one is needed, VSAM sets up a new control area at the end of the data set. The old control area is relieved of enough of its records that the old and the new control areas share the records and the free space. This action is called a *control-area split.* A control-area split automatically provides free control intervals: it facilitates the insertion of a record into a data set that has no free space. Since about half of the control intervals of each of these control areas are now free, subsequent insertions won't require control-area splitting. Splitting should be an infrequent occurrence for data sets with sufficient distributed free space; splitting a control area does make it possible, however, to insert records into a key-sequenced data set without previously distributed free space.

The free space in a control interval lies between the records at the beginning and the control information at the end. VSAM combines the space vacated by a deleted or a shortened record with the free space already in a control interval, if any. This reclaimed space is available for future insertions or lengthenings.

The RBAs of records can change when free space is used or reclaimed. If a record is inserted or lengthened, any succeeding records in the control interval are moved to the right into free space and their RBAs are changed. Conversely, if a record is deleted or shortened, any succeeding records in the control interval are moved to the left and their RBAs are changed so that the space vacated can be combined with the free space already in the control interval.

Key Ranges

When you define a key-sequenced cluster, you can specify KEYRANGES to divide the cluster's data records into groups according to ranges of keys. You can then process each group independently of the others.

The key ranges can cover the whole possible range (for example: A-I, J-R, and S-Z) or leave gaps (for example: B-F, J-R, and X-Z). However, one key range cannot overlap another key range. You might divide a large number of records among several volumes so that there is enough free space on each volume to allow for additional records. You can process the records on each volume separately or you can process as many volumes together as your program requires.

The number of volumes can be less than, equal to, or greater than the number of key ranges. When there are more volumes than key ranges, each key range is allocated space on a separate volume; the remaining volumes can be used as overflow volumes for any key range. When there are fewer volumes than key ranges, as many key ranges as possible are allocated space on separate volumes; the remaining key ranges are allocated space on the last volume.

When space is allocated to a cluster with key ranges, each volume that contains a key range is allocated the primary amount of space specified for the cluster.

Entry-Sequenced Data

Entry-sequenced records are sequenced according to the order in which they are added. Each new record is stored at the end; none is inserted. Records are not deleted, shortened, or lengthened; they are not moved from one location to another. Once a record is added to an entry-sequenced data set, it stays there and keeps its original RBA. An entry-sequenced data set is essentially a sequential data set, but one whose records can be retrieved at random by direct access and can be updated. The search argument for direct retrieval is a record's RBA.

An entry-sequenced data set is appropriate for applications that require no special ordering of data by the contents of a record. It is appropriate for a log or a journal, since its order corresponds to the chronology of events. To retrieve records directly from an entry-sequenced data set, you must keep track of the records' RBAs and somehow associate RBAs with the contents of records.

Figure 7 summarizes the differences between key-sequenced data and entry-sequenced data.

Key-Sequenced Data	Entry-Sequenced Data
Records are in collating sequence by key field	Records are in the order in which they are entered
Access is by key through an index or by RBA	Access is by RBA
A record's RBA can change	A record's RBA cannot change
Distributed free space is used for inserting records and changing their length in place	Space at the end of the data set is used for adding records
Space given up by a deleted or shortened record is automatically reclaimed	A record cannot be deleted, but you can replace it with a record of the same length

Figure 7. Comparison of Key-Sequenced and Entry-Sequenced Data

VSAM's Use of Catalogs

Most uses of Access Method Services involve doing something to the VSAM catalogs. For example, establishing a VSAM data set involves creating an entry in a catalog; deleting a VSAM data set involves removing an entry from the catalog; and moving a VSAM data set from one system to another involves moving an entry from a VSAM catalog in one system to a VSAM catalog in another system. The use of Access Method Services with VSAM requires an understanding of how VSAM uses catalogs.

VSAM uses catalogs as a central information point for all VSAM data sets and the direct-access storage volumes on which they are stored. There are two kinds of catalogs—master catalogs and user catalogs. A master catalog is required with VSAM, and any number of user catalogs are optional. Each VSAM catalog contains records that describe itself. The nucleus contains a pointer to the VSAM master catalog, which is the system catalog. User catalogs are pointed to by the VSAM master catalog.

Catalogs provide VSAM with the information to allocate space for data sets, verify authorization to gain access to them, compile usage statistics on them, and relate RBAs to physical locations.

For a VSAM data set to exist, it must be defined in a VSAM catalog. That is, you must enter in the catalog a data set's name and other facts about it by using Access Method Services to define it.

All VSAM data sets on a volume must be cataloged in the same catalog, either the master catalog or a user catalog. A data set is defined in only one catalog.

When you execute a program to process a data set, catalogs are searched to find out which volume(s) the data set is stored on, unless you give volume serial number(s) by way of JCL.

The order in which catalogs are searched depends on whether catalogs are specified for the current job step (STEPCAT) or job (JOBCAT), or the name of the data set, and on whether the catalog is to be used to define a new entry or searched for existing entries. See "Order of Catalog Use" in the chapters "Defining Entries," "Altering Entries," "Listing Entries," and "Deleting Entries" for an explanation of how catalogs are used.

Note: An unqualified name and a qualified name cannot exist in the same catalog if the first qualifier of the qualified name is the same as the unqualified name. For example, DATA and DATA.PAYROLL cannot exist in the same catalog.

Figure 8 illustrates how data sets in a VS2 system can be divided up for cataloging among the VSAM master catalog, VSAM user catalogs, and OS control volume catalogs (CVOLs).

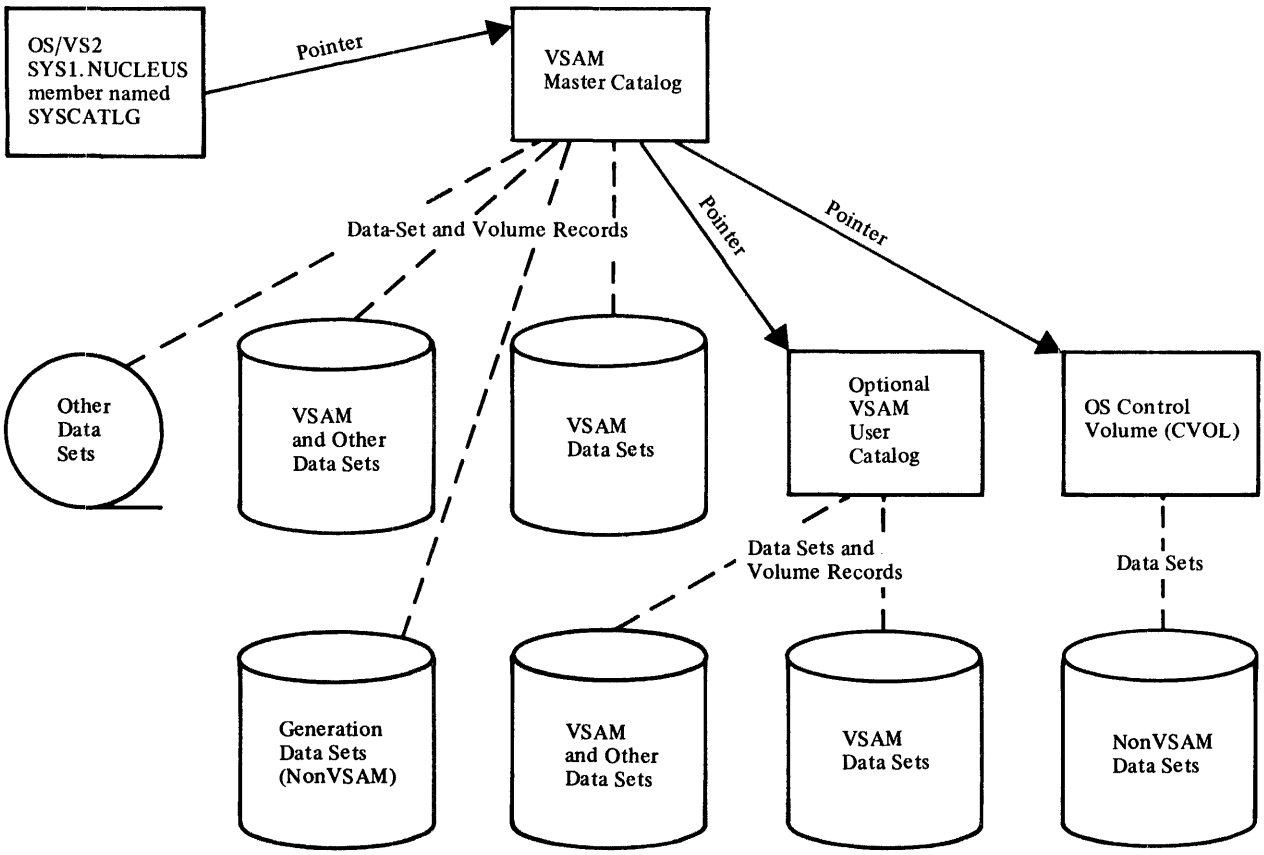


Figure 8. Catalog Relationships

Information Contained in a Catalog

A VSAM catalog has records describing:

- VSAM and nonVSAM data sets.
- Direct-access volumes in terms of the allocation of data spaces and the location of available space.
- User catalogs.
- Alias names.
- Generation data groups.

The nonVSAM data-set information is restricted to the location of the data set.

Data-Set Information

A catalog contains data-set information required to make the connection between the RBA of a record in the data set and its physical location in terms of a storage volume's physical attributes. Besides the type of storage device and a list of volume serial numbers (which the system catalog contains for each data set cataloged in it), a VSAM catalog keeps other information about VSAM data sets, including:

- Each extent of the data set.
- Statistics on the results of operations performed on the data set and its records, such as the number of insertions and deletions and the amount of free space remaining
- Attributes of the data set determined when it was defined, such as control-interval size, physical-record size, number of control intervals in a control area, and, for a key-sequenced data set, location of the key field
- Password-protection information, consisting of passwords and operator prompting codes.
- A connection between the cluster, data, and index components of VSAM data sets or catalogs.
- Information used to determine whether a data or index component has been processed by itself.

Volume Information

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

- The volume serial number, device type, and device characteristics of each owned volume.
- The physical extents of data spaces on a volume.
- The names of data sets that are using space on the volume.
- The location and size of free areas available for allocation to data sets.

The Master Catalog

The system catalog in OS/VS2 is a VSAM master catalog. When you use SYSGEN to generate your system, you set up the master catalog. You can cause a previously created VSAM catalog to be used or a new one to be built during SYSGEN. See *OS/VS2 System Programming Library: System Generation Reference* for more information about system generation.

You should not put the master catalog and VSAM objects cataloged in it on the IPL (initial program load) volume—if the master catalog has to be recovered, then restoring the master catalog will not affect the IPL volume. SYS1.NUCLEUS, a nonVSAM data set on the IPL volume, contains a member SYSCATLG, which points to the volume that contains the master catalog.

The master catalog is the only catalog referenced during nucleus initialization. Therefore, all system data sets, data sets entered in LNKLSTxx, paging data sets, and libraries specified in the JES (job entry subsystem) start procedure must be cataloged in the master catalog, and not in a user catalog or an OS control volume (CVOL).

You cannot move the master catalog from one system to another by using the EXPORT and IMPORT commands. You can, however, use the REPRO command to make a copy of the master catalog, then move the master catalog's copy to another system.

A master catalog cannot be used simultaneously as the master catalog for two OS/VS2 systems. However, one system's master catalog can be used as a user catalog on another system that includes VSAM when the catalog is on a shared direct-access device. Passwords should be assigned to each of the page spaces and system data sets defined in the catalog to prevent their accidental or unauthorized use.

OS Control Volumes (CVOLs)

The OS/VS2 system catalog is a VSAM master catalog. In addition to the master catalog, your system can include VSAM user catalogs and OS control volumes (CVOLs). You can have one of the following four sets of catalogs:

- A master catalog only
- A master catalog and one or more user catalogs
- A master catalog and one or more OS CVOLs
- A master catalog and one or more user catalogs and OS CVOLs

The master catalog points to each user catalog and OS CVOL in the system. A user catalog or OS CVOL does not point to another user catalog or OS CVOL.

The master catalog serves the same functions in an OS/VS2 system that the system catalog served in previous systems. User catalogs and OS CVOLs provide for:

- The portablility of catalogs between systems—user catalogs can be moved only to other VS1 and VS2 systems that include VSAM; OS CVOLs can be moved to VS1 and VS2 systems that might not include VSAM. (The next section discusses exporting and importing user catalogs.)
- The ability to mount, as needed, the volumes containing catalogs.

- Minimizing the effect of errors or damaged entries in a catalog.
- Performance improvements by reducing contention for a catalog.

However, you cannot use Access Method Services DEFINE or ALTER with an OS CVOL (that is, in particular, you cannot catalog a VSAM object in an OS CVOL). You can use CNVTCAT to convert an OS CVOL's entries to VSAM catalog entries, but there is no corresponding way to convert VSAM catalog entries to OS CVOL entries.

Many users choose to use OS CVOLs initially, then convert the CVOLs to user catalogs when their system is running smoothly. Whether you use an OS CVOL or a user catalog, nonVSAM data sets are cataloged and uncataloged by your specification of the DD statement's DISP parameter (see *OS/VS2 JCL* for more details on DD statements and job control language). VSAM objects are cataloged and uncataloged in VSAM catalogs by using Access Method Services commands.

Transporting User Catalogs

You can move a user catalog from one VS2 system to another (that is, to a VS1 or VS2 system that includes VSAM) to make the objects cataloged in the user catalog available to the other system. The user catalog might own only the volume on which it resides. In this case, the volume can be moved as a single unit to another system. When the catalog owns more than one volume, either all volumes owned by the catalog must be moved to the other system or the catalog's ownership of the volumes that are not moved must be taken away before the catalog is moved.

You can use the EXPORT command with the DISCONNECT parameter to remove a user catalog from a VS2 system. The EXPORT command deletes the user catalog's connector entry in the master catalog. It also deletes all alias entries that point to the user catalog's connector entry. The catalog, its cataloged objects, and its volume(s) become unavailable to the old system.

When the user catalog has aliases, you should use the LISTCAT command to list the aliases before you issue the EXPORT command. You can use the list of aliases to help you define the alias entries in the new system's master catalog.

You use the IMPORT command with the CONNECT option to make the exported catalog available to the new VS1 or VS2 system. The IMPORT command builds a connector entry in the master catalog that identifies the user catalog's volume. You issue a DEFINE ALIAS command to build an alias entry in the master catalog for each of the user catalog's aliases.

Allocating User Catalogs

User catalogs can be allocated dynamically or can be allocated directly with a JOBCAT or STEPCAT DD statement. A previous section, "JCL and Dynamic Allocation," discusses dynamic allocation of catalogs and cataloged objects. The JOBCAT DD statement identifies a catalog or catalogs that are to be available to all steps of a job. The STEPCAT DD statement identifies a catalog or catalogs that are to be available to only one of the job steps, and are to be used instead of the JOBCAT catalog(s) during that job step. If a JOBCAT or STEPCAT DD statement is supplied, the system first searches the catalog(s) identified by the DD statement and defines data sets created in the job or job step in the first catalog identified (unless some other catalog is specified in the Access Method Services command's CATALOG parameter).

You do not have to provide a STEPCAT or JOBCAT DD statement to access a cluster or a nonVSAM data set. Instead, you can use a qualified name for the object, where the first qualifier is the name or alias of the catalog (user catalog or OS CVOL) in which the object is defined.

When the catalog is not identified with a DD statement or explicitly named in the CATALOG parameter, the system searches the master catalog for the specified entryname. If the entry is not found, the system uses the entryname's first qualifier to locate the catalog.

Using qualified names to identify a catalog minimizes the use of JOBCAT and STEPCAT DD statements for your jobs. However, it requires the master catalog to be searched to locate the catalog.

Note: You must supply a JOBCAT or STEPCAT DD statement to allocate the catalog when you:

- Delete a user catalog
- Copy a user catalog
- Use IEHDASDR to dump a volume owned by a user catalog

VSAM Catalog Structure

A VSAM catalog is a VSAM key-sequenced cluster with 44-byte keys and 512-byte control intervals, in both the data and the index components.

The *data component* is divided into two key ranges. The index level of the index component is physically imbedded between the low-key range and the high-key range. Each record in the sequence set of the index component is replicated and placed on a track at the beginning of the control area that it governs.

Every object defined in a catalog has one record in the high-key range and one record in the low-key range (plus, possibly, one or more extension records there).

A record in the *high-key range* is 47 bytes long (up to ten per control interval). It contains the full 44-byte name of the cataloged object and the control-interval number of a record in the low-key range.

A record in the *low-key range* is 505 bytes long (one per control interval). It describes and gives the location of the cataloged object. The key of a record in the low-key range contains a byte of binary 0s, the 3-byte number of the control interval that contains the record, and 40 bytes of control information. The first four bytes can be used as a generic key. There are numerous types of records in the low-key range; each type is described in detail in *OS/VS2 Catalog Management Logic.*

Mass Storage System (MSS)

The 3850 Mass Storage System can be used with OS/VS2 to store a massive amount of data online to the operating system. It is described in the *Introduction to the IBM 3850 Mass Storage System (MSS)* and the *OS/VS Mass Storage System (MSS) Planning Guide.*

When you have the Mass Storage System, you can define VSAM data spaces, user catalogs, and data sets and nonVSAM data sets on mass storage volumes. The master catalog and VS2 page spaces cannot be stored on mass storage volumes.

A VSAM catalog may have defined in it both objects stored on direct-access storage volumes and objects stored on mass storage volumes. In particular, the data component of a key-sequenced cluster may be stored on a mass storage volume and the index component on a direct-access storage volume, or *vice versa.* However, if the sequence set is imbedded in the data, both components must be stored on a mass storage volume or on a direct-access storage volume. For example, both components of a user catalog must be stored on a mass storage volume or on a direct-access storage volume.

Space for an object larger than one cylinder that is stored on a mass storage volume should be allocated in cylinders to optimize data transferral (staging and destaging) between mass storage and direct-access storage.

Data stored in the Mass Storage System is *staged* from mass storage to a direct-access storage staging drive when the object to which the data belongs is opened or when the data is requested. It is *destaged* from the staging drive to mass storage when the object is closed. (The *Introduction to the IBM 3850 Mass Storage System (MSS)* tells what direct-access storage devices can be used for staging.)

Access Method Services for the Mass Storage System provides a set of commands for the management of mass storage volumes. These commands are described in *OS/VS Mass Storage System (MSS) Services for Space Management.*

Access Method Services for managing VSAM catalogs provides parameters for options of the Mass Storage System in the DEFINE and ALTER commands, which enable you to specify how a VSAM data set that is stored on a mass storage volume is to be staged and destaged and how a user catalog that is stored on a mass storage volume is to be destaged. These parameters are described in this book.

The staging attributes, BIND, CYLINDERFAULT, and STAGE, affect performance. BIND causes an object to be staged when it is opened and to be retained (bound) in direct-access storage. CYLINDERFAULT causes portions of an object to be staged only as needed during processing. STAGE is a compromise: the object is staged when it is opened, but not retained in direct-access storage. When the staging activity of other objects is light, STAGE can achieve results similar to BIND: the data might remain in direct-access storage, available for requests for access without staging. When the staging activity of other objects is heavy, STAGE can achieve results similar to CYLINDERFAULT: the data might not remain in direct-access storage and might have to be restaged when needed.

A user catalog that is stored on a mass storage volume is always staged and bound when it is opened—that is, it is retained in direct-access storage until it is closed. Not binding a user catalog might degrade performance.

The destaging attributes, DESTAGEWAIT and NODESTAGEWAIT, affect data integrity. DESTAGEWAIT causes VSAM to return control to the program that closes an object synchronously—only after destaging is complete. VSAM can notify the program whether destaging was successful. NODESTAGEWAIT causes VSAM to return control to the program asynchronously—as soon as the object is closed, but before it has been destaged.

A failure in destaging causes a message to be written to the operator and to the messages (SYSPRINT) data set. With DESTAGEWAIT, an error code as well is returned from CLOSE to the processing program. But there are no recovery procedures for a program to undertake: one use of DESTAGEWAIT is for a processing program to periodically issue a temporary CLOSE of a bound object to cause it to be destaged at various checkpoints. The processing program can terminate processing if a failure occurs in destaging. The last copy destaged would be the copy to fall back to, pending correction of the error that caused the failure.

A data component that is defined with the ERASE parameter and stored on a mass storage volume, is overwritten with binary 0s on the staging drive after it is destaged.

Time Sharing Option (TSO)

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

- Execute Access Method Services
- Execute a program to call Access Method Services

When TSO is used with Access Method Services, the following differences must be observed:

- TSO allows the initial characters of a keyword to be supplied as an abbreviation of the keyword. The only restriction is that enough initial characters must be supplied to make the keyword unique. TRACKS, for example, could be abbreviated TR, TRA, or TRAC, because no other keyword can be abbreviated in the same way.
- In addition, the abbreviations described in the remainder of this publication (for example, TRK for TRACKS) are acceptable to TSO.
- When a parameter's value consists of a list of one or more parenthesized parameter sets, the outer parentheses surrounding the list are always required. For example, if lowkey and highkey form a parameter set that can be repeated several times, then the outer parentheses are required even when just one parameter set is specified, as follows:

  KEYWORD((*lowkey highkey* **))**

- A name can be specified in quotation marks or not in quotation marks. Under TSO, however, a prefix (for example, the userid) is added to a name that is not in quotation marks. The prefix becomes the first qualifier in the name. If the name is a volume serial number (as it may be in the DELETE command, for example), enclose it in quotation marks; if you don't, a prefix is added to the volume serial number.
- Under TSO, a volume serial number can consist of alphabetic, national, numeric, or special (hyphen only) characters; if any other characters are

used, the volume serial number cannot be used as an entry name under TSO.

- Under TSO, a password can be supplied with any entry name; the password is ignored if not required.
- The modal commands, used to control execution (IF-THEN-ELSE command sequence, DO-END command sequence, SET, and PARM), are not allowed under TSO.
- The CHKLIST command is not allowed under TSO.
- Under TSO, the user is prompted to complete a fully-qualified name. The user is also prompted to supply required, but omitted, parameters.

For details about the format of a displayed catalog entry (resulting from a LISTCAT command) for a TSO user, see "Appendix B: Interpreting LISTCAT Output Listings."

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

OPTIMIZING VSAM'S PERFORMANCE

The options that determine or influence VSAM's performance are specified in the Access Method Services DEFINE command when a data set is created and in the ACB or GENCB macro when a processing program prepares to open a data set.

Control-Interval Size

Control-interval size, which can be specified for entry- and key-sequenced data sets, affects record-processing speed and requirements for virtual and direct-access storage, as follows:

- As the size of your data records increases, you may need larger control intervals, because a data record cannot span control intervals.
- As control-interval size increases, more buffer space is required in virtual storage for each control interval.
- As control-interval size increases, fewer I/O operations (control-interval accesses) are required to bring a given number of records into virtual 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 control-interval size increases relative to data-record size, especially with variable-length records. (Free space in a control interval isn't used if there isn't enough for a complete data record.)

You can let the system 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. The size you specify must, however, fall within acceptable limits determined by the system, or the DEFINE will fail. These limits depend on the maximum size of the data records, which you specify by the required RECORDSIZE parameter of the DEFINE command, and on the smallest amount of virtual-storage space your processing programs will provide for I/O buffers, which you specify by the optional parameter BUFFERSPACE.

In the first place, the size of a control interval must be a multiple of 512 bytes, because a control interval is a whole number of physical records and physical-record size is 512, 1024, 2048, or (only for the IBM 2305 Fixed Head Storage or the IBM 3330 Disk Storage) 4096 bytes. (The physical record size of 4096 bytes does not apply to the IBM 2314 Direct Access Storage Facility.) 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, 2048, 2560, ..., 8192, 10240, 12288, ..., 32768. A control interval in an index is the same size as a physical record, and its size is therefore restricted to 512, 1024, 2048, or 4096.

The system uses the largest physical-record size it can for a given control-interval size. For example, consider data-control-interval sizes 1024, 1536, and 2048. For 1024, the system uses physical-record size 1024; for 1536, it uses 512; for 2048, it uses 2048.

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

If you specify TRACKS for space allocation in the DEFINE command, control-interval size is further limited: with track allocation, a control interval may not span tracks. Therefore, its maximum size is the largest multiple of 512 that will fit on a track of the device used to store the data set.

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. The minimum amount of control information in a control interval is 7 bytes. Therefore, a control interval must be at least 7 bytes larger than the largest record in the component.

Because VSAM transmits the contents of a control interval between direct-access storage and virtual storage, the amount of space allowed for I/O buffers limits the size of a control interval. The BUFFERSPACE parameter of the DEFINE command indicates the smallest amount of virtual-storage space a processing program will provide for buffers.

BUFFERSPACE, if you specify it, limits control-interval size to values such that the buffer space can hold at least two data control intervals and one index control interval. If you don't specify BUFFERSPACE, control-interval sizes are set independently, and the buffer-space value is then set equal to the size of two data control intervals and one index control interval.

If you don't specify a size for data control intervals, the system uses 2048 as the size, if possible. For a key-sequenced data set, after control-interval size has been set, the system determines the number of bytes to be reserved for free space, if any. For example, if control-interval size is 4096, and the percent of free space in a control interval is 20%, approximately 820 bytes is reserved.

With a key-sequenced data set, if you don't specify a size for index control intervals, the system uses 512, if possible. After the system determines the number of control intervals in a control area (see the next section), it estimates whether an index record is large enough to handle all of the control intervals in a control area. If not, the size of an index control interval is increased, if possible. If it's not possible, the number of control intervals in a control area is decreased.

To find out what values are actually set in a defined a data set, you can issue the Access Method Services LISTCAT command.

If neither control-interval size nor bufferspace is specified and the maximum record size is specified to be 200 bytes: the system sets data control-interval size to 2048 bytes and buffer space to 4096 bytes.

For a key-sequenced data set, the system sets index control-interval size to 512. The maximum data-record size specified in RECORDSIZE is 200 bytes. 4000 bytes is specified for BUFFERSPACE.

If control interval size is not specified, the maximum record size is specified to be 200 bytes, and buffer space is specified to be 4000 bytes: 2048 bytes cannot be used for data control-interval size, because 2 times 2048 is greater than 4000. The system uses 1536 instead.

For a key-sequenced data set, index control-interval size is set to 512. (2 times 1536, plus 512, is less than 4000.) The maximum data-record size specified in RECORDSIZE is 2500 bytes. A data control-interval size of 2500 is specified for the data component.

The system adjusts the 2500-byte control-interval size to the next higher multiple of 512: 2560.

Note that a BUFFERSPACE of 2 times 2500 is not sufficient. If BUFFERSPACE=2500 were specified, the specified control-interval size would be decreased to 2048 bytes. However, in this case, 2048 doesn't accommodate the maximum-sized data record of 2500 bytes, and the DEFINE won't work.

Control-Area Size

A control area is never larger than one cylinder. The size of a control area depends on the size of control intervals in the index component and on device type. 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 required for sequential access. Generally, the greater the size of the control area, the better performance and space utilization is.

Distributed Free Space

You can specify in the DEFINE command the percentage of free space in a control interval and the number of free control intervals in a control area. This free space improves performance by reducing the likelihood of control-interval and control-area splits; avoiding control-interval and control-area splits, in turn, reduces the likelihood of moving records to a different cylinder, away from other records in key sequence.

The amount of free space to be provided depends on the number and location of records to be inserted, lengthened, or deleted. If the data set is for reference only, it will need no free space. If you do not provide enough free space, extra processing time is required to split control intervals or control areas and to process the records sequentially when they are not physically in sequence. If too much free space is provided, more direct-access storage than necessary is used to contain the data set and additional I/O operations are required to sequentially process the same number of records. In general, the percent and uniformity of growth should be estimated and a proportionate amount of free space should be left.

For example, if you estimate growth of 25%, distributed evenly across a data set, you might specify 20% of the total space as free space, because the data set is initially 80% of its eventual size. If you estimate growth of 25%, unevenly distributed across the data set, you might specify a small amount of free space when the data set is defined and increase the percent after the records are loaded into the data set. The amount of free space in existing control intervals and control areas remains the same. As new control intervals and control areas are required, they are created with the increased free-space specification; the chance of additional splits is reduced in the part of the data set with the most growth, but control intervals that are unchanged do not contain unneeded free space.

Random insertions in a key-sequenced data set tend to cause control-area splits throughout the data set at about the same time. Because each split results in two control areas for the original one control area, the amount of space used by the data set tends to double in a short time. You should provide enough free space within control intervals and control areas for a certain percentage of growth, rather than providing extra primary space beyond the end or providing for secondary space allocation. You can then monitor the growth of the data set, and as it approaches the percentage you decided on,

you can use the REPRO command to reorganize the data set to reestablish free space. Thus, you can avoid time-consuming control-area splits.

Index Options

Three options for the use of the index with a key-sequenced data set influence performance and storage requirements. The options are:

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

Index and Data on Separate Volumes

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

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

Sequence-Set Records Adjacent to Control Areas

When the data set is defined, you can specify that the sequence-set index record for each control area is to be on the first 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* (that is, as many copies of the sequence set as will fit on the track are written on the track).

On disk storage, the sequence-set index record may be placed adjacent to the control area to avoid moving the arm separately for index and for data; the imbedded sequence-set record is replicated to reduce rotational delay. Figure 9 illustrates replication of a sequence-set record that has been placed adjacent to its control area.

Replication of Index Records

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

Of course, this option costs direct-access storage space; it requires a full track of storage for each index record replicated. You have to weigh the relative values of direct-access storage space and processing speed.

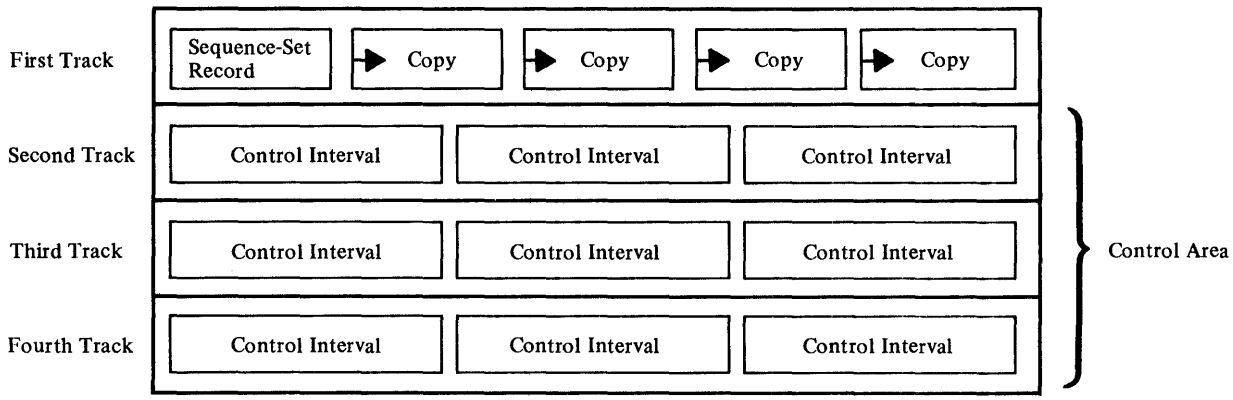


Figure 9. Index-Record Replication

I/O-Buffer Space

I/O-buffer space is important because VSAM transmits the contents of a control interval to a buffer in virtual storage; therefore, control-interval size is limited by the size of I/O buffers. For keyed access with the ACB operand STRNO=1, VSAM requires a minimum of three buffers, two for data control intervals and one for an index control interval. You may specify a minimum buffer space in the DEFINE command; if you do not specify a minimum buffer space, the default is enough buffer space for the minimum of three buffers.

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

The one-buffer minimum assumes that requests that require concurrent data-set positioning are not being issued. If such requests are issued, each requires exclusive control of an index I/O buffer. The value specified for the STRNO operand (ACB or GENCB macro or AMP parameter), therefore, is the minimum number of index I/O buffers required when requests that require concurrent positioning are used.

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

To improve performance when you have adequate real-storage available, you can increase the I/O-buffer space for index records in virtual storage by specifying I/O buffers for index records through the BUFNI and BUFSP operands of the ACB macro. With direct access, you should provide at least two index I/O buffers per request, to allow VSAM to keep the highest-level index record always resident. With sequential access, having only one index I/O buffer doesn't hinder performance, because VSAM uses the horizontal pointer in a sequence-set record, not vertical pointers in the index set, to get to the next control interval.

Performance Measurement

VSAM keeps statistical information about a data set in its catalog record. Some statistics, such as number of extents in a data set, 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 data set or altering the type of processing, to improve performance.

You can list the entire catalog record, the statistics and the parameters selected when the data set was defined, by using the LISTCAT command. You can use the SHOWCB and TESTCB macros in a processing program to display or test one or more data-set statistics. These statistics include:

- Control-interval size
- Percent of free control intervals per control area
- Number of bytes of available space (includes distributed free control intervals and allocated space beyond the last used control interval)
- Length and displacement of the key
- Maximum record length
- Number of levels in the index
- Number of extents
- Number of records retrieved, added, deleted, and updated
- 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 data set

Note: When a cluster or component is exported, that is, is named in an EXPORT command, the statistics are lost in the exported catalog record.

DATA SECURITY AND INTEGRITY

The protection of data includes data security, or the safety of data from theft or intentional destruction, and data integrity, or the safety of data from accidential loss or destruction. Security and integrity options are specified for an entry in the DEFINE command.

Data-Set Security

Access Method Services provides options to protect data sets against unauthorized use and loss of data. To effectively use the protection features, you must understand the difference between two different operations: 1) referring to a cataloged entry and 2) using the data set represented by a catalog entry. A catalog entry is referred to when new entries are defined (DEFINE), or old entries are altered (ALTER), deleted (DELETE), or listed (LISTCAT). The distinction between these two operations is the key to understanding how the VSAM passwords work; different passwords may be needed for the two operations.

The data-set security options are described in the sections that follow.

Passwords to Authorize Access

You can optionally define passwords for clusters, cluster components (data and index), page spaces, and VSAM catalogs, which a person must give to get permission to gain access to them. There are different passwords for various degrees of security. The higher levels provide greater protection than the lower levels. The levels are:

- *Full access.* This is the master password, which allows you to perform all operations (retrieving, updating, inserting, and deleting) on a data set and any index and catalog record associated with it. Using this password to gain access allows you to delete an entire data set and to alter any catalog information (including passwords) about a data set, index, or catalog, except the cataloged object's physical-extent descriptors. The master password allows all operations and bypasses any additional verification checking by the user-security verification routine.
- *Control access.* This password authorizes you to use control-interval access. See *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications* for information on the use of control-interval access.
- *Update access.* This password authorizes you to retrieve, update, insert, or delete records in a data set. Specifying a catalog's update password authorizes you to define data sets in it, to delete or alter nonVSAM entries, and to delete user catalogs.
- *Read access.* This is the read-only password, which allows you to examine data records and catalog records, but not to add, alter, or delete them, nor to see password information in a catalog record.

If you define passwords for any data sets in a catalog, you must also define passwords for the catalog in order for the data-set passwords to have effect.

Operations on a catalog may be authorized by the catalog's appropriate password or, in some cases, by the appropriate password of the data set

whose definition in the catalog is being operated on. For example, defining a data set in a password-protected catalog requires the catalog's update (or higher) password. Listing or deleting a definition requires the appropriate password of either the catalog or the data set. However, if the catalog, but not the data set, is protected, you need give no password to list the data set's catalog definition.

Because a user catalog defines itself, it may be password-protected without the master catalog being password-protected. To delete a user catalog, you must give its master password, whether the master catalog is password-protected or not.

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 giving all of the higher passwords the value of the highest password specified. Thus, if you specify only a read level password, that password will become the update, control, and master level password as well. If you specify a read password and a control password, the control password value will become the master level password as well. However, the update level password will not be affected.

One reason for password-protecting the components of a cluster is to prevent access to the index of a key-sequenced data set, since the only way to gain access to an index is to open it independently of the cluster. (See *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications* for a description of access to an index.)

Some Access Method Services operations may involve more than one password authorization. For example, importing a data set involves defining the data set and loading records into it. If the catalog into which the data set is being imported is password-protected, its update (or higher) password is required for the definition; if the data set 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 data set is represented in the VSAM catalog by two or more entries: a cluster entry and a data entry, or, if the data set is a key-sequenced data set, 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 have no relationship to each other. For example, if you password-protect a cluster but don't password-protect the cluster's data component, someone 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.

Catalogs are themselves VSAM data sets, and may have passwords. 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, the update or higher level password is required when defining a user catalog. This is 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.

The following protection considerations and precautions should be observed when using commands which refer to the VSAM catalog:

- To gain access to passwords in a catalog (for example to list or change passwords), you must specify the master level password of either the entry or the catalog. Similarly, a master level password must be specified with the DEFINE command if you want to model the entry's passwords.
- To delete a protected data set entry from a VSAM catalog requires the master level password of the entry or the master level password of the catalog containing the entry. However, if the entry describes a VSAM data space, the update level password of the catalog is sufficient. If the entry to be deleted is a nonVSAM data set, the update level password is required. Whenever a catalog entry is created (with a DEFINE command), the update or higher level password is required.
- If the password of the catalog is the read level, catalog entries with the read level passwords may be listed by specifying the read password of the entry or the catalog's read level password. However, entries without passwords may be listed without specifying the catalog's read level password.
- If the proper password is not specified with the Access Method Services command, a password prompt will occur. Unless you have specified the CODE parameter on either the DEFINE or ALTER command, the prompt will include the name of the data set; if you have specified CODE, the prompt will include the code name you specified. In some circumstances, due to the manner in which catalog entries are referenced, more than one prompt may occur. For example, when an ALTER or DELETE request is processed, the catalog must be referred to twice, once to locate the information, and once to perform the requested function. Again, incorrect password specification when listing catalog entries may cause numerous prompts. To avoid unnecessary prompts, specify the catalog's password, which allows access to all entries contained in that catalog. A catalog's master level password will allow you to refer to all catalog entries.
- Specification of a password where none is required is always ignored.

The following protection considerations and precautions should be observed when using commands which cause a data set access:

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

Operator Prompting Code

Computer operators and TSO-terminal users may be given the opportunity to supply a correct password if a processing program doesn't give the correct one when it tries to open a password-protected data set. When the data set is defined, you may specify a code to be used in lieu of the data-set name to prompt the operator or terminal user for a password. The prompting code keeps your data secure by not allowing the operator or terminal user to know both the name of the data set and its password.

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

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

Attempts to Supply a Password

When you define a data set, you can specify the number of times the computer operator or terminal user is allowed to try to give the correct password when a processing program is trying to open a data set. If the allowed number of attempts is exceeded and you are using the System Management Facilities, a record is written to the SMF data set to indicate a security violation.

Passwords for NonVSAM Data Sets

When you define a nonVSAM data set in a VSAM catalog, the data set cannot be protected with passwords in its catalog entry. But you can password-protect a nonVSAM data set when you create it, by specifying LABEL=(PASSWORD | NOPWREAD) in the DD statement that describes the data set (for more details, see *OS/VS2 JCL*) and issuing the PROTECT macro to assign a password to the data set (for more details, see *OS/VS Data Management Services Guide* and *OS/VS2 System Programming Library: Data Management.*

A nonVSAM entry (not the data set itself) in a VSAM catalog is password-protected by the catalog's passwords. When you issue the following Access Method Services commands to process a nonVSAM entry, you are requested to supply the appropriate password:

- ALTER—you supply the catalog's update password, if it exists.
- DEFINE—you supply the catalog's update password, if it exists.
- DELETE—you supply the catalog's update password, if it exists.
- LISTCAT—you supply the catalog's read password, if it exists.

User-Security-Verification Routine

In addition to password protection, VSAM allows you to protect data by specifying a program to verify a user's authorization. Specific requirements of the user-security-verification routine are described in the chapter "User-Written Exit Routines" in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.* To use this routine, simply specify the name of the authorization routine you have written in the AUTHORIZATION parameter of the DEFINE or ALTER command.

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

Protecting Shared Data

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

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

Data-set sharing is controlled by the interaction of:

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

If a data set cannot be shared for the type of processing you specify, your request to open the data set is denied.

When you issue the OPEN macro for an access-method control block, VSAM enqueues the names of the associated cluster components (entry-sequenced data set or key-sequenced data set and its index). If DISP=OLD is specified in a DD statement, only the dsname associated with that DD statement (the cluster name) is reserved for the job by the operating system. To have the names of the cluster components reserved as well (to avoid having one of them unavailable), you may include DD statements with DISP=OLD for the components of the cluster. This practice, if adopted by all users in an installation, ensures that all resources needed to open the data set will be reserved for the job before it is initiated.

Sharing a Data Set Among Systems

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

- The data set may be fully shared, but the user must assume full responsibility for read and write integrity; VSAM does nothing to ensure integrity.
- The data set may be fully shared, and buffers used for direct processing are read in again for each request. This option allows you to use the RESERVE and DEQ macros to maintain data integrity while sharing the data set.

Jobs of two or more OS/VS systems may gain access to the same data set regardless of the disposition specified in each job's JCL. To get exclusive control of the data set, a job in one system must issue a RESERVE macro.

With shared direct-access storage devices, integrity cannot be guaranteed when users are allowed to share a data set among systems for output processing.

Sharing a Data Set Among Jobs

Independent jobs may request the use of a data set at the same time. To share a data set, each job must specify a disposition of SHR in its DD statement for that data set. The type of sharing allowed depends on the share attribute that was given the data set when it was defined.

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

- The data set may be shared by any number of users for input processing *or* used by one user for output processing. Full integrity is maintained with this option.
- The data set may be used by any number of users for input processing *and* by one user for output processing. Write integrity is maintained with this option; however, you must assume responsibility for read integrity.
- The data set may be fully shared, but you must assume full responsibility for read and write integrity; VSAM does nothing to ensure integrity.
- The data set may be fully shared. Buffers used for direct processing are read in again for each request. This option allows you to use the ENQ and DEQ macros to maintain data integrity while sharing the data set.

Data Integrity

Data integrity is ensured by standard provisions of VSAM and can be enhanced by options available to you.

VSAM provides integrity by:

- Inserting records in a control interval held in virtual storage by shifting records in the control interval without input/output operations instead of, for example, chaining records together in an overflow area.
- Minimizing index handling and modification: record insertion, update, and deletion don't require that an index record be modified unless a control interval or a control area is split.
- Preformatting each new control area of a data set and updating the catalog to indicate the RBA of the end of the data set and, for a key-sequenced data set, the highest-keyed record in the data set. Preformatting prevents the loss of data that has been added to the data set.
- Keeping track of the physical end and, for a key-sequenced data set, the logical end of a data set. These addresses are updated when a processing program issues a temporary CLOSE and at the end of data-set processing, when the data set is fully closed. (You may use the VERIFY command of Access Method Services to investigate whether a data set and its index have been properly closed.)
- Optionally preformatting control areas when loading records into a data set; you may use either the REPRO command of Access Method Services or your own processing program.
- Optionally verifying each write operation when writing data to auxiliary storage.

You may additionally improve data integrity by:

- Using the JRNAD exit to journalize transactions against a data set.
- Processing records in the SMF data set that contain information about VSAM.
- Backing up data sets.
- Ensuring accessibility of secondary extents.

Preformatting Control Areas

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 the interest of performance (SPEED). Preformatting clears all previous information from the direct-access storage area and writes an end-of-file indicator:

- For an entry-sequenced or relative record data set, in every control interval in the control area.
- For a key-sequenced data set, in the first control interval in the control area following the preformatted control area. (The preformatted control area contains free control intervals.)

As records are loaded into a preformatted control 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 at that point.

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

Backing up Data

Sometimes, because of system or hardware failure, data is destroyed or made inaccessible. You can take steps that will enable you to recover your data if such a failure occurs. Two Access Method Services commands enable you to create a backup copy of your data:

- EXPORT copies a VSAM cluster and its catalog entries. The copy can be imported to another OS/VS1 or OS/VS2 system that includes VSAM or can be used as a backup.
- REPRO copies a VSAM cluster or a sequential data set (but not the catalog entries). The copy can be used as a backup.

Using EXPORT and IMPORT to transport or back up a key-sequenced data set or using REPRO to copy it reorganizes the data set—it rearranges data records physically in ascending key sequence (control-interval and control-area splits might have got them physically out of order) and balances free space quantities.

After you have replaced a damaged data set with its backup copy, you can bring the backup copy up to date with the original by rerunning the jobs that updated the original between the time it was backed up and the time it became inaccessible.

Backing up the data sets in a user catalog enables you to recover from damage to the catalog. You can import the backup copy of a data set whose entry was lost or redefine the entry and reload the backup copy. If the catalog was completely lost, you could redefine it, then import or redefine and reload all of the data sets that were defined in the catalog. (Backing up the catalog itself is discussed in the section "Protecting VSAM Catalogs.")

Exporting and Importing a Data Set

You can use the EXPORT command (with the TEMPORARY parameter) to copy a VSAM cluster and its catalog entries onto a movable volume (that is, a demountable direct-access volume or a magnetic-tape volume). You can then use the IMPORT command either to move the portable copy to another VS1 or VS2 system that includes VSAM or to replace the original cluster and its catalog entries. The portable copy itself is inaccessible for processing.

Making a Copy of a Data Set

You can use the REPRO command to copy a VSAM or sequential data set into another VSAM or sequential data set, newly defined (for protection) in another catalog. (The REPRO command does not copy the data set's cataloged information.) You can use the REPRO command with the REPLACE parameter to merge the backup copy into the original data set, or you can delete and redefine the original data set and use REPRO to reload the backup copy into it. Or, because the backup copy is itself accessible for processing, you can replace the original with it. (You have to make available

for processing the catalog in which you defined the backup, and you have to use the name of the backup.)

If you periodically process a data set sequentially, you can easily create a backup copy as a by-product of normal processing. You can use this backup copy like one made by way of REPRO.

Ensuring Accessibility of Secondary Extents

When a VSAM catalog is damaged so that you have to use a backup copy of the catalog (see the following section for information about backing up catalogs), any extents that were allocated to the catalog's data sets since the backup copy was made become inaccessible.

(In an entry-sequenced data set, the records in the new extents will have been added at the end. But in a key-sequenced data set, some of the records in the new extents could have been moved there because of a control-area split.)

To ensure that all of the records in a catalog's data set are accessible after you fall back to a backup copy of the catalog, you can:

- Eliminate the need for secondary extents in the data set by providing a sufficiently large primary space allocation.
- Reduce the number of secondary extents by providing for sufficiently large secondary space allocations. Whenever a secondary extent is allocated, you can make a new backup copy of the catalog. You can monitor the growth of a data set by way of the LISTCAT command or the SHOWCB macro. (The use of SHOWCB to display statistics is described in the *VSAM Programmer's Guide.*)

For growth that did not require secondary allocation in a data set since the backup copy of its catalog was made, you can use the VERIFY command to reset the end-of-file indicators in the catalog after you fall back to the backup copy.

Journaling Transactions for Possible Reconstruction

You may use the JRNAD exit to record information in a journal data set for each transaction against a VSAM data set. You could keep track of the data that is added to or deleted from the data set and of the users that gain access to it. The journal could be used to reconstruct the data set if you were forced to fall back to a backup copy of it.

Using SMF to Record Changes to Catalogs

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

- Record type 62, VSAM cluster or component opened, is written at the successful or unsuccessful opening of a VSAM data set, index, or cluster. It identifies the catalog in which the object is defined and the volumes on which the catalog and object are stored.
- Record type 63, VSAM cluster or component cataloged), is written after a data set or cluster is cataloged in a VSAM catalog or its catalog definition is altered. It gives the catalog record for a newly defined object and the old and new versions of an altered catalog record.
- Record type 64, VSAM component status upon closing or reaching end of space on a volume, is written when a VSAM data set, index, or cluster is

closed, when it becomes necessary to switch to another volume to continue processing, or when no more space is available on a volume. It indicates what condition occurred, identifies the volume(s) on which the object is stored, gives the extents of the object on the volume(s), and gives statistics about processing events that have occurred since the object was defined.

- Record type 67, VSAM entry deleted, is written when a data set or cluster is deleted. It identifies the VSAM catalog in which the object was defined and gives the catalog record.
- Record type 68, VSAM entry renamed, is written when a data set, index, or cluster is renamed. It identifies the VSAM catalog in which the object is defined and gives the old and new names.
- Record type 69, VSAM data space defined or deleted is written when a data space is defined or deleted. It identifies the catalog in which the data space is defined and the volume on which it is (or was) allocated. It also gives the number of data spaces on that volume and the amount of space available in them.

See the chapter "Records Written to the SMF Data Set" for detailed information about the contents of record types 63 and 67. See *OS/VS System Management Facilities (SMF)* for detailed information about the contents of record types 62, 64, 68, and 69.

Protecting VSAM Catalogs

Because you cannot gain access to a VSAM data set without the master or user catalog in which the data set is defined, you need to make sure that your catalogs can always be made available. You can take preventive measures to minimize the possibility of catalog loss and corrective measures to recover from loss, should it occur.

The effect of the loss of a catalog can be minimized by the following:

- Widespread use of user catalogs—a user catalog for each volume, where possible. (It is not possible for multivolume data sets—one catalog must control all the volumes of a data set.) Compare the effect of the loss of a catalog when (a) 10 data sets are cataloged in each of 10 catalogs and (b) 50 data sets are cataloged in each of 2 catalogs. The fewer the catalogs the greater the disruption in the event of loss of a catalog.
- Limited use of the master catalog.
- Dedicated use of volumes for VSAM data sets to segregate VSAM and nonVSAM recovery. You can dedicate a volume by defining a VSAM data space that occupies the whole volume.

Using a Backup Copy of a Catalog

You can use the REPRO command to unload (make a backup copy of) a catalog. If the catalog becomes inaccessible, you can use REPRO to reload the copy. The section "Backing up a Catalog" in the chapter "Copying and Printing" describes catalog unload/reload.

The backup copy might not reflect current information for some of the cataloged objects. The section "Updating a Backup Catalog" below describes how to make a backup catalog current.

Backing up the Master Catalog

An OS/VS2 system requires a master catalog—if a system or hardware failure damages the master catalog, the system cannot be used until the damage is corrected, unless you have a backup master catalog. Without a copy of the master catalog, you might have to generate a new system (using SYSGEN processing).

Because the system requires a master catalog, the reload procedure described in "Backing up a Catalog" in the chapter "Copying and Printing" requires modification for reloading the master catalog. Reloading the master catalog with REPRO works fine, *if you have a master catalog to reload.* When only some of the entries in the master catalog have become inaccessible, and the master catalog itself is still operational, you might be able to reload the backup into it. Otherwise, you can get a master catalog by:

- Using IEHDASDR (on another system) to restore the volume that contains the master catalog (from a tape onto which you have previously dumped the volume). You can do an IPL (initial program load) to bring your system up with the restored volume and then use REPRO to reload the backup into the restored version of the catalog. (If the backup is no more recent than the restored catalog, you can use the restored version without reloading.)

  With DUMP/RESTORE, you should design the volume that contains the master catalog to contain information that changes very little. Thus, you won't lose changes when you restore the volume. And, to avoid needing another system to restore the volume from a tape, you could have two direct-access volumes (with the same volume serial number) that contain a copy of the master catalog.

- Using DEFINE (on another VS2 system) to define a user catalog with the same name as your master catalog. You then use REPRO (still on the other system) to reload the user catalog with the backup copy of the master catalog. You can bring up your system with the reloaded user catalog as your master catalog.

Dumping a Catalog and Its Data Sets

This procedure calls for periodically dumping the volume(s) upon which a catalog and its data sets are stored with the IEHDASDR utility program (or with some other utility that achieves the same effect). Then if the catalog is lost or somehow becomes inaccessible, you can restore the volumes (or alternate volumes with the same volume serial numbers). The volumes would then contain what they contained when the backup copies were made. See "Updating a Backup Catalog" below about how to bring a restored catalog up to date.

If some of the space on a volume doesn't belong to a VSAM data space, the system may have allocated that space to nonVSAM data sets. With DUMP/RESTORE, you will have to fall back to a previous copy not only of the VSAM data sets and catalog, but also of the nonVSAM data sets.

Before a volume is dumped, the system asks the operator for the correct password of each password-protected object on the volume. For more information on how to use IEHDASDR to dump and restore volumes containing VSAM objects, see *OS/VS Utilities.*

To use IEHDASDR to dump a user catalog, include a JOBCAT or STEPCAT DD statement to describe the catalog.

Updating a Backup Catalog

A reloaded or restored backup catalog won't reflect any changes in the original catalog between the time you unloaded it to make the backup and the time it became inaccessible. Three types of changes can have occurred:

- Entries can have been deleted by way of DELETE or permanently exported by way of EXPORT PERMANENT. You can remove these entries from the backup catalog with DELETE NOSCRATCH, which is described below under "VSAM Catalog Cleanup."
- Entries can have been defined by way of DEFINE, imported by way of IMPORT, or defined (nonVSAM entries only) by the VS2 scheduler. You can use DEFINE NONVSAM to bring the backup catalog up to date on nonVSAM entries. But a VSAM object defined by an entry missing from the backup catalog is no longer accessible. To regain the use of the space on a volume or in a VSAM data space whose entry is missing from the backup catalog, you can use ALTER REMOVEVOLUMES (described below under "VSAM Volume Cleanup").

  To recover objects that are lost in this way, you will have to redefine them and rerun the jobs that loaded and updated them, or you will have to have backed them up as described above under "Backing up Data."
- Entries can have been updated to describe extensions. For an extension that used space already allocated to an object (that is, before the original catalog was unloaded to make the backup), you can use the VERIFY command to update the object's end-of-file indicator. But for an extension that used newly allocated space, you *cannot* update the backup catalog to recover the data in the extension. (See "Ensuring Accessibility of Secondary Extensions" above about how to protect yourself from this loss of data.)

VSAM Volume Cleanup

You normally take away the ownership of a volume from a VSAM user catalog by deleting all the objects and all the data spaces on the volume. But if a user catalog is inaccessible for some reason, or it no longer contains entries for the volume or its data spaces (as, for example, when you reload a backup catalog), you cannot use the DELETE command to take away ownership.

The ALTER command with the REMOVEVOLUMES parameter enables you to get rid of all the VSAM data spaces on a volume *without gaining access* to the catalog that owns the volume. ALTER REMOVEVOLUMES overwrites the data spaces with binary 0s and rewrites the VTOC to remove the data spaces' Format 1 DSCBs and to turn off the VSAM ownership bit in the Format 4 DSCB.

When the user catalog itself is on the volume, ALTER REMOVEVOLUMES overwrites it as well. Thus, you can use ALTER REMOVEVOLUMES to clean up a volume that contains a user catalog that has become inaccessible or to delete a user catalog without first deleting all of its entries.

To remove a single data space from a volume (where there are two or more data spaces), *do not use* ALTER REMOVEVOLUMES—use a utility program to scratch the data space's Format 1 DSCB.

ALTER REMOVEVOLUMES doesn't remove nonVSAM data sets from a volume. Nor does it remove VSAM objects from a volume owned by the master catalog.

Caution:

- ALTER REMOVEVOLUMES is also used simply to take away from a catalog ownership of a candidate volume that doesn't yet contain a VSAM data space. Be careful not to specify a volume that contains a VSAM catalog or VSAM data sets that you don't want to remove.
- You should not use ALTER REMOVEVOLUMES to get rid of VSAM objects on a volume owned by a catalog that you can still use—you should use the DELETE command. However, if you do use ALTER REMOVEVOLUMES to get rid of VSAM objects whose entries are accessible, you should use DELETE NOSCRATCH to remove the entries from the catalog. Otherwise, the catalog indicates that VSAM still owns space on the volume, and both VSAM and OS/VS2 might allocate it. (The next section describes how to remove entries from a VSAM catalog without gaining access to the volume that contains, or contained, the objects defined by the entries.)

VSAM Catalog Cleanup

When a volume becomes inaccessible, you can use the DELETE command with the NOSCRATCH parameter to remove from the catalog that owns the volume the entries for the VSAM objects on the volume. You can also use DELETE NOSCRATCH to remove from a backup catalog entries for the VSAM objects that were deleted from the original catalog between the time it was unloaded to make the backup and the time it became inaccessible. DELETE NOSCRATCH removes an entry *without gaining access* to the volume indicated in the entry. (DELETE *without* NOSCRATCH specified allows Access Method Services to gain access to a volume to change the VTOC or overwrite an object with binary 0s.)

VSAM cleans up the catalog:

- When CLUSTER or PAGESPACE is specified, by deleting the cluster or pagespace entry.
- When SPACE is specified, by deleting the volume entry. The volume entry must be empty (that is, you must first delete the entries of all VSAM objects that are indicated in the entry).

DEFINING ENTRIES

VSAM uses catalogs as a central information point for all VSAM data sets and the direct-access volumes on which they are stored. Before a data set—a VSAM cluster, nonVSAM data set, or a user catalog—can be defined, you must have a VSAM catalog in which to catalog the data set. A master catalog is established during the SYSGEN process.

You can define VSAM entries in the catalog only by using the DEFINE command. You can define entries for:

- User catalogs.
- Data spaces, which are spaces to be used for catalogs and VSAM data sets; these spaces are controlled by VSAM.
- Clusters, which are data components or data components and associated index components.
- NonVSAM data sets, which are data sets that are not in VSAM format but are cataloged in a VSAM catalog. NonVSAM data sets can be cataloged through either the DEFINE command, or JCL, or IEHPROGM.
- Generation data groups, which are collections of nonVSAM data sets that are grouped together in a time dependent manner.
- Alias names, which are alternate names for user catalogs or for nonVSAM data sets, excluding generation data groups.
- Page spaces, which are VS2 system data sets.

When you define an entry, you specify attributes to be associated with the entry. The attributes include, for example, any passwords required to use data and how space is to be allocated. After an entry is defined, you may load records into a data set by using the REPRO command. See the chapter "Copying and Printing Data" later in this publication for more information on loading records into a data set.

VSAM data sets are stored in data spaces. You may allow a data space to contain many data sets by simply not specifying otherwise (the usual case and the default), or define each component of a cluster as the only one in its data space. In the usual case, you define a data space first, then define the data sets. To define a component as the only one in its data space, specify the parameter UNIQUE when you define the data set. VSAM allocates a unique data space at the same time as it sets up the data set.

Preventing Duplicate Names in Catalogs

With multiple catalogs (the master catalog and optional user catalogs), you need to take some precautions to prevent duplicating names of data sets in two or more catalogs. In a VS1 system, a check is made for OS/VS CATLG requests and Access Method Services DEFINE requests to disallow duplicate names, but the system does not prevent duplication in these cases:

- A user catalog in a VS1 system is not available (that is, not included in JCL) for a check, and a name used in it is used to catalog or define a data set in another catalog.
- For Access Method Services ALTER requests, a check is made only for duplicate names within the catalog that contains the definition being altered.
- A name is entered in a master or user catalog that duplicates the higher level(s) of a qualified name in a VS1 system catalog (a check is made only for fully-qualified names). For example, A.B in a user catalog duplicates the first part of A.B.C.D in the system catalog. A subsequent attempt to locate data set A.B.C.D by means of the generic name A.B locates data set A.B instead if the user catalog is present to be searched.

Additionally, duplication is not prevented when a user catalog is imported into a system; no check is made to determine whether the imported catalog contains a name that another catalog already in the system contains.

JCL and Dynamic Allocation (DEFINE)

Defining an entry doesn't normally require that a volume be mounted, because the availablity of space in data spaces can be determined by examining the volume information in the catalog. A volume must be mounted whenever its VTOC has to be consulted or modified, that is, when a data space, a unique data set or page space, or a catalog is being defined.

A data set or volume can be dynamically allocated by the system, if no DD statement is supplied, by specifying the data set's dsname or the volume's serial number. The volume must be mounted as permanently RESIDENT or RESERVED.

If JCL is used to allocate a volume, include a DD statement of the form:

```
// ddname DD DISP=OLD,UNIT=( type [ , unitcount ]
//       [ ,DEFER ] ),VOLUME=SER=( serial1 [ , serial2 , ...] )
```

The disposition is always OLD and never NEW because VSAM allocates the space. NEW causes the operating system to allocate space that wouldn't be used by VSAM. SPACE information is not specified in the DD statement because you specify it directly in the DEFINE command. You can specify a device type for any direct-access storage device that is supported by your VS2 system. Volumes indicated in a single DD statement must be of the same type. The VOLUME parameter indicates the serial number or list of serial numbers of volume(s) on which data spaces are to be allocated.

VSAM defines a data space or a data set in the master catalog unless you specify a user catalog in a DD statement. The data-set name can be a qualified name—the first qualifier is the name or alias of a user catalog, or the alias of a control volume. You can specify a catalog for availability

throughout a job or for a single job step. The ddname JOBCAT is provided for job user catalog(s); STEPCAT is provided for job-step user catalog(s):

```
// { JOBCAT | STEPCAT } DD DISP= { OLD | SHR } ,
//          DSNAME= usercatalogname
```

Because the location of the user catalog is given in the master catalog, unit and volume information is not specified. If both JOBCAT and STEPCAT are specified, the user catalog identified by STEPCAT is used for the job step.

More than one user catalog can be indicated for a job or job step. To indicate another user catalog, follow the JOBCAT or STEPCAT DD statement with a concatenated DD statement—an unlabelled DD statement that specifies the name of another user catalog:

```
//JOBCAT DD DISP=SHR,DSN=MGMTCAT1
// DD DISP=SHR,DSN=MGMTCAT2
```

Order of Catalog Use: DEFINE

The order in which catalogs are used when an entry is defined is:

- If a catalog is specified in the CATALOG parameter, that catalog is used.
- The first user catalog specified in the current job step (STEPCAT) or, if none is specified for the job step, the first user catalog in the current job (JOBCAT).
- If no user catalog is specified for the current job step or job, and the entry's name is a qualified name, and the first qualifier (that is, the first one to eight characters before the period) is the same as:
  - the name of user catalog, or
  - the alias of a user catalog, or
  - the alias of a control volume catalog (CVOL),

  that user catalog or control volume (CVOL) is used.
- The master catalog is used.

Defining a Catalog

The DEFINE command can be used to define user catalogs.

A master catalog, which is the system catalog, is established at system-generation time. Without a master catalog you can't define user catalogs, data spaces, or data sets. The volume on which the master catalog is defined must be permanently mounted. The master catalog is selected for use at initial program load (IPL) time.

In a system with the Mass Storage System, a user catalog can be defined on a mass storage volume. If it is, it is staged and bound when it is opened (that is, it is retained until closed on the direct-access storage staging drive to which it is staged). A catalog can contain entries for both objects stored on mass storage volumes and objects stored on direct-access storage volumes.

User catalogs are pointed to by the master catalog. Each user catalog resides on a different volume. All VSAM clusters on a volume must be defined in the volume's user catalog. A user catalog can control (that is, catalog the objects of) more than one volume.

Because the master catalog is created at system-generation time and because only one master catalog is allowed on a system, you cannot issue DEFINE MASTERCATALOG to create a master catalog. If you issue DEFINE MASTERCATALOG, VSAM creates a user catalog.

The data space that contains the catalog is built when the catalog is defined. The data space is built for the catalog's exclusive use when the DATA and INDEX parameters do not include space subparameters. Otherwise, part of the data space can contain other VSAM data sets.

To share the data space with other VSAM data sets, the user specifies space for the data and, optionally, index components of the catalog. The space specified for the data and index components determine the amount of space used by the catalog; the space specified for the catalog as a whole determines the amount of space in the data space. If space is specified for the data component (using the DATA parameter) but not for the index component, VSAM determines the amount of space required for the catalog's index. The remainder of the data space is available for allocation to other data sets. The amount of space in the catalog's data space that is available for VSAM data sets is, in essence, the space that remains unallocated after the catalog's space is allocated to its components:

- If DATA space subparameters are not coded, the entire data space is available to contain catalog entries only. (INDEX space subparameters are invalid unless DATA space subparameters are also specified.)
- If DATA *and* INDEX space subparameters are coded, the sum of the space subparameters determines the amount of space available to contain the catalog. VSAM determines the proportion of this amount of space to be allocated to the catalog's data and index components. The remainder of the data space's space is available to contain suballocated VSAM data sets.
- If DATA *but not* INDEX space subparameters are coded, VSAM determines the amount of space required for the index and adds it to the DATA space subparameter specification. That is, VSAM determines the amount of space needed for the catalog's index based on the amount of

space specified for the catalog's data component. The remainder of the data space (that is, the space that is not allocated to either the catalog's data or index components) is available to contain suballocated VSAM data sets.

For example, if the user specified for the catalog as a whole 100 cylinders of a 3330 Disk Storage device (that is, for the catalog's data space) and for the catalog's data and index components 5 cylinders, 95 cylinders remain in the data space and are available for suballocated VSAM data sets.

Catalog space is allocated in tracks (even when you specify cylinders or records). The number of tracks is a multiple of an allocation unit. An allocation unit is the size of a control area, plus one track for the control area's replicated sequence-set record. The size of an allocation unit is:

- 3 tracks for an IBM 3330 or 3330-II Disk Storage or 2305 Fixed Head Storage (2 tracks for the control area, plus 1 track for the sequence-set record).
- 5 tracks for an IBM 2314 Direct Access Storage Facility or 3340 Disk Storage (4 tracks for the control area, plus 1 track for the sequence-set record).

When the amount of space you specify is not a multiple of the device's allocation unit, the amount of space is rounded downward to a multiple and the additional tracks aren't used.

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

For example, assume that one cylinder on a 3330 is specified for the entire catalog (1 cylinder = 19 tracks):

- The number of tracks is rounded down to 18 tracks, which is six allocation units of three tracks each (the nineteenth track is not used and not available unless the catalog's data space is suballocated).
- One allocation unit (three tracks) is allocated to the catalog's index set.
- Because ten percent of the remaining 15 tracks is less than three tracks, one allocation unit is allocated to the high-key range of the data component. The first track contains a replicated sequence-set record. The next two tracks contain a control area with 40 control intervals, which can hold a maximum of 400 records. (Records in the high-key range are 47 bytes long.)
- The remaining four allocation units (12 tracks) are allocated to the low-key range of the data component. Each allocation unit includes one track for the replicated sequence-set record and two tracks for a control area with 40 control intervals, which can hold 40 records. (Records in the low-key range are 505 bytes long.) The low-key range can hold a maximum of 160 records. From 12 to 15 of these records (13 for a 3330) are used to describe the catalog itself.

When the maximum number of records is reached in the low-key or high-key range, the catalog must be extended to hold additional records. When you

define the catalog, you can specify an amount of space for secondary allocation.

A user catalog may be password-protected without the master catalog's being password-protected. To delete a user catalog, you must give its master password, whether the master catalog is password-protected or not.

When you define a catalog, you can use a JCL DD statement to identify and mount the volume on which the catalog is to reside. Instead of using a JCL DD statement, you can rely on dynamic allocation to identify the catalog's volume. The volume must be mounted as permanently RESIDENT or RESERVED. If JCL is used, include a DD statement of the form:

// *ddname* **DD DISP=OLD,UNIT=(** *type* **[,DEFER]),**
// **VOLUME=SER=** *serial*

The volume on which the catalog and its data space are to be allocated must be mounted, because the data space must be described in a data set control block (DSCB) that is stored in the volume's VTOC (volume table of contents). VSAM determines whether the space is available on the volume and, if so, formats a DSCB for the data space.

Catalog Space Estimates

To calculate the approximate number of tracks required for a catalog, use the following worksheet:

Variable Quantities	Formulas	Estimates
Basic requirement = 10	10	10
A = number of key-sequenced data sets	3*A	
B = number of entry-sequenced data sets	2*B	
C = number of nonVSAM data sets	C	
D = number of generation data groups	D	
E = number of alias entries	E	
F = number of volumes owned by the catalog depending on device type:		
F_1 = number of 2305 volumes	$2*F_1$	
F_2 = number of 2314, 3330, and 3340 volumes	$4*F_2$	
F_3 = number of 3330-II volumes	$6*F_3$	
G = for each key-sequenced data set (KSDS) with more than two volumes: add "1" for each additional group of one to five volumes. For example: for each KSDS of 2 volumes, G = 0 for each KSDS of 7 volumes, G = 1 for each KSDS of 8 volumes, G = 2	G	
H = for each entry-sequenced data set with more than five volumes: add "1" for each additional group of from one to eight volumes.	H	
I = for each group of 4 data spaces on a volume, add "1."	I	
J = number of entry records (subtotal of the previous items.)	J	
L = number of records for the catalog's data component.	L = 1.4*J	
M = number of records in the catalog's index component, where: X = 2 for 3330, 3330-II, and 2305 X = 4 for 2314 and 3340 Y = 3 for 3330, 3330-II, and 2305 Y = 4 for 2314 and 3340	M = (L/X)+Y	
N = total catalog size, in records. Note: N = 200 if L+M is less than 200.	N = L+M	
T = **total number of tracks:** T = N/11 if the catalog is on a 2314 T = N/20 if the catalog is on a 3330, 2305, or 3330-II T = N/12 if the catalog is on a 3340 Note: Round T upwards to the next multiple of: 5 if the catalog is on a 2314 or 3340 3 if the catalog is on a 3330 or 2305	T T (final)	

DEFINE (Catalog)

The format of the DEFINE command when it is used to define a catalog is:

```
DEFINE    MASTERCATALOG | USERCATALOG
          (    NAME(entryname)
               FILE(dname)
               VOLUME(volser)
               {TRACKS(primary [ b secondary]) |
                  CYLINDERS(primary [ b secondary]) |
                  RECORDS(primary [ b secondary])}
               [BUFFERSPACE(size)]
               [MASTERPW(password)]
               [CONTROLPW(password)]
               [UPDATEPW(password)]
               [READPW(password)]
               [CODE(code)]
               [ATTEMPTS(number)]
               [AUTHORIZATION(entrypoint [ b string])]
               [OWNER(ownerid)]
               [TO(date) | FOR(days)]
               [WRITECHECK | NOWRITECHECK]
               [DESTAGEWAIT | NODESTAGEWAIT]
               [MODEL(entryname [/password]
                    [ b catname [/password][ b dname ]])]     )

          [DATA
          (    [TRACKS(primary [ b secondary]) |
                  CYLINDERS(primary [ b secondary]) |
                  RECORDS(primary [ b secondary])]
               [WRITECHECK | NOWRITECHECK]
               [DESTAGEWAIT | NODESTAGEWAIT]
               [BUFFERSPACE(size)]     )]

          [INDEX
          (    [TRACKS(primary) |
                  CYLINDERS(primary) |
                  RECORDS(primary)]
               [WRITECHECK | NOWRITECHECK]
               [DESTAGEWAIT | NODESTAGEWAIT]     )]

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

DEFINE can be abbreviated DEF. The parameters of this command are described in the following groups:

- Entry type, which describes the MASTERCATALOG, USERCATALOG, DATA, and INDEX parameters. These parameters indicate that a user catalog is to be defined. They also govern whether the attributes are separately specified for the data and index components.
- Name, which describes the NAME parameter.
- Allocation, which describes the FILE, VOLUME, TRACKS, CYLINDERS, RECORDS, and BUFFERSPACE parameters. These parameters are used to identify the volume on which the catalog is to

reside, the space to be set aside for the catalog, and the space to be used for buffers when the catalog is accessed.

- Protection and integrity, which describes the MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, and AUTHORIZATION, OWNER, TO, FOR, WRITECHECK, NOWRITECHECK, DESTAGEWAIT, and NODESTAGEWAIT. These parameters are used: to associate passwords with the catalog; to provide a mechanism by which the console operator can be prompted to supply a password without disclosing the name of the catalog; to identify a user-written routine for additional authorization verification; to identify the owner of the catalog; to specify the time for which the catalog is to be kept; to specify whether the write-check operation is to be performed; and to specify whether a user catalog that is stored on a mass storage volume is to be destaged synchronously or asynchronously with respect to the program that closes it.
- Model, which describes the MODEL parameter. This parameter is used to identify an existing user-catalog entry from which attributes are to be copied for the definition of a new user catalog. The MODEL parameter cannot be specified when a master catalog is to be defined.
- Catalog, which describes the CATALOG parameter. This parameter is used to supply the name and password, when required, of the master catalog.

Entry Type (Catalog)

The entry-type parameters govern the attributes to be associated with the entries created as a result of the DEFINE command.

Attributes can be specified for only the catalog as a whole or can be separately specified for the data and index components.

MASTERCATALOG(*options* **)**
specifies that a master catalog is to be defined. If MASTERCATALOG is specified, a user catalog is created. MASTERCATALOG is followed by the parameters specified for the catalog as a whole; these parameters are enclosed in parentheses and, optionally, are followed by the DATA and/or INDEX parameters and their subparameters, enclosed in parentheses. MASTERCATALOG can be abbreviated MRCAT or MCAT.

USERCATALOG(*options* **)**
specifies that a user catalog is to be defined. USERCATALOG is followed by the parameters specified for the catalog as a whole; these parameters are enclosed in parentheses and, optionally, are followed by the DATA and/or INDEX parameters and their subparameters, enclosed in parentheses. USERCATALOG can be abbreviated UCAT.

DATA(*options* **)**
specifies attributes of the data component of the catalog. The parameters that can be specified are: TRACKS, CYLINDERS, RECORDS, WRITECHECK, NOWRITECHECK, DESTAGEWAIT, NODESTAGEWAIT, and BUFFERSPACE. These parameters, their abbreviations, and any restrictions are described in the following section. If one of these values is not specified as a parameter of DATA, the data component shares the value or attribute specified for the catalog as a whole.

INDEX(*options* **)**

specifies attributes of the index component of the catalog. The parameters that can be specified are: TRACKS, CYLINDERS, RECORDS, WRITECHECK, NOWRITECHECK, DESTAGEWAIT, and NODESTAGEWAIT. These parameters, their abbreviations, and any restrictions are described in the following section. TRACKS, CYLINDERS, or RECORDS must have been specified as a parameter of DATA to be specified as a parameter of INDEX. If any of these values are not specified as a parameter of INDEX, the index component shares the value or the attribute specified for the catalog as a whole. INDEX can be abbreviated IX.

DATA and INDEX space subparameters together specify the total amount of space that is to be used by the catalog (VSAM determines the proportion of the total amount to allocate to each catalog component.) If you don't code the INDEX space parameter, VSAM determines the amount of space required for the index and adds it to the amount specified for the data component. The space specified for the catalog as a whole determines the size of the data space that contains the catalog. The amount of space available for VSAM data sets is, in essence, the space that remains unallocated after the catalog's space is allocated to its components:

- If DATA space subparameters are not coded, the entire data space is available to contain catalog entries only. (INDEX space subparameters are invalid unless DATA space subparameters are also specified.)
- If DATA *and* INDEX space subparameters are coded, the sum of the space subparameters determines the amount of space available to contain catalog entries. VSAM determines the proportion of this amount of space to be allocated to the catalog's data and index components. The remainder of the data space's space is available to contain suballocated VSAM data sets.
- If DATA *but not* INDEX space subparameters are coded, VSAM determines the amount of space required for the index and adds it to the DATA space subparameter specification. That is, VSAM determines the amount of space needed for the catalog's index based on the amount of space specified for the catalog's data component. The remainder of the data space (that is, the space that is not allocated to either the catalog's data or index components) is available to contain suballocated VSAM data sets.

Name (Catalog)

The name parameter is used to uniquely identify the catalog to be defined.

NAME(*entryname* **)**

specifies the name of the catalog being defined. The name may contain from 1 through 44 alphanumeric characters, national characters (@, #, and $), and two special characters (the hyphen and the 12-0 overpunch). Names containing more than eight characters must be segmented by periods; one to eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or national character.

Allocation (Catalog)

The allocation parameters are used to specify:

- The volume on which the catalog is to reside.
- The amount of space to be allocated.
- The space to be provided for buffers.

See "Catalog Space Estimates" earlier in this chapter for information about estimating the amount of space to be specified for a catalog.

FILE(*dname* **)**
specifies the name of the DD statement that identifies the device and volume to be used for the catalog. The DD statement should specify DISP=OLD to prevent premature space allocation on the volume. If FILE is not specified and the volume is physically mounted, the volume identified with the VOLUME parameter is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED.

VOLUME(*volser* **)**
specifies the volume that is to contain the catalog. A user catalog can be defined on a mass storage volume; the master catalog can't. The volume cannot be currently owned by any other VSAM catalog. VOLUME is required and must be specified as a parameter of USERCATALOG or MASTERCATALOG. A volume serial number, volser, may contain one to six alphameric, national (@, #, and $), and special characters; the special characters include commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs. A volume serial number must be enclosed in single quotation marks if it contains a special character. Single quotation marks within a volume serial number must be coded as two single quotation marks. VOLUME can be abbreviated VOL.

TRACKS(*primary* [ƀ *secondary*]) |
CYLINDERS(*primary* [ƀ *secondary*]) |
RECORDS(*primary* [ƀ *secondary*])
specifies the amount of space to be allocated in terms of tracks, cylinders, or number of records. One of these parameters must be specified as a parameter of USERCATALOG or MASTERCATALOG. TRACKS, CYLINDERS, and RECORDS can be abbreviated TRK, CYL, and REC, respectively.

primary [ƀ *secondary*]
specify the size of the primary and secondary extents to be allocated. Once the primary extent is filled, the space can expand to include a maximum of 13 additional secondary extents if you have specified a secondary allocation amount. These values can be expressed in decimal, hexadecimal, or binary. If a value is specified in hexadecimal or binary, it must be preceded by X or B, be enclosed in single quotation marks, and cannot be longer than one fullword. Secondary allocation should be specified in case the catalog has to be extended.

BUFFERSPACE(*size* **)**
specifies the minimum space, in bytes, to be provided for buffers when using the catalog. If BUFFERSPACE is not coded, 3072 is the default. If BUFFERSPACE is coded, the amount must be a multiple of 1024 and must be at least 3072 bytes; the maximum that can be specified is 8192. If a value greater than 8192 is specified, the BUFFERSPACE size defaults to 8192. BUFFERSPACE can be abbreviated BUFSPC or BUFSP.

size
is the amount of space, in bytes, to be provided for buffers. This value can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B, respectively, be enclosed in single quotation marks, and cannot be longer than one fullword.

Protection and Integrity (Catalog)

The protection and integrity parameters permit you to:

- Specify passwords to be associated with the catalog.
- Specify a prompting code and number of attempts allowed to provide the correct password in response to prompting.
- Identify a user's security-verification routine.
- Identify the owner of the catalog.
- Specify a retention period.
- Indicate whether write-check operations are to be performed for integrity.
- Specify whether a user catalog that is stored on a mass storage volume is to be destaged synchronously or asynchronously with respect to the program that closes it.

Note: A catalog must be master-password protected in order to password-protect any VSAM data sets cataloged in it. The catalog must be update-password (or higher-level password) protected in order to password-protect the catalog's nonVSAM data set entries or to prevent unauthorized users from cataloging data sets.

MASTERPW(*password* **)**
specifies a master level password for the catalog being defined. The AUTHORIZATION, CODE, and ATTEMPTS parameters have no effect unless the catalog has a master password associated with it. If MASTERPW is not specified, the highest level password specified becomes the password for all higher levels. The master password allows all operations; it is required to open the catalog as a data set. MASTERPW can be abbreviated MRPW.

CONTROLPW(*password* **)**
specifies a control level password for the catalog being defined. The control level password permits the same operations as the update level password.

UPDATEPW(*password*)
specifies an update level password for the catalog being defined. The update password permits entries to be added to the catalog being defined. The update level password permits you to delete nonVSAM entries and to disconnect user catalogs. UPDATEPW can be abbreviated UPDPW.

READPW(*password*)
specifies a read level password for the catalog being defined. The read level password permits you to list the catalog's entries (passwords and protection attributes are listed only when the master-level password is supplied). READPW can be abbreviated RDPW.

password
is a one-to-eight EBCDIC character password. If the password contains commas, semicolons, blanks, parentheses, or slashes, the password must be enclosed in single quotation marks. Single quotation marks within a password must be coded as two single quotation marks if the password is enclosed in single quotation marks. The password can be expressed in hexadecimal, where two hexadecimal characters represent an EBCDIC character. If the password is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks.

CODE(*code*)
specifies a code name for the catalog being defined. If an attempt is made to access a password-protected catalog without a password, the code name is used in a prompting message; the code enables the operator to be prompted for the password without disclosing the name of the catalog. If CODE is not specified, the operator is prompted with the name of the catalog. The code may contain one to eight EBCDIC characters. The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. The code can be specified in hexadecimal, where two hexadecimal characters represent an EBCDIC character. If code is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks.

ATTEMPTS(*number*)
specifies the maximum number of times the operator can try to enter a correct password in response to a prompting message. The number can be any number, 0 through 7. If 0 is specified, the operator is not prompted and is not allowed to enter a password from the console. If ATTEMPTS is not specified, 2 is the default. This value can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks. ATTEMPTS can be abbreviated ATT.

Note to TSO Users: At a TSO terminal the logon password is checked first before you are prompted to supply a password for the catalog. Checking the logon password counts as one attempt to obtain a password. If ATTEMPTS is not specified, you have one attempt to supply the catalog's password, because the default is 2.

AUTHORIZATION(*entrypoint* [b *string*]**)**
specifies that a user-security-verification routine (USVR) is available for additional security verification. When a protected catalog is accessed and the user supplies a correct password other than the catalog's master password, the USVR receives control. See "User-Written Exit Routines" in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* for information on the user-security-verification routine. AUTHORIZATION can be abbreviated AUTH.

entrypoint
specifies the name of the user's security-verfication routine. The name can contain one to eight alphanumeric, national (@, #, and $), or special (the hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or national character.

string
specifies information to be passed to the user-security-verification routine when it receives control to verify authorization. The string may contain 1 through 255 bytes of information in EBCDIC characters. The string must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. If a single quotation mark within a string must be coded as two single quotation marks if the string is enclosed in single quotation marks. If string is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks; it may contain up to 255 hexadecimal characters.

OWNER(*ownerid* **)**
specifies the identification of the owner of the catalog being defined. The ownerid may contain one to eight EBCDIC characters. The ownerid must be enclosed in single quotation marks if it contains a comma, semicolon, blank, parenthesis, or slash. If a single quotation mark appears within ownerid, it must be coded as two single quotation marks when the ownerid is enclosed in single quotation marks. Ownerid can be expressed in hexadecimal, where two hexadecimal characters represent an EBCDIC character. If ownerid is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks. If Access Method Services is being used interactively with TSO to define a catalog, and OWNER is not specified, the TSO user's userid is the default ownerid.

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the catalog being defined. If no value is coded, the catalog can be deleted whenever it is empty.

TO(*date* **)**
specifies the date, in the form yyddd, where yy is the year and ddd is the number (001 through 365) of the day, through which the cluster being defined is to be kept.

FOR(*days* **)**
specifies the number of days for which the catalog being defined is to be kept. The maximum number that can be specified is 9999. If the number specified is 0 through 1830, the catalog is retained for the number of days specified; if the number is between 1831 and 9999, the catalog is retained through the year 1999. This value can be expressed in decimal, hexadecimal, or binary. If the value is specified in hexadecimal or

binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks.

WRITECHECK | NOWRITECHECK
specifies whether the catalog is to be checked by a machine action called *write-check* when a record is written into it. If WRITECHECK is specified, a record is written and then read, without data transfer, to test for the data check condition. If no value is coded, NOWRITECHECK is the default. WRITECHECK and NOWRITECHECK can be abbreviated WCK and NWCK, respectively.

DESTAGEWAIT | NODESTAGEWAIT
specifies whether a user catalog that is stored on a mass storage volume is to be destaged synchronously or asynchronously with respect to the program that closes it.

DESTAGEWAIT
indicates that destaging is to be completed before control is returned from VSAM to the program that issues the CLOSE macro. VSAM can thus notify the program whether destaging was successful. DESTAGEWAIT can be abbreviated DSTGW.

NODESTAGEWAIT
indicates that notification of unsuccessful destaging is to be made only by a message to the operator and to the messages (SYSPRINT) data set. NODESTAGEWAIT can be abbreviated NDSTGW.

These attributes cannot be specified for a master catalog. If the user catalog isn't stored on a mass storage volume, the attribute is ineffective until the direct-access storage volume the user catalog is stored on is converted to a mass storage volume (by way of the CONVERTV command, which is described in *OS/VS Mass Storage System (MSS) Services for Space Management.*)

If one of these parameters is specified for the data component of a user catalog and the other parameter is specified for the index component, the parameter specified for the index component applies to both components, because the sequence set of the index of a catalog is imbedded in the data.

Model (Catalog)

It is possible to use an already defined master or user catalog as a model for another user catalog. When one entry is used as a model for another, its attributes are copied as the new entry is defined. You may use some attributes of the model and override others by explicitly specifying them in the definition of the user catalog.

If a model is used, you must specify certain parameters even though no attributes are to be changed or added. The name of the user catalog to be defined and volume and space information always must be specified. The volume and space information must be specified as parameters of USERCATALOG.

If the FILE parameter is not specified, the catalog's volume is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED.

MODEL
specifies that an existing master or user catalog is to be used as a model for the user catalog being defined. MODEL cannot be specified if a master catalog is being defined.

entryname
specifies the name of the master or user catalog to be used as a model.

password
specifies a password. If the catalog to be used as a model is password protected, a password is required. If the protection attributes are to be copied, substitute the master password of the catalog being used as a model. If passwords are not to be copied, any password can be used.

catname
specifies the name of the catalog to be used as a model. This parameter is required if (1) you are going to specify the password of the catalog that contains the entry instead of specifying the password of the entry itself, or (2) the catalog is not the master catalog or a catalog identified by a JOBCAT or STEPCAT DD statement.

dname
specifies the name of a DD statement that identifies the catalog that contains the entry to be used as a model. If this parameter is not coded and the catalog's volume is physically mounted, the catalog is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED.

If you use the MODEL parameter and explicitly (1) specify a different device type through the VOLUME parameter, (2) change the buffer space through the BUFFERSPACE parameter, or (3) change the unit of allocation through the TRACKS, CYLINDERS, or RECORDS parameters, your job may be terminated because of allocation problems.

Catalog (Catalog)

The catalog parameter is used to supply the name and password, when required, of the master catalog when a user catalog is to be defined.

CATALOG(*mastercatname* [*/ password*] **)**
specifies the name and password of the master catalog. If the master catalog is password protected, the password must be provided in this parameter or in response to prompting. The password must be the update or higher level password. This parameter is applicable only when a user catalog is being defined. CATALOG can be abbreviated CAT.

DEFINE CATALOG Examples

Define a User Catalog: Example 1

In this example, a user catalog is defined.

```
//DEFCAT1  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//VOL2     DD     VOL=SER=VSER02,UNIT=2314,DISP=OLD
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DEFINE USERCATALOG( -
               NAME(D27UCAT1) -
               MASTERPW(MRPWD27) -
               UPDATEPW(UPPWD27) -
               FOR(365) -
               CYLINDERS(150 5) -
               FILE(VOL2) -
               VOLUME(VSER02)  ) -
          DATA( -
               CYLINDERS(8 5)  ) -
          INDEX( -
               CYLINDERS(5)  ) -
          CATALOG(AMASTCAT/MRCATPW2)
/*
```

The job control statements are:

- VOL2 DD, which describes the volume on which the catalog is to be defined.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DEFINE USERCATALOG command defines a user catalog, D27UCAT1. Its parameters are:

- NAME, which names the user catalog, D27UCAT1.
- MASTERPW and UPDATEPW, which specify master and update passwords for the catalog: MRPWD27 and UPPWD27.
- FOR, which specifies that the user catalog is to be retained for 365 days.
- FILE, which points to the VOL2 DD statement. The VOL2 DD statement allocates the volume on which the user catalog is to reside.
- CYLINDERS, which specifies that 150 cylinders are to be allocated for the user catalog's data space. When the user catalog's data space is extended, it is to be extended in increments of 5 cylinders.
- VOLUME, which specifies that the user catalog is to reside on volume VSER02.

- DATA and INDEX, which specify that VSAM is to allocate 13 cylinders for the user catalog. (VSAM determines the proportion of space that is allocated to the catalog's data and index components.) The remainder of the catalog's data space (137 cylinders) is available to contain suballocated VSAM clusters. If the catalog's data component is extended, it is to be extended in increments of five cylinders. The catalog's index component cannot be extended.
- CATALOG, which specifies that the user catalog connector is to be defined in the AMASTCAT catalog. The update password of AMASTCAT is MRCATPW2.

Define a User Catalog Using the MODEL Parameter: Example 2

In this example, the user catalog defined previously, D27UCAT1, is used as a model for the user catalog being defined, D27UCAT2.

```
//DEFCAT2  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//STEPCAT  DD      DSNAME=D27UCAT1,DISP=SHR
//VOL3     DD      VOL=SER=VSER03,UNIT=2314,DISP=OLD
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE USERCATALOG( -
               NAME(D27UCAT2) -
               MASTERPW(MRPWD27) -
               UPDATEPW(UPPWD27) -
               CYLINDERS(150 5) -
               VOLUME(VSER03) -
               FILE(VOL3) -
               MODEL(D27UCAT1/MRPWD27 D27UCAT1/MRPWD27)   )
          CATALOG(AMASTCAT/MRCATPW2)
/*
```

The job control statements are:

- STEPCAT DD, which makes a catalog available for this job step: D27UCAT1.
- VOL3 DD, which describes the volume on which the catalog is to be defined.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DEFINE USERCATALOG command defines user catalog D27UCAT2. Its parameters are:

- NAME, which names the user catalog, D27UCAT2.
- MASTERPW and UPDATEPW, which specify master and update passwords for the catalog: MRPWD27 and UPPWD27.
- CYLINDERS, which specifies that 150 cylinders are to be allocated for the user catalog's data space. When the user catalog's data space is extended, it is to be extended in increments of 5 cylinders.
- FILE, which points to the VOL3 DD statement. The VOL3 DD statement has allocated a unit for the volume on which the user catalog is to reside.

- VOLUME, which specifies that the user catalog is to reside on volume VSER03.
- MODEL, which identifies D27UCAT1 as the catalog to use as a model for D27UCAT2. The attributes and specifications of D27UCAT1 that aren't otherwise specified with the above parameters are used to define the attributes and specifications of D27UCAT2. Remember that D27UCAT1 contains the catalog entries that describe itself. This is the reason it is specified in MODEL's catname subparameter.
- CATALOG, which specifies that the user catalog connector is to be defined in the AMASTCAT catalog. The update password of AMASTCAT is MRCATPW2.

Define a User Catalog—Determining the Catalog's Space Requirements: Example 3

In this example, a user catalog is defined. The user catalog's data space is not available to contain VSAM available clusters. The catalog is to be large enough to contain information about:

- 100 key-sequenced clusters (that is, indexed VSAM data sets)
- 10 entry-sequenced clusters (that is, sequential nonindexed VSAM data sets)
- 5 volumes (the number to be controlled by the catalog). Each volume is a 3330 volume and contains 20 data spaces.

To determine how much space, in records or tracks, is required for the primary allocation of a to-be-defined catalog in your system, see the previous section, "Catalog Space Estimates." The number of records for the primary allocation of the catalog in this example is shown in the following filled-in worksheet:

Variable Quantities	Formulas	Estimates
Basic requirement = 10	10	10
A = number of key-sequenced data sets	3*A	300
B = number of entry-sequenced data sets	2*B	20
C = number of nonVSAM data sets	C	0
D = number of generation data groups	D	0
E = number of alias entries	E	0
F = number of volumes owned by the catalog depending on device type:		
F_1 = number of 2305 volumes	$2*F_1$	0
F_2 = number of 2314, 3330, and 3340 volumes	$4*F_2$	20
F_3 = number of 3330-II volumes	$6*F_3$	0
G = for each key-sequenced data set with more than two volumes: add "1" for each additional group of one to five volumes. For example: for each KSDS of 2 volumes, G = 0 for each KSDS of 7 volumes, G = 1 for each KSDS of 8 volumes, G = 2	G	0
H = for each entry-sequenced data set with more than five volumes: add "1" for each additional group of from one to eight volumes.	H	0
I = for each group of 4 data spaces on a volume, add "1."	I	25
J = number of entry records (subtotal of the previous items.)	J	375
L = number of records for the catalog's data component.	L = 1.4*J	525
M = number of records in the catalog's index component, where: X = 2 for 3330, 3330-II, and 2305 X = 4 for 2314 and 3340 Y = 3 for 3330, 3330-II, and 2305 Y = 5 for 2314 and 3340	M = (L/X)+Y	266
N = total catalog size, in records	N = L+M	791

The number of records required for the catalog's primary extent is at least 791. Rounded upward the example specified 800 records.

If VSAM extends the catalog, it should extend it in increments of 100 records.

```
//DEFCAT3  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//VOL      DD      VOL=SER=VSER04,UNIT=3330,DISP=OLD
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE USERCATALOG( -
          NAME(MYCAT) -
          FILE(VOL) -
          VOLUME(VSER04) -
          RECORDS(800 100) )
/*
```

The job control statements are:

- VOL DD, which describes the volume on which the catalog is to be defined.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DEFINE USERCATALOG command defines a user catalog. Its parameters are:

- NAME, which specifies the name of the catalog, MYCAT.
- FILE, which identifies the DD statement that describes the volume on which the catalog is to be defined.
- VOLUME, which specifies that the catalog is to reside on the volume whose serial number is VSER04.
- RECORDS, which specifies that the catalog's data space is to be 800 records. When the user catalog is extended, VSAM extends it in increments of 100 records.

Defining a Data Space

The DEFINE command can be used to define VSAM data spaces or to reserve volumes for future VSAM use. A VSAM data space is space on a direct access volume that is owned and managed by VSAM. All of the data spaces (and VSAM data sets) on a volume must be defined in the same VSAM catalog. When you define the volume's first data space (or you specify the volume as a candidate to contain VSAM objects) you are, in effect, giving control over the volume to the catalog that contains the data space entry. All future data spaces and VSAM objects on that volume must be defined in the defining catalog (that is, the catalog that describes the volume's first-defined data space).

In a system with the Mass Storage System, a data space can be defined on a mass storage volume.

A VSAM data space may include several extents on the same volume, but it cannot span volumes. The volume that contains the data space is owned by the catalog that contains the catalog entry for the space.

All VSAM data sets are stored in VSAM data spaces. A *unique* component is defined as the only component in its data space. VSAM allocates the data space for a unique component when you define the data set. To define a data set that can share a data space with other data sets, a data space must have been defined beforehand in which to allocate space for the data set.

A data space can contain any number of nonunique data sets, and a data set can be stored in more than one data space, on the same volume or on different volumes.

A VSAM data space can take up all or part of the space of a direct-access volume. It cannot take up space on more than one volume. There can be more than one data space on a volume.

When VSAM allocates space for a new data set or extends the space for an old data set, it allocates the space from that available in a data space. For a new data set, if not enough space is available, the DEFINE fails, and you must define a data space with sufficient space. For an old data set, if not enough space is available, VSAM tries to extend the data space. If the data space can't be extended (it may already have the maximum number of extents or there may not be enough space on the volume), VSAM tries to set up a new data space on the same volume as the old. If there isn't enough space on the volume and you identified another volume for expansion when you defined the data set, VSAM tries to set up a data space on that volume.

When you define a data space in a catalog, you must specify a volume for the data space in the DEFINE command. Optionally, you can specify more than one volume. A data space of the size you indicate is allocated on each volume, and each volume comes under the ownership of the indicated catalog. Because a single DEFINE command can cause more than one data space to be created, an entry is created for each volume on which a data space resides. A volume entry describes each volume on which one or more data spaces have been defined.

When you define a data space, you may use JCL to cause the volumes on which the data space is to be allocated to be mounted, or you may rely on dynamic allocation. The volume must be mounted as permanently

RESIDENT or RESERVED. If the volume is owned by a catalog other than the one you specify, the DEFINE is not allowed. If there is space available and there is no volume ownership conflict in the catalogs, VSAM ownership is indicated in the VTOC and a VSAM data space is allocated on the volume. An ownership indicator is set in a Format 4 DSCB in the volume's table of contents (VTOC). A Format 1 (Identifier) DSCB describing the data space is written into the volume's table of contents (VTOC).

If JCL is used, include a DD statement of the form:

// ddname **DD DISP=OLD,UNIT=(** *type* **[,** *unitcount* **][,DEFER]),**
//VOLUME=SER=(*serial1* **,** *serial2* **, ...])**

DEFINE (Space)

The format of the DEFINE command when it is used to define a data space is:

DEFINE	**SPACE** **(VOLUMES(** *volser* **[b** *volser* **...])** **FILE(** *dname* **)** **{TRACKS(** *primary* **[b** *secondary* **]) |** **CYLINDERS(** *primary* **[b** *secondary* **]) |** **RECORDS(** *primary* **[b** *secondary* **])** **b RECORDSIZE(** *average* **b** *maximum* **) |** **CANDIDATE})** **[CATALOG(** *catname* **[/** *password* **][b** *dname* **])]**

where:

SPACE
specifies that a data space is to be defined. DEFINE can be abbreviated DEF. SPACE can be abbreviated SPC.

VOLUMES(*volser* **[b** *volser...***])**
specifies the volumes on which data spaces are to be defined. A data space can be defined on a mass storage volume. If two or more volumes are specified and an amount of space is specified (that is, TRACKS, CYLINDERS, or RECORDS, b RECORDSIZE is specified) the amount specified for the primary allocation is allocated on each volume. The specified volumes must all be of the same device type. A volume serial number, volser, may contain one to six alphameric, national (@, #, and $), and, except under TSO, special characters; the special characters include commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs. A volume serial number must be enclosed in single quotation marks if it contains a special character. Single quotation marks within a volume serial number must be coded as two single quotation marks. VOLUMES can be abbreviated VOL.

FILE(*dname* **)**
specifies the name of the DD statement that identifies the devices and volumes to be used for space allocation. If FILE is not specified and the volume is physically mounted, the volume identified with the VOLUME parameter is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED.

TRACKS(*primary* [b *secondary*]) |
CYLINDERS(*primary* [b *secondary*]) |
RECORDS(*primary* [b *secondary*])
specifies the amount of space to be allocated in terms of tracks, cylinders, or number of records. If RECORDS is specified, the space required is calculated in terms of the number of records, but the space is allocated by tracks. If RECORDS is specified, RECORDSIZE must also be specified. TRACKS, CYLINDERS, RECORDS, or CANDIDATE must be specified. TRACKS, CYLINDERS, and RECORDS can be abbreviated TRK, CYL, and REC, respectively.

primary [b *secondary*]
specify the size of the primary and secondary extents to be allocated. Once the primary extent is filled, the space can expand to include a maximum of 15 secondary extents if you have specified a secondary-extent amount. These values can be expressed in decimal, hexadecimal, or binary. If a value is specified in hexadecimal or binary, it must be preceded by X or B, be enclosed in single quotation marks, and cannot be longer than one fullword.

RECORDSIZE(*average* b*maximum* **)**
specifies the average and maximum lengths, in bytes, of the records. These values can be expressed in decimal, hexadecimal, or binary. If a value is specified in hexadecimal or binary, it must be preceded by X or B, respectively, be enclosed in single quotation marks, and cannot be longer than one fullword. The minimum that can be specified is one; the maximum is 32,761. RECORDSIZE can be abbreviated RECSZ.

CANDIDATE
specifies that the volumes listed in the VOLUMES parameter are reserved for future use by VSAM. The volumes are reserved for use by the catalog specified in the CATALOG parameter, but no space is allocated. TRACKS, CYLINDERS, RECORDS, or CANDIDATE must be specified. CANDIDATE can be abbreviated CAN.

CATALOG(*catname* [/ *password*][b *dname*])
identifies the catalog in which the data space is to be defined. CATALOG can be abbreviated CAT.

catname
specifies the name of the catalog in which the data space is to be defined.

password
specifies a password. If the catalog is password protected, you must supply the update or higher level password.

dname
specifies the name of a DD statement that identifies the catalog in which the data space is to be defined. If dname is not specified and the catalog's volume is physically mounted, the catalog's volume is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED. See "Order of Catalog Use: DEFINE" for information about the order in which catalogs are used.

DEFINE SPACE Example

Define a Data Space: Example 1

In this example, a data space is defined and will be used to allocate space for VSAM clusters that are subsequently defined.

```
//DEFSPC   JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//DATA     DD     VOL=SER=VSER05,UNIT=3330,DISP=OLD
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DEFINE SPACE -
          (CYLINDERS(100 10) -
          FILE(DATA) -
          VOLUMES(VSER05))
/*
```

VSAM assigns a name to the data space and keeps track of it for future cluster space-allocation. Because no user catalog is specified in this example, the master catalog is assumed as the default. (It is the first catalog found because no JOBCAT or STEPCAT DD statements were specified. See the previous section "Order of Catalog Use: DEFINE" for more details.)

The job control statements are:

- DATA DD, which describes the volume on which the data space is to be defined.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DEFINE SPACE command defines a VSAM data space. Its parameters are:

- CYLINDERS, which specifies that 100 cylinders are to be allocated for the data space. When the data space is extended, it is to be extended in increments of ten cylinders.
- FILE, which identifies the DATA DD statement. The DATA DD statement specifies the volume on which the data space is to be defined.
- VOLUMES, which specifies the volume serial number of the volume on which the data space is to be defined: VSER05.

  Access Method Services defines the data space and allocates its space on volume VSER05. The data space is cataloged in the master catalog, because the DEFINE command doesn't include a CATALOG parameter, and because no JOBCAT or STEPCAT DD statements are specified. All future data spaces and clusters on volume VSER05 must also be cataloged in the master catalog. NonVSAM data sets on the volume reside in areas that are not allocated to a VSAM data space.

To reserve a 3330 volume for VSAM's exclusive use, define the data space on it as 403 cylinders (this allows one cylinder to be used for the volume's table of contents—VTOC—and the volume's label) or define other data spaces on the volume with additional DEFINE SPACE commands (which can be given in the same job step) so that all the volume's space is part of a VSAM data space.

Defining a Cluster

The DEFINE command can be used to define a catalog entry and allocate space for an indexed or nonindexed cluster. When a DEFINE command is used to define a cluster, as many as three entries are created in the catalog: an entry for the cluster, its data component, and, if the cluster is indexed, its index component.

Attributes of components can be specified separately from the attributes of the cluster. If attributes are specified for the cluster as a whole and are not specified for the components, the attributes of the cluster (except for its passwords and other protection attributes) apply to its components. If an attribute that is applicable to the data or index component is specified for both the cluster and the component, the explicit component specification overrides that specified for the cluster.

You specify a name for the cluster when you define it and generally give this name as the dsname in JCL. You can optionally name the components of a cluster and work with them individually. For instance, you may open the index of a key-sequenced data set and process it as data. This processing is described in *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications.*

Specifying Cluster Information

You specify all of the necessary descriptive information and the performance, security, and integrity options when you create a cluster. Some of the information you specify can apply to either the data or the index or to both. You can specify information for the cluster as a whole in the CLUSTER parameter. Information for data only or for index only goes with the corresponding parameter (DATA or INDEX).

Descriptive Information

Descriptive information includes:

- Average and maximum lengths of data records
- Length and position of the key field in the records of a key-sequenced data set
- Identity of the catalog in which to define the cluster
- Identity of the volume(s) on which to allocate space for the cluster
- Amount of space to allocate for the cluster and whether the cluster is to reside in a separate data space
- Least amount of I/O-buffer space a processing program will provide to process the data set

With a multivolume key-sequenced data set, you may assign data to the various volumes according to ranges of key values. For example: if you have three volumes you might assign records with keys A-E to the first volume, F-M to the second, and N-Z to the third. Key-range allocation facilitates processing the data set with only one of the volumes mounted: you know which volume contains what records.

If a component is unique, that is, is to reside alone in a separate data space, the data space is not defined before the cluster is defined. The data space is created as the cluster is defined; in this case, the volume(s) must be mounted.

If you don't specify the space to be provided for buffers, VSAM determines control-interval size first and then sets buffer-space amount equal to the size of two data control intervals plus, for a key-sequenced data set, one index control interval. If you do specify it, data and index control intervals are limited to sizes such that the buffer space can hold two data control intervals and one index control interval.

If the values you specify for record length and key length require control intervals too large for the buffer space you specify, the DEFINE command will fail.

The relationship between control-interval size and least amount of I/O-buffer space is further discussed in "Control-Interval Size," in the chapter "Optimizing VSAM's Performance." If you specify track allocation or specify record allocation for a very small number of records, space is actually allocated by tracks and, for data integrity, a whole number of control intervals is required to fit on a track—that is, control intervals are not allowed to span tracks. If you cause space to be allocated in tracks in a case where control-interval size forces control intervals to span tracks, the DEFINE command will fail.

Performance-Options Information

Information for performance options (mostly for an indexed cluster) include:

- Whether to replicate index records
- Whether to place the sequence set of an index adjacent to data
- Whether to place an index on a separate volume from data
- Percentages of free space to be distributed throughout a data set
- Control-interval size for VSAM to use instead of calculating the size itself

These options are discussed in the chapter "Optimizing VSAM's Performance."

In a system with the Mass Storage System, you can also indicate how a cluster or component that is stored on a mass storage volume is to be staged.

Protection and Integrity Information

Information for protection and integrity options include:

- Passwords and related information
- Identity of your own authorization routine to verify that a requester has the right to gain access to data
- Whether to preformat control areas during loading
- Whether to verify that write operations are without error
- Whether and to what extent to share data among systems, jobs, and subtasks
- Whether to erase the information that a data set contains when you delete the data set

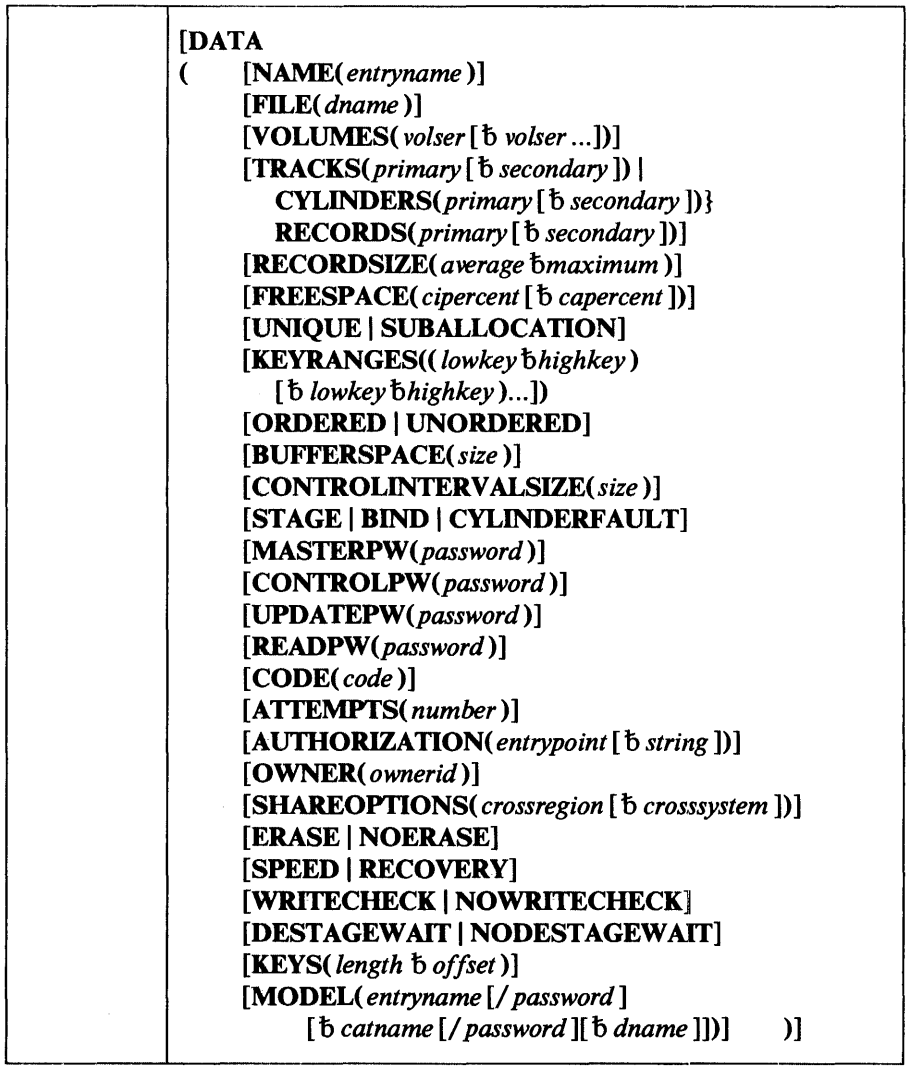
These options are discussed in the chapter "Data Security and Integrity."

In a system with the Mass Storage System, you can also indicate whether a cluster or component that is stored on a mass storage volume is to be destaged synchronously or asynchronously with respect to the program that closes it.

DEFINE (Cluster)

The format of the DEFINE command when it is used to define a cluster is:

DEFINE	**CLUSTER** **(** **NAME(***entryname***)** **[FILE(***dname***)]** **[VOLUMES(***volser* [b *volser* ...])] {**TRACKS(***primary* [b *secondary*]) | **CYLINDERS(***primary* [b *secondary*]) | **RECORDS(***primary* [b *secondary*])} **[RECORDSIZE(***average* b*maximum***)]** **[FREESPACE(***cipercent* [b *capercent*])]** **[UNIQUE | SUBALLOCATION]** **[KEYRANGES((***lowkey* b*highkey***)** [b (*lowkey* b*highkey*)...]) **[ORDERED | UNORDERED]** **[BUFFERSPACE(***size***)]** **[CONTROLINTERVALSIZE(***size***)]** **[STAGE | BIND | CYLINDERFAULT]** **[MASTERPW(***password***)]** **[CONTROLPW(***password***)]** **[UPDATEPW(***password***)]** **[READPW(***password***)]** **[CODE(***code***)]** **[ATTEMPTS(***number***)]** **[AUTHORIZATION(***entrypoint* [b *string*])]** **[OWNER(***ownerid***)]** **[TO(***date***) | FOR(***days***)]** **[SHAREOPTIONS(***crossregion* [b *crosssystem*])]** **[ERASE | NOERASE]** **[SPEED | RECOVERY]** **[WRITECHECK | NOWRITECHECK]** **[DESTAGEWAIT | NODESTAGEWAIT]** **[INDEXED | NOINDEXED]** **[KEYS(***length* b *offset***)]** **[REPLICATE | NOREPLICATE]** **[MODEL(***entryname* [/ *password*] [b *catname* [/ *password*][b *dname*]])]** **[IMBED | NOIMBED]** **)**


[DATA
([NAME(entryname)]
[FILE(dname)]
[VOLUMES(volser [volser ...])]
[TRACKS(primary [secondary]) |
CYLINDERS(primary [secondary])}
RECORDS(primary [secondary])]
[RECORDSIZE(average maximum)]
[FREESPACE(cipercent [capercent])]
[UNIQUE | SUBALLOCATION]
[KEYRANGES((lowkey highkey)
[lowkey highkey)...])
[ORDERED | UNORDERED]
[BUFFERSPACE(size)]
[CONTROLINTERVALSIZE(size)]
[STAGE | BIND | CYLINDERFAULT]
[MASTERPW(password)]
[CONTROLPW(password)]
[UPDATEPW(password)]
[READPW(password)]
[CODE(code)]
[ATTEMPTS(number)]
[AUTHORIZATION(entrypoint [string])]
[OWNER(ownerid)]
[SHAREOPTIONS(crossregion [crosssystem])]
[ERASE | NOERASE]
[SPEED | RECOVERY]
[WRITECHECK | NOWRITECHECK]
[DESTAGEWAIT | NODESTAGEWAIT]
[KEYS(length offset)]
[MODEL(entryname [/ password]
[catname [/ password][dname]])])]

```
[INDEX
(       [NAME(entryname)]
        [FILE(dname)]
        [VOLUMES(volser[ b volser...])]
        [TRACKS(primary[ b secondary]) |
           CYLINDERS(primary[ b secondary]) |
           RECORDS(primary[ b secondary])]
        [UNIQUE | SUBALLOCATION]
        [ORDERED | UNORDERED]
        [CONTROLINTERVALSIZE(size)]
        [STAGE | BIND | CYLINDERFAULT]
        [MASTERPW(password)]
        [CONTROLPW(password)]
        [UPDATEPW(password)]
        [READPW(password)]
        [CODE(code)]
        [ATTEMPTS(number)]
        [AUTHORIZATION(entrypoint[ b string])]
        [OWNER(ownerid)]
        [SHAREOPTIONS(crossregion[ b crosssystem])]
        [WRITECHECK | NOWRITECHECK]
        [DESTAGEWAIT | NODESTAGEWAIT]
        [REPLICATE | NOREPLICATE]
        [MODEL(entryname[/password]
              [ b catname[/password][ b dname]])]
        [IMBED | NOIMBED]       )]

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

DEFINE can be abbreviated DEF. The parameters of this command are described in the following groups:

- Entry type, which describes the CLUSTER, DATA, and INDEX parameters. These parameters govern which attributes are to be associated with which entry.
- Name, which describes the NAME parameter.
- Data organization, which describes the INDEXED, NONINDEXED KEYS, REPLICATE, NOREPLICATE, IMBED, and NOIMBED parameters. These parameters specify whether the data is to be key sequenced or entry sequenced; for key-sequenced data, these parameters specify the length and location of the key, whether an index record is to be copied as many times as possible on a track, and whether the lowest level of the index (that is, sequence set records) is to be placed next to the data.
- Allocation, which describes the FILE, VOLUMES, TRACKS, CYLINDERS, RECORDS, RECORDSIZE, FREESPACE, UNIQUE, SUBALLOCATION, KEYRANGES, ORDERED, UNORDERED, BUFFERSPACE, CONTROLINTERVALSIZE, STAGE, BIND, and CYLINDERFAULT parameters. These parameters are used to identify the volume on which the cluster or component is to reside; to specify the amount of space to be allocated on each volume; to specify whether the cluster or component is to reside alone in a data space; to specify whether

the data is to be divided by key range among volumes; to specify space to be used for buffers and for control intervals; and to specify how a cluster or component that is stored on a mass storage volume is to be staged.

- Protection and integrity, which describes the MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, AUTHORIZATION, OWNER, TO, FOR, SHAREOPTIONS, ERASE, NOERASE, SPEED, RECOVERY, WRITECHECK, NOWRITECHECK, DESTAGEWAIT, and NODESTAGEWAIT parameters. These parameters are used: to associate passwords with the cluster or one of its components; to provide a mechanism by which the console operator can be prompted to supply a password without disclosing the name of the entry; to identify a user-written routine for additional authorization verification; to identify the owner of the data; to specify share options to be associated with the entry; to indicate whether the data is to be erased when the entry is deleted; to specify whether control intervals are to be preformatted; to indicate whether write-check operations are to be performed; and to indicate whether a cluster or component that is stored on a mass storage volume is to be destaged synchronously or asynchronously with respect to the program that closes it.
- Model, which describes the MODEL parameter. This parameter is used to identify an existing cluster or component entry from which information is to be copied.
- Catalog, which describes the CATALOG parameter. This parameter is used to supply the name and password, when required, of the catalog in which the cluster is to be defined.

Entry Type (Cluster)

The entry-type parameters allow you to associate information with each of the entries—the cluster entry, the data-component entry, and, in the case of an indexed cluster, the index-component entry—created as a result of the DEFINE command.

CLUSTER
specifies that a cluster is to be defined. CLUSTER is followed by the parameters specified for the cluster as a whole; these parameters are enclosed in parentheses and, optionally, are followed by the DATA and/or INDEX parameters and their subparameters, enclosed in parentheses. CLUSTER can be abbreviated CL.

DATA(*options* **)**
specifies attributes of the data component of the cluster; if like attributes are specified for the cluster as a whole, they are overridden by the DATA parameter. The parameters that can be specified for a cluster's data component are: NAME, FILE, VOLUMES, (one of TRACKS, CYLINDERS, or RECORDS), RECORDSIZE, FREESPACE, UNIQUE or SUBALLOCATION, KEYRANGES, ORDERED or UNORDERED, BUFFERSPACE, CONTROLINTERVALSIZE, STAGE, BIND, or CYLINDERFAULT, MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, AUTHORIZATION, OWNER, SHAREOPTIONS, ERASE or NOERASE, SPEED or RECOVERY, WRITECHECK or NOWRITECHECK, DESTAGEWAIT or NODESTAGEWAIT, KEYS and MODEL. The parameters, their abbreviations, defaults, and restrictions, are described later in this chapter. If a name is not specified for the data component, a name is generated and listed for it.

INDEX(*options* **)**
specifies attributes of the index component of the cluster; if these attributes were specified for the cluster as a whole, they are overridden. The parameters that can be specified for a cluster's index component are: FILE, VOLUMES, (one of TRACKS, CYLINDERS, or RECORDS), UNIQUE or SUBALLOCATION, ORDERED or UNORDERED, CONTROLINTERVALSIZE, STAGE, BIND, or CYLINDERFAULT, MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, AUTHORIZATION, OWNER, SHAREOPTIONS, WRITECHECK or NOWRITECHECK, DESTAGEWAIT or NODESTAGEWAIT, REPLICATE or NOREPLICATE, MODEL, and IMBED or NOIMBED. The parameters, their abbreviations, defaults, and restrictions, are described later in this chapter. If a name is not specified for the index component, a name is generated and listed for the index component. INDEX can be abbreviated IX.

Name (Cluster)

The name parameter allows you to specify the name of the cluster and, optionally, the names of its components. A name can be explicitly specified for a component; if no name is specified, a name is generated. Because the cluster, data component, and index component are individually named, each can be addressed.

NAME(*entryname* **)**
specifies the name of the entry (cluster or component) being defined. NAME must be specified for the cluster; it can optionally be specified for a data or index component. The name may contain from 1 through 44 alphanumeric characters, national characters (@, #, and $), and two special characters (the hyphen and the 12-0 overpunch). Names containing more than eight characters must be segmented by periods; one to eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or national character.

Data Organization (Cluster)

When a cluster is defined, you specify whether the data is to be indexed (key sequenced) or nonindexed (entry sequenced). For an indexed cluster, you may also specify index options. The index-options include whether an index record is to be replicated (written as many times as it will fit on a track) and whether the sequence set, the lowest level of the index, is to be placed with the data component.

Index records can be replicated in these combinations of sequence set and index set:

- Sequence-set records adjacent to control areas, and only sequence-set records replicated
- Sequence-set records adjacent to control areas, but all index records replicated
- Sequence set and index set together and all index records replicated

Index options can improve performance. See the chapter "Optimizing VSAM's Performance" for information on how index options affect performance.

INDEXED
specifies that the cluster being defined is for key-sequenced data. If INDEXED is specified, an index component is automatically defined and cataloged. If neither INDEXED nor NONINDEXED is specified, INDEXED is the default. INDEXED may be abbreviated IXD.

NONINDEXED
specifies that the cluster being defined is for entry-sequenced data. NONINDEXED may be abbreviated NIXD.

KEYS
specifies that information about the key field for key-sequenced data follows. KEYS may be specified as a parameter of CLUSTER or DATA. This parameter is required for key-sequenced data; it is not allowed for entry-sequenced data.

length b *offset*
specifies the length and offset of the key. The sum of length and offset cannot exceed the length of the shortest record. These values can be expressed in decimal, hexadecimal, or binary. If a value is specified in hexadecimal or binary, it must be preceded by X or B, respectively, be enclosed in single quotation marks, and be no longer than one fullword. The length of the key can be from 1 through 255 bytes.

REPLICATE | NOREPLICATE
specifies whether each index record is to be written on a track as many times as it will fit to reduce rotational delay and improve performance. If neither is coded, NOREPLICATE is the default. This parameter applies only for key-sequenced data. REPLICATE and NOREPLICATE can be abbreviated REPL and NREPL.

IMBED | NOIMBED
specifies whether the sequence set, the lowest level of the index, is to be placed with the data component If IMBED is coded, each sequence-set record for each control area is written as many times as it will fit on a first track that is adjacent to the control area. If neither is coded, NOIMBED is the default. These parameters apply only for key-sequenced data. IMBED and NOIMBED can be specified as either CLUSTER or INDEX parameters. IMBED and NOIMBED can be abbreviated IMBD and NIMBD, respectively.

Allocation (Cluster)

The allocation parameters are used to specify:

- The volume(s) on which a cluster or its components are to reside.
- The amount of space to be allocated.
- For an indexed cluster, the free space to be left in control intervals and control areas.
- Whether the cluster is to reside alone in a data space.
- For an indexed cluster, whether the data is to be divided by key among volumes and, if so, whether the volumes are to be allocated in the order specified.
- The space to be provided for buffers and the size of control intervals.
- How a cluster or component that is stored on a mass storage volume is to be staged.

FILE(*dname* **)**
specifies the name of the DD statement that identifies the devices and volumes to be used for space allocation. FILE can be specified as a CLUSTER, DATA, or INDEX parameter. All volumes must be of the same device type. Therefore, if data and index components are to reside on different device types, FILE must be specified as DATA and INDEX parameters so that separate DD statements can be referenced. If UNIQUE is specified, FILE is required. If FILE is not specified and the volume is physically mounted, the volume identified with the VOLUME parameter is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED.

VOLUMES(*volser* [b *volser* ...])
specifies the volumes to contain the cluster or component. In a system with the Mass Storage System, a cluster or component can be defined on a mass storage volume. VOLUMES must be specified. It can be specified for the cluster as a whole. Alternatively, it can be specified as a parameter of DATA and, if the cluster is key-sequenced, as a parameter of INDEX. A volume serial number, volser, may contain one to six alphameric, national (@, #, and $), and special characters; the special characters include commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs. A volume serial number must be enclosed in single quotation marks if it contains a special character. Single quotation marks within a volume serial number must be coded as two single quotation marks. If the data and index components are to reside on different device types, VOLUMES must be specified as both a DATA and an INDEX parameter. VOLUMES can be abbreviated VOL.

TRACKS(*primary* [b *secondary*]) |
CYLINDERS(*primary* [b *secondary*]) |
RECORDS(*primary* [b *secondary* **)**
specifies the amount of space to be allocated in terms of tracks, cylinders, or number of records. If RECORDS is specified, the space required is calculated in terms of the number of records, but the space is allocated by cylinders; if the space is allocated by a cluster with the UNIQUE attribute, the spacee is rounded up to the nearest cylinder. One of these parameters must be specified as a CLUSTER parameter, a DATA parameter, or as both a DATA and INDEX parameter. TRACKS, CYLINDERS, and RECORDS can be abbreviated TRK, CYL, and REC, respectively.

primary [b *secondary*]
specify the size of the primary and secondary extents to be allocated. If secondary is specified, space for a component can be expanded to include a maximum of 127 extents. For a multivolume component with the UNIQUE attribute, 1 primary and 15 secondary extents are allowed per volume. If KEYRANGES is specified, the amount specified for primary is immediately allocated on each of the specified volumes required for the ranges. Otherwise, the primary amount is allocated only on one volume. The secondary amount can be allocated on all volumes regardless of the specification of KEYRANGES. These values can be

expressed in decimal, hexadecimal, or binary. If the amount is specified in hexadecimal or binary, it must be preceded by X or B, be enclosed in single quotation marks, and cannot be longer than one fullword.

RECORDSIZE(*average* ƀ *maximum* **)**
specifies the average and maximum lengths, in bytes, of the records in the data component. RECORDSIZE must be specified as a parameter of either CLUSTER or DATA. These values can be expressed in decimal, hexadecimal, or binary. If a value is specified in hexadecimal or binary, it must be preceded by X or B, respectively, be enclosed in single quotation marks, and cannot be longer than one fullword. The minimum record size that can be specified is one. The maximum record size cannot exceed control-interval size less seven bytes (maximum control-interval size is 32,768). RECORDSIZE can be abbreviated RECSZ.

FREESPACE(*cipercent*[ƀ *capercent*])
specifies the amount of space that is to be left free after any allocation and after any split of control intervals (cipercent) and control areas (capercent). The amounts are specified as percentages. FREESPACE can be specified as a parameter of CLUSTER or DATA only for key-sequenced data. The percentages, which must be equal to or less than 100, may be expressed in decimal, hexadecimal, or binary. If you specify 100, one record is placed in each control interval and one control interval is placed in each control area. If the percentage is expressed in hexadecimal or binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks. If no value is coded, the default is 0 free space. FREESPACE can be abbreviated FSPC.

UNIQUE | SUBALLOCATION
specifies whether the cluster's components are allocated space of their own or whether a portion of previously defined VSAM data space is to be used for each component. If UNIQUE is specified, the components of the cluster are allocated space of their own and their names appear in the VTOC of the volume(s) under their own names. If SUBALLOCATION is specified, the name of the data space, not of the component, appears in the VTOC. If no value is coded, SUBALLOCATION is the default. If SUBALLOCATION is coded, a data space must exist on the volume on which the cluster or components are to reside. If both UNIQUE and KEYRANGES are specified, each key range must be on a different volume. UNIQUE and SUBALLOCATION can be abbreviated to UNQ and SUBAL, respectively.

KEYRANGES((*lowkey* ƀ*highkey* **)**[ƀ**(** *lowkey* ƀ*highkey* **)...**])
specifies that portions of key-sequenced data are to be placed on different volumes. KEYRANGES interacts with VOLUMES and with ORDERED and UNORDERED. Each key range is associated with a volume specified in VOLUMES. If a volume serial number was duplicated in VOLUMES, multiple key ranges of a cluster or component with the SUBALLOCATION attribute are placed on that volume. If the number of volumes is greater than the number of key ranges, the excess volumes are used for overflow records from any key range without consideration of range boundaries. If there are fewer volumes than key ranges, the excess key ranges are placed on the last volume specified. The maximum number of key-range pairs is 123. Key ranges may not overlap; gaps may exist within a specified set of ranges, but records cannot be inserted within a gap. Keys can contain 1 to 64 characters; if coded in hexadecimal, 1 to 128 characters. All EBCDIC characters are allowed. Keys consisting of

characters must be enclosed in single quotation marks if they contain commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a key must be coded as two single quotation marks if the key is enclosed in single quotation marks. If the key is specified in hexadecimal, it must be preceded by X, and be enclosed in single quotation marks. If both UNIQUE and KEYRANGES are specified for the same entry, each key range must reside on a separate volume. KEYRANGES can be abbreviated KRNG.

lowkey
specifies the low key of the key range. If lowkey is shorter than the actual keys, it will be padded to the right with binary zeros.

highkey
specifies the high key of the key range. If highkey is shorter than the actual keys, it will be padded to the right with binary ones.

ORDERED | UNORDERED
specifies whether volumes are to be used in the order in which they were listed in the VOLUMES parameter. If KEYRANGES is also specified, all of the records within the range specified by the first low-key/high-key pair are placed on the first volume specified in VOLUMES; all of the records within the second range are placed on the second volume; etc. If it is impossible to allocate volumes in the given order and ORDERED is specified, the command is terminated. If no value is specified, UNORDERED is the default. ORDERED and UNORDERED can be abbreviated to ORD and UNORD, respectively.

BUFFERSPACE(*size* **)**
specifies the minimum space to be provided for buffers. If BUFFERSPACE is not coded, VSAM attempts to get enough space to contain two data component control intervals and, if the data is key sequenced, one index component control interval. BUFFERSPACE can be specified as a parameter of CLUSTER and DATA. BUFFERSPACE can be abbreviated BUFSPC or BUFSP.

size
is the amount of space to be provided for buffers. This value can be expressed in decimal, hexadecimal, or binary. If it is expressed in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. If size is specified in hexadecimal or binary, it must be preceded by X or B, respectively, be enclosed in single quotation marks, and cannot be longer than one fullword. The size specified cannot be less than enough space to contain two data component control intervals and, if the data is key sequenced, one index control interval.

CONTROLINTERVALSIZE(*size* **)**
specifies the size of the control interval for the cluster or component. This value can be expressed in decimal, hexadecimal, or binary. If it is expressed in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. The size of the control interval depends on the maximum size of logical records and the amount of buffer space provided. Control interval size for a data component can range from 512 bytes to

32,768 bytes; size must be a multiple of 512 when the size is between 512 and 8192. The size must be a multiple of 2048 when the size is between 8192 and 32,768. The size must be at least seven bytes greater than the maximum record size. Control interval size for an index component can be 512, 1024, 2048, or 4096. If the size coded is not a proper multiple, VSAM chooses the next higher multiple. If CONTROLINTERVALSIZE is not coded, VSAM determines the size of control intervals. If you have not specified BUFFERSPACE and the size of your records permits, VSAM uses 2048 for the data component and 512 for the index component. CONTROLINTERVALSIZE can be abbreviated CNVSZ or CISZ.

STAGE | BIND | CYLINDERFAULT
specifies how a cluster or component that is stored on a mass storage volume is to be staged.

STAGE
indicates that the cluster or component is to be staged from mass storage to a direct-access storage staging drive when the cluster or component is opened. If the cluster or component can't be staged at open time because of heavy staging activity of other objects, data is staged as a processing program needs it.

BIND
indicates that the cluster or component is not only to be staged, but also to be bound—that is, retained on the direct-access storage staging drive until it is closed. If the cluster or component can't be staged at open time because of heavy staging activity of other objects, data is staged as a processing program needs it.

CYLINDERFAULT
indicates that the cluster or component is not to be staged when it is opened, but that data from it is to be staged as a processing program needs it. CYLINDERFAULT can be abbreviated CYLF.

When the cluster or component isn't stored on a mass storage volume, the attribute is ineffective until the direct-access storage volume the cluster or component is stored on is converted to a mass storage volume (by way of the CONVERTV command, which is described in *OS/VS Mass Storage System (MSS) Services for Space Management*).

When one of these parameters is specified for the data component and another parameter is specified for the index component, the components are staged separately as specified, except when the sequence set of the index component is imbedded in the data. In that case, the parameter specified for the index component applies to both components.

Protection and Integrity (Cluster)

The protection and integrity parameters permit you to:

- Specify passwords to be associated with the cluster or its data or index components.
- Specify a prompting code and number of attempts allowed to provide the correct password in response to prompting at the operator's console.
- Specify a user-supplied authorization-verification routine.
- Identify the owner of the cluster or its data or index components.

- Specify a retention period and whether the cluster's data component is to be erased when its entry is deleted.
- Specify the share options to be associated with the cluster or its data or index components.
- Specify whether space is to be preformatted before data is initially loaded and whether write-check operations are to be performed as records are inserted in the data set.
- Specify whether a cluster or component that is stored on a mass storage volume is to be destaged synchronously or asynchronously with respect to the program that closes it.

MASTERPW(*password* **)**
specifies a master level password for the entry being defined. The AUTHORIZATION, CODE, and ATTEMPTS parameters have no effect unless the entry has a master password associated with it. If MASTERPW is not specified, the highest level password specified becomes the password for all higher levels. The master password allows all operations. MASTERPW can be abbreviated MRPW.

CONTROLPW(*password* **)**
specifies a control level password for the entry being defined. The control level password permits read and write operations using control-interval access to read, write, and update control intervals that contain the cluster's records.

UPDATEPW(*password* **)**
specifies an update level password for the entry being defined. The update level password permits read and write operations against the cluster's records. UPDATEPW can be abbreviated UPDPW.

READPW(*password* **)**
specifies a read level password for the entry being defined. The read level password permits read operations against the cluster's records. READPW can be abbreviated RDPW.

password
is a one-to-eight EBCDIC character password. If the password contains commas, semicolons, blanks, parentheses, or slashes, the password must be enclosed in single quotation marks. Single quotation marks within a password must be coded as two single quotation marks if the password is enclosed in single quotation marks. Passwords can be expressed in hexadecimal. If password is specified in hexadecimal, it must be preceded by X and enclosed in single quotation marks.

CODE(*code* **)**
specifies a code name for the entry being defined. If an attempt is made to access a password-protected entry without a password, the code name is used in a prompting message; the code enables the operator to be prompted for the password without disclosing the name of the entry. The code may contain one to eight EBCDIC characters. The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. Code can be specified in hexadecimal, where two hexadecimal characters represent an EBCDIC character. If code is specified in hexadecimal, it must be preceded by X and be enclosed in

single quotation marks. If CODE is not specified and an attempt is made to access a cluster or component that is password protected without supplying a password, the operator is prompted with the name of the entry.

ATTEMPTS(*number* **)**
specifies the maximum number of times the operator can try to enter a correct password in response to a prompting message. The number can be any number, 0 through 7. If 0 is specified, the operator is not prompted and is not allowed to enter a password from the console. If ATTEMPTS is not specified, 2 is the default. This value can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. ATTEMPTS can be abbreviated ATT.

Note to TSO Users: At a TSO terminal, the logon password is checked first before the user is prompted to supply a password for the cluster. Checking the logon password counts as one attempt to obtain a password. If ATTEMPTS is not specified, the user has one attempt to supply the catalog's password because the default is 2.

AUTHORIZATION(*entrypoint* [*string*])
specifies that a user-security-verification routine (USVR) is available for additional security verification. When a protected cluster is accessed and the user supplies a correct password other than the cluster's master password, the USVR receives control. AUTHORIZATION can be abbreviated AUTH.

entrypoint
specifies the name of the user's security verification routine. The name can contain one to eight alphanumeric, national (@, #, and $), or special (the hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or national character.

string
specifies information to be passed to the user-security-verification routine when it receives control to verify authorization. The string must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a string must be coded as two single quotation marks if the string is enclosed in single quotation marks. If string is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks; it may contain up to 255 hexadecimal characters.

OWNER(*ownerid* **)**
specifies the identification of the owner of the entry being defined. The ownerid may contain one to eight EBCDIC characters. The ownerid must be enclosed in single quotation marks if it contains a comma, semicolon, blank, parenthesis, or slash. If a single quotation mark appears within ownerid, it must be coded as two single quotation marks when the ownerid is enclosed in single quotation marks. Ownerid may be coded in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. If ownerid is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks. If Access Method Services is being used interactively with TSO to define a catalog or a cluster, and OWNER is not specified, the TSO user's userid is the default ownerid.

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the cluster being defined. This parameter cannot be specified as a DATA or INDEX parameter.

TO(*date* **)**
specifies the date, in the form yyddd, where yy is the year and ddd is the number (001 through 365) of the day, through which the cluster being defined is to be kept.

FOR(*days* **)**
specifies the number of days for which the cluster being defined is to be kept. This value can be expressed in decimal, hexadecimal, or binary. If it is expressed in hexadecimal or binary, it must be preceded by X or B and be enclosed in sinqle quotation marks. The maximum number that can be specified is 9999. If the number specified is 0 through 1830, the cluster is retained for the number of days specified; if the number is between 1831 and 9999, the cluster is retained through the year 1999. If the number is specified in hexadecimal or binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks. If neither TO nor FOR is specified, the cluster can be deleted at any time.

SHAREOPTIONS(*crossregion*[*crosssystem*])
specifies how a component or cluster can be shared. SHAREOPTIONS can be abbreviated SHR.

crossregion
specifies the amount of sharing allowed among regions. The values that can be specified are:

1
specifies that any number of users can share the component or cluster being defined if only read operations are being performed. When a write operation is being performed, only one user can use the component or cluster.

2
specifies that any number of users can use the component or cluster for read operations even if one user is using it for a write operation.

3
specifies that any number of users can share the component or cluster for both read and write operations; VSAM does not monitor accesses to ensure data integrity.

4
specifies that any number of users can share the component or cluster for both read and write operations; VSAM provides some assistance to ensure data integrity.

crosssystem
specifies the amount of sharing allowed among systems. The values that can be specified are:

3
specifies that any number of users can share the component or cluster for both read and write operations; VSAM does not monitor accesses to ensure data integrity.

4

specifies that any number of users can share the component or cluster for both read and write operations; VSAM provides some assistance to ensure data integrity.

ERASE | NOERASE

specifies whether the cluster's data component is to be erased when its entry in the catalog is deleted. If ERASE is specified, the data component is overwritten with binary zeros when its catalog entry is deleted. If no value is specified, NOERASE is the default. ERASE and NOERASE can be abbreviated to ERAS and NERAS, respectively.

SPEED | RECOVERY

specifies whether storage allocated to the data component is to be preformatted before records are inserted. This parameter can be specified as a parameter of CLUSTER or DATA. It applies only to initial loading. SPEED specifies that space is not to be preformatted. RECOVERY specifies that space is to be preformatted. If no value is specified, RECOVERY is the default. SPEED cannot be abbreviated. RECOVERY can be abbreviated to RCVY.

WRITECHECK | NOWRITECHECK

specifies whether the cluster or component is to be checked by a machine action called *write-check* when a record is written into it. If WRITECHECK is specified, a record is written and then read, without data transfer, to test for the data check condition. If no value is coded, NOWRITECHECK is the default. WRITECHECK and NOWRITECHECK can be abbreviated WCK and NWCK.

DESTAGEWAIT | NODESTAGEWAIT

specifies whether a cluster or component that is stored on a mass storage volume is to be destaged synchronously or asynchronously with respect to the program that closes it.

DESTAGEWAIT indicates that destaging is to be completed before VSAM returns control to the program that issued the CLOSE macro. VSAM can thus notify the program whether destaging was successful. DESTAGEWAIT can be abbreviated DSTGW.

NODESTAGEWAIT indicates that notification of unsuccessful destaging is to be made only by a message to the operator and to the messages (SYSPRINT) data set. NODESTAGEWAIT can be abbreviated NDSTGW.

When the cluster or component isn't stored on a mass storage volume, the attribute is ineffective until the direct-access storage volume the cluster or component is stored on is converted to a mass storage volume (by way of the CONVERTV command, which is described in *OS/VS Mass Storage System (MSS) Services for Space Management*).

When one of these parameters is specified for the data component and the other parameter is specified for the index component, the components are destaged separately as specified, except when the sequence set of the index component is imbedded in the data. In that case, the parameter specified for the index component applies to both components.

Model (Cluster)

You can use an existing cluster's catalog entry as a model for the attributes of the cluster being defined.

You may use some attributes of the model and override others by explicitly specifying them in the definition of the cluster or component. If you do not want to add or change any attributes, you need specify only the entry-type (cluster, data, or index) of the model to be used and the name of the entry to be defined.

If MODEL is used at the cluster level, (1) the attributes of the model are defined for the cluster and (2) then any attributes explicitly specified at the cluster level are defined to override those of the model.

If MODEL is specified at the cluster level but not as a subparameter of DATA or INDEX:

1. the attributes of the model are defined for the cluster's data and index components, then
2. attributes explicitly specified at the cluster level are defined for the cluster's data and index components, then
3. attributes explicitly specified for the cluster's data and index components (using DATA and INDEX subparameters) are defined.

Each step overrides the attributes specified by the previous step.

If MODEL is specified at the cluster level and as a subparameter of DATA or INDEX:

1. attributes explicitly specified at the cluster level are defined for the cluster's data and index components, then
2. attributes of the model specified for the data or index component are defined, finally
3. attributes explicitly specified for the data or index component (using DATA and INDEX subparameters) are defined.

Note: The MODEL parameter is designed to let you easily define data sets that are identical except for their names and security attributes. If you use the MODEL parameter, your job might be terminated because of allocation problems if you explicitly:

- specify a different type of device with the VOLUMES parameter, or
- change the length of the keys with the KEYS parameter, or
- change the size of records, bufferspace, or control intervals with the RECORDSIZE, BUFFERSPACE, or CONTROLINTERVALSIZE parameters, or
- change from UNIQUE to SUBALLOCATION, or vice versa, or
- change the unit of allocation with the TRACKS, CYLINDERS, or RECORDS parameters.

MODEL
specifies that an existing entry is to be used as a model for the entry being defined.

entryname
specifies the name of the entry to be used as a model. The entry to be used as a model must be of the same entry type as the entry being defined.

password
specifies a password. If the entry to be used as a model is password protected and is cataloged in a master password-protected catalog, either the model entry's master password or its catalog's master password is required. If the model's protection attributes are to be copied, substitute the master password of either the entry being used as a model (following entryname) or its catalog (following catname). If the model's passwords aren't to be copied, any password of the model or its catalog's master password can be used.

catname
specifies the name of the catalog in which the entry to be used as a model is defined. This parameter is required if (1) you are going to specify the password of the catalog that contains the entry instead of specifying the password of the entry itself, or (2) the catalog is not the master catalog or a catalog identified by a JOBCAT or STEPCAT DD statement.

dname
specifies the name of a DD statement that identifies the catalog that contains the entry to be used as a model. The dname specification is ignored; the catalog is dynamically allocated.

Catalog (Cluster)

The catalog parameter is used to supply the name and password, when required, of the catalog in which the cluster is to be defined.

CATALOG(*catname* [*/ password*][b*dname*])
identifies the catalog in which the cluster is to be defined. CATALOG can be abbreviated CAT.

catname
specifies the name of the catalog in which the entry is to be defined.

password
specifies a password. If the catalog is password protected, you must supply the update or higher level password. If no password is specified, VSAM asks the operator or TSO terminal user for the correct password.

dname
specifies the name of a DD statement that identifies the catalog in which the cluster is to be defined. If dname is not specified and the catalog's volume is physically mounted, the catalog's volume is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED. See "Order of Catalog Use: DEFINE" for information about the order in which catalogs are used.

DEFINE CLUSTER Examples

Define a Key-Sequenced Cluster: Example 1

In this example, a key-sequenced cluster is defined. The DATA and INDEX parameters are specified so the cluster's data and index components are explicitly named.

```
//DEFCLU1  JOB     ...
//JOBCAT   DD      DSNAME=D27UCAT1,DISP=SHR
//STEP1    EXEC    PGM=IDCAMS
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE CLUSTER -
               (NAME(MYDATA) -
               VOLUMES(VSER02) -
               RECORDS(1000 500)   ) -
          DATA -
               (NAME(KSDATA) -
               KEYS(15 0) -
               RECORDSIZE(250 250) -
               FREESPACE(20 10) -
               BUFFERSPACE(25000)   ) -
          INDEX -
               (NAME(KSINDEX) -
               IMBED) -
          CATALOG (D27UCAT1/UPPWD27)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DEFINE CLUSTER command builds a cluster entry, a data entry, and an index entry to define the key-sequenced cluster MYDATA. The parameters specified for the cluster as a whole are:

- NAME, which specifies the cluster's name is MYDATA.
- VOLUMES, which specifies that the cluster is to reside on volume VSER02.
- RECORDS, which specifies that the cluster's space allocation is 1,000 data records. When the cluster is extended. it is extended in increments of 500 records. After the space is allocated, VSAM calculates the amount required for the index and subtracts it from the total.

In addition to the parameters specifies for the cluster as a whole, DATA and INDEX subparameters specify values and attributes the apply only to the cluster's data or index component. The parameters specified for the data component of MYDATA are:

- NAME, which specifies that the data component's name is KSDATA.
- KEYS, which specifies that the length of the key field is 15 bytes and that the key field begins in the first byte (byte 0) of each data record.
- RECORDSIZE, which specifies fixed-length records of 250 bytes.

- BUFFERSPACE, which specifies that a minimum of 25000 bytes must be provided for I/O buffers. This large value will improve access time when the cluster's records are processed.
- FREESPACE, which specifies that 20 percent of each control interval and 10 percent of each control area are to be left free when records are loaded into the cluster. After the cluster's records are loaded, the free space can be used to contain new records.

To determine the approximate number of fixed-length records that can be initially be loaded into the cluster's primary extent, use the following formula to determine the available space (AS):

AS= .9*(PAQ – (CI%*PAQ) – (CA&*PAQ*(1–CI%)))

where:

PAQ = primary allocation quantity = 1000
CI% = 20% = .2
CA% = 10% = .1

AS = .9*(1000 – (.2*1000) – (.1*1000*(1–.2)))
= .9*(1000 – 200 –(.1*1000*.8))
= .9*(1000 – 200 – 80)
= .9*720 = 648 records

When the data records are loaded, approximately 648 data records can be loaded into the data component's primary extent. When more than 648 records are loaded, a secondary space allocation results.

The parameters specified for the index component of MYDATA are:

- NAME, which specifies that the index component's name is KSINDEX.
- IMBED, which specifies that sequence-set index records are to be placed in the data component's control areas (the sequence-set records will be replicated automatically).

VSAM assumes that the cluster MYDATA is to be defined in D27UCAT1 because this is the catalog that is identified in the JOBCAT DD statement.

Define a Key-Sequenced Cluster and an Entry-Sequenced Cluster: Example 2

In this example, two VSAM clusters are defined. This example assumes that a VSAM data space that can contain the data sets already exists on volumes VSER02 and VSER03. The first DEFINE command defines a key-sequenced VSAM cluster, EXAMPLE.KSDS1. The second DEFINE command defines an entry-sequenced VSAM cluster, EXAMPLE.ESDS1.

```
//DEFCLU2  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//STEPCAT  DD      DSNAME=D27UCAT1,DISP=SHR
//         DD      DSNAME=D27UCAT2,DISP=SHR
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE CLUSTER -
               (NAME(EXAMPLE.KSDS1) -
               MODEL(MYDATA D27UCAT1) -
               VOLUMES(VSER02) -
               NOIMBED   ) -
          CATALOG(D27UCAT1/MRPWD27)

     DEFINE CLUSTER -
               (NAME(EXAMPLE.ESDS1) -
               RECORDS(100 500) -
               RECORDSIZE(250 250) -
               VOLUMES(VSER03) -
               NONINDEXED   ) -
          CATALOG(D27UCAT2/MRPWD27)
/*
```

The job control statements are:

- STEPCAT DD, which makes two catalogs available for this job step: D27UCAT1 and D27UCAT2. The DD statements that identify the catalogs are concatenated.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The first DEFINE command builds a cluster entry, a data entry, and an index entry to define the key-sequenced cluster EXAMPLE.KSDS1. Its parameters are:

- NAME, which specifies the name of the key-sequenced cluster, EXAMPLE.KSDS1.
- MODEL, which identifies MYDATA as the cluster to use as a model for EXAMPLE.KSDS1. The attributes and specifications of MYDATA that aren't otherwise specified with the DEFINE command's parameters are used to define the attributes and specifications of EXAMPLE.KSDS1.
- VOLUMES, which specifies that the cluster is to reside on volume VSER02.
- NOIMBED, which specifies that space is not to be allocated for sequence-set control intervals within the data component's physical extents.
- CATALOG, which specifies that the cluster is to be defined in the D27UCAT1 catalog. The master password of D27UCAT1 is MRPWD27.

The second DEFINE command builds a cluster entry and a data entry to define an entry-sequenced cluster EXAMPLE.ESDS1. Its parameters are:

- NAME, which specifies the name of the entry-sequenced cluster, EXAMPLE.ESDS1.
- RECORDS, which specifies that the cluster's space allocation is 100 records. When the cluster is extended, it is entended in increments of 500 records.
- RECORDSIZE, which specifies that the cluster's records are fixed-length (the average record size equals the maximum record size) and 250 bytes long.
- VOLUMES, which specifies that the cluster is to reside on volume VSER03.
- NONINDEXED, which specifies that the cluster is to be an entry-sequenced cluster.
- CATALOG, which specifies that the cluster is to be defined in the D27UCAT2 catalog. The master password of D27UCAT2 is MRPWD27.

Define a Key-Sequenced Cluster (In a Unique Data Space): Example 3

In this example, a key-sequenced cluster is defined. The cluster is unique; that is, it is the only cluster in a VSAM data space.

```
//DEFCLU3  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//STEPCAT  DD      DSNAME=MYCAT,DISP=SHR
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE CLUSTER -
          (NAME(ENTRY)
          RECORDSIZE(80 80) -
          KEYS(10 10) -
          VOLUMES(VSER04) -
          UNIQUE -
          CYLINDERS(5 10)  )
/*
```

The job control statements are:

- STEPCAT DD, which makes a catalog available for this job step: MYCAT.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DEFINE CLUSTER command builds a cluster entry, a data entry, and an index entry to define the key-sequenced cluster ENTRY. The DEFINE CLUSTER command also allocates a data space and allocates it for the cluster's exclusive use. The command's parameters are:

- NAME, which specifies the cluster's name is ENTRY.
- RECORDSIZE, which specifies that the records are fixed-length, 80 byte records.
- KEYS, which specifies that the length of the key field is 10 bytes and that the key field begins in the 11th byte (byte 10) of each data record.

- VOLUMES and UNIQUE, which specify that ENTRY is to reside alone in a data space on volume VSER04. This example assumes that volume VSER04 has enough available space to contain the new data space. This example also assumes that the volume's entry is either in the MYCAT catalog or the master catalog. The volume is dynamically allocated.
- CYLINDERS, which specifies that five cylinders are allocated for the cluster's data space. When the cluster's data or index component is extended, the component is to be extended in increments of ten cylinders.

Define an Entry-Sequenced Cluster (The Cluster has a Generic Name): Example 4

In this example, two entry-sequenced clusters are defined. Each has the generic name "GENERIC.*.BAKER", where the asterisk (*) is replaced with a simple name that uniquely identifies each cluster.

```
//DEFCLU4  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//STEPCAT  DD      DSNAME=D27UCAT1,DISP=OLD
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE CLUSTER -
               (NAME(GENERIC.A.BAKER) -
               VOLUMES(VSER02) -
               RECORDS(100 100) -
               RECORDSIZE(80 80) -
               NONINDEXED  ) -
     CATALOG(D27UCAT1/MRPWD27)
     DEFINE CLUSTER -
               (NAME(GENERIC.B.BAKER) -
               MODEL(GENERIC.A.BAKER D27UCAT1) -
          CATALOG(D27UCAT1/MRPWD27)
/*
```

The job control statements are:

- STEPCAT DD, which makes a catalog available for this job step: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The first DEFINE CLUSTER command defines an entry-sequenced cluster, GENERIC.A.BAKER. Its parameters are:

- NAME, which specifies the name of the entry-sequenced cluster, GENERIC.A.BAKER.
- VOLUMES, which specifies that the cluster is to reside on volume VSER02.
- RECORDS, which specifies that the cluster's space allocation is 100 records. When the cluster is extended, it is extended in increments of 100 records.
- RECORDSIZE, which specifies that the cluster's records are fixed-length (the average record size equals the maximum record size) and 80 bytes long.

- NONINDEXED, which specifies that the cluster is entry-sequenced.
- CATALOG, which specifies that the cluster is to be defined in the D27UCAT1 catalog. The master password of D27UCAT1 is MRPWD27.

The second DEFINE CLUSTER command uses the attributes and specifications of the previously defined cluster, GENERIC.A.BAKER, as a model for the to-be-defined cluster, GENERIC.B.BAKER. Its parameters are:

- NAME, which specifies the name of the entry-sequenced cluster, GENERIC.B.BAKER.
- MODEL, which identifies GENERIC.A.BAKER, cataloged in user catalog D27UCAT, as the cluster to use as a model for GENERIC.B.BAKER. The attributes and specifications of GENERIC.A.BAKER that aren't otherwise specified with the DEFINE command's parameters are used to define the attributes and specifications of GENERIC.B.BAKER.
- CATALOG, which specifies that the cluster is to be defined in the D27UCAT1 catalog. The master password of D27UCAT1 is MRPWD27.

Defining a NonVSAM Data Set

The DEFINE command can be used to catalog a nonVSAM data set in a VSAM catalog. Any already existing data set can be introduced into a master or user catalog through the DEFINE command. The only result of a DEFINE command when it is used to define a nonVSAM entry is that an entry is created in a master or user catalog; no space is allocated or reserved.

The format of the DEFINE command when it is used to define a nonVSAM data set is:

DEFINE	**NONVSAM** **(** **NAME(***entryname***)** **DEVICETYPES(***devtype* **[** ƀ *devtype* ...**])** **VOLUMES(***volser* **[** ƀ*volser* ...**])** **[FILESEQUENCENUMBERS(***number* **[** ƀ*number* ...**])]** **[OWNER(***ownerid***)]** **[TO(***date***) | FOR(***days***)]** **)** **[CATALOG(***catname* **[/***password* **][** ƀ*dname* **])]**

where:

NONVSAM
specifies that a nonVSAM data set is to be defined. DEFINE and NONVSAM can be abbreviated DEF and NVSAM, respectively.

NAME(*entryname* **)**
is a required parameter that specifies the name of the nonVSAM data set being defined. The entryname is the name that appears in the catalog; it is the name used in all future references to the data set. The entryname must be unique within the catalog in which it is defined. The name may consist of 1 through 44 alphanumeric characters, national characters (@, #, and $), and two special characters (the hyphen and the 12-0 overpunch). Names containing more than eight characters must be segmented by periods; one to eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or national character.

DEVICETYPES(*devtypes* **[** ƀ *devtype...* **])**
is a required parameter that specifies the device types of the volumes containing the nonVSAM data set being defined. If the nonVSAM data set resides on different device types, the device types must be specified in the same order as the volume serial numbers listed in the VOLUMES parameter. DEVICETYPES can be abbreviated DEVT. You can specify a device type for any device that is supported by your system. You can specify a generic device type (that is, 3330, 2314, etc.) or a specific unit address (that is, 121, 247, etc.). You cannot specify an esoteric device type (that is, SYSDA or TAPE).

VOLUMES(*volser* **[** ƀ*volser* ... **])**
specifies the volumes to contain the nonVSAM data set. In a system with the Mass Storage System, a nonVSAM data set can be defined on a mass storage volume. You can specify more than one volume serial number if the data set resides on many volumes. If the data set resides on magnetic tape and more than one file belongs to the data set on a single tape volume,

you must repeat the volume's serial number in order to maintain a one-to-one correspondence between the volume serial numbers and the file sequence numbers (FILESEQUENCENUMBERS). A volume serial number, volser, may contain one to six alphameric, national (@, #, and $), and special characters; the special characters include commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs. A volume serial number must be enclosed in single quotation marks if it contains a special character. Single quotation marks within a volume serial number must be coded as two single quotation marks.

FILESEQUENCENUMBERS(*number* [*number ...*] **)**
specifies the file sequence number of the nonVSAM data set being defined. This number indicates the position of the file being defined with respect to other files of the tape. If the data set spans volumes, the file sequence number on each volume must be specified. The numbers must be specified in the same order as the volumes in the VOLUMES parameter. This value can be expressed in decimal, hexadecimal, or binary. If it is expressed in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. FILESEQUENCENUMBERS can be abbreviated FSEQN.

OWNER(*ownerid*
specifies the identification of the owner of the data set being defined. The ownerid may contain one to eight EBCDIC characters. The ownerid must be enclosed in single quotation marks if it contains a comma, semicolon, blank, parenthesis, or slash. If a single quotation mark appears within ownerid, it must be coded as two single quotation marks when the ownerid is enclosed in single quotation marks. Ownerid may be coded in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. If ownerid is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks. If Access Method Services is being used interactively with TSO to define a nonVSAM data set, and OWNER isn't specified, the TSO userid is the default ownerid.

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the nonVSAM data set being defined.

TO(*date* **)**
specifies the date, in the form yyddd, where yy is the year and ddd is the number (001 through 365) of the day, through which the nonVSAM data set being defined is to be kept.

FOR(*days* **)**
specifies the number of days for which the nonVSAM data set being defined is to be kept. This value can be expressed in decimal, hexadecimal, or binary. If it is expressed in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. The maximum number that can be specified is 9999. If the number specified is 0 through 1830, the data set is retained for the number of days specified; if the number is between 1831 and 9999, the data set is retained through the year 1999. If neither TO nor FOR is specified, the data set can be deleted at any time.

CATALOG(*catname* [*/ password*][*dname*])
identifies the catalog in which the nonVSAM data set is to be defined.

catname
specifies the name of the catalog in which the entry is to be defined.

password
specifies a password. If the catalog is password protected, you must supply the update or higher level password.

dname
specifies the name of a DD statement that identifies the catalog in which the nonVSAM data set is to be defined. If dname is not specified and the catalog's volume is physically mounted, the catalog's volume is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED. See "Order of Catalog Use: DEFINE" for information about the order in which catalogs are used.

Note: Relative generation numbers cannot be used in the DEFINE command when it is used to attach a nonVSAM data set to a generation data group.

DEFINE NONVSAM Example

Define a NonVSAM Data Set: Example 1

In this example, two existing nonVSAM data sets are defined in a VSAM catalog, D27UCAT1. The DEFINE NONVSAM command cannot be used to create a nonVSAM data set because the command doesn't allocate space.

```
//DEFNVS   JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//STEPCAT  DD     DSNAME=D27UCAT1,DISP=SHR
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DEFINE NONVSAM -
               (NAME(EXAMPLE.NONVSAM) -
               DEVICETYPES(2314) -
               VOLUMES(VSER02) ) -
     CATALOG(D27UCAT1/MRPWD27)

     DEFINE NONVSAM -
               (NAME(EXAMPLE.NONVSAM2) -
               DEVICETYPES(2314) -
               VOLUMES(VSER02) ) -
          CATALOG(D27UCAT1/MRPWD27)
/*
```

The job control statements are:

- STEPCAT DD, which makes a catalog available for this job step: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

Both DEFINE NONVSAM commands define a nonVSAM data set in catalog D27UCAT1. The DEFINE NONVSAM commands' parameters are:

- NAME, which specifies the name of the nonVSAM data sets, EXAMPLE.NONVSAM and EXAMPLE.NONVSAM2.
- DEVICETYPES, which specifies the type of device that contains the nonVSAM data sets, 2314 Direct Access Storage Facility.
- VOLUMES, which specify the volume that contains the nonVSAM data sets, VSER02.
- CATALOG, which identifies the catalog that is to contain the nonVSAM entries, D27UCAT1, and its update (or higher level) password, MRPWD27.

Defining an Alternate Name

The DEFINE command can be used to define an alternate name—an alias—for nonVSAM data set (even if it is a generation data set). DEFINE ALIAS can also be used to define an alias for a user catalog connector in the master catalog. An alias cannot, however, be defined for a generation data group.

DEFINE (Alias)

The format of the DEFINE command when it is used to define an alias is:

DEFINE	**ALIAS** **(** **NAME(***aliasname***)** **RELATE(***entryname***)** **)** **[CATALOG(***catname* **[/** *password* **])]**

where:

NAME(*aliasname* **)**
specifies the alias.

RELATE(*entryname* **)**
specifies the name of the entry for which an alias is being defined. RELATE can be abbreviated REL.

An alias entry must reside in the same catalog as the entry to which it is related. When an alias is defined, catalogs are searched for the related entry. When the related entry is found, the alias entry is added to the catalog. An alias must be unique within a catalog. An alias can be defined only for a nonVSAM data set or a user catalog connector in the master catalog.

Catalog (Alias)

The catalog parameter is used to supply the name and password, when required, of the catalog in which the alias is to be defined.

CATALOG(*catname* [/ *password*] **)**
identifies the catalog in which the alias is to be defined. CATALOG can be abbreviated CAT.

catname
specifies the name of the catalog. When the alias is for a user catalog connector, *catname* is the name of the master catalog.

password
specifies a password. If the catalog is password protected, you must supply the catalog's update or higher level password. If no password is specified and the catalog is password protected, VSAM asks the operator or TSO terminal user for the correct password.

How to Use an Alias to Identify a User Catalog

When you define an alias for a user catalog connector, you should structure the alias so that the catalog's cataloged data sets can be located when the alias is used. If VSAM is searching[1] for a user-specified entryname (that identifies a data set) and doesn't find its entry in the master catalog or a user catalog identified with the JOBCAT or STEPCAT DD statements, and if the entryname's catalog is not specified, VSAM assumes that:

- The entry resides in a user catalog, *and*
- The user catalog's name is the first simple name of the qualified entryname.

If the entryname is ABC.DE.DATA, VSAM searches the master catalog for a user-catalog connector entry (that is, an entry with the name or alias of ABC). If found, VSAM next searches catalog ABC for an entry identified by the name or alias ABC.DE.DATA.

If the user identifies a catalog with an alias of ABC.DE and catalogs the entry whose name is ABC.DE.DATA in that catalog, the catalog won't be found when VSAM searches for it. In order for VSAM to find the entryname ABC.DE.DATA (an entry in the catalog whose alias is ABC.DE), the catalog's alias, ABC.DE, must also be specified (either explicitly in the Access Method Services command that refers to data set ABC.DE.DATA, or as a JOBCAT or STEPCAT DD statement).

DEFINE ALIAS Example

Define an Alias for a NonVSAM Data Set: Example

In this example, an alias is defined for a nonVSAM data set.

```
//DEFALS   JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//STEPCAT  DD     DSNAME=D27UCAT1,DISP=OLD
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DEFINE ALIAS -
          (NAME(EXAMPLE.NONVSAM1) -
          RELATE(EXAMPLE.NONVSAM) )
/*
```

The job control statements are:

- STEPCAT DD, which makes a catalog available for this job step: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DEFINE ALIAS command defines an alias, EXAMPLE.NONVSAM1, for the nonVSAM data set EXAMPLE.NONVSAM. Its parameters are:

- NAME, which specifies the alias (alternate entryname), EXAMPLE.NONVSAM1.
- RELATE, which specifies the name that the alias is an alternate entryname for, EXAMPLE.NONVSAM.

---

[1] For a detailed description of, and the correct search sequence for, how VSAM searches catalogs to locate an entry, see the previous section "Order of Catalog Use: DEFINE" near the beginning of the "Defining Entries" chapter.

Defining a Generation Data Group

The DEFINE command can be used to create a catalog entry for a generation data group.

Once a catalog entry for a generation data group has been created, existing data sets can be attached to it as generation data sets. This is accomplished through JCL, in the form DSNAME=name(+1), or by defining each data set as a nonVSAM data set with a generation data set name, as follows:

name.GnnnnVnn

If the *name* is found to match the name of a previously defined generation data group, the nonVSAM data set is associated with the generation data group as a generation data set. See "Defining a NonVSAM Data Set" for a description of how to define a nonVSAM entry.

DEFINE (Generation Data Group)

The format of the DEFINE command when it is used to define a generation data group is:

DEFINE	**GENERATIONDATAGROUP** **(** **NAME(***entryname***)** **LIMIT(***limit***)** **[EMPTY | NOEMPTY]** **[SCRATCH | NOSCRATCH]** **[OWNER(***ownerid***)]** **[TO(***date***) | FOR(***days***)]** **)** **[CATALOG(***catname* **[/***password***])]**

where:

GENERATIONDATAGROUP
specifies that a generation-data-group entry is to be defined. GENERATIONDATAGROUP can be abbreviated GDG.

NAME(*entryname* **)**
specifies the name of the generation data group that is being defined. The name may consist of 1 through 35 alphanumeric characters, national characters (@, #, and $), and two special characters (the hyphen and the 12-0 overpunch). If the name greater than eight characters, it is specified as a qualified name. A qualified name is segmented by periods; one to eight characters can be specified between periods. The first character of any name or name segment must be either an alphabetic or national character.

LIMIT(*limit* **)**
specifies the maximum number of generation data sets that can be associated with the generation data group to be defined. This parameter is required. The minimum number that can be specified is 1. The maximum number that can be specified is 255. This value can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. LIMIT can be abbreviated LIM.

EMPTY | NOEMPTY
specifies what action is to be taken when the maximum number of generation data sets has been reached for the generation data group and another generation data set is to be cataloged. If no value is coded, NOEMPTY is the default.

EMPTY
specifies that all of the generation data sets are to be uncataloged when the limit is reached. EMPTY can be abbreviated EMP.

NOEMPTY
specifies that only the oldest generation data set is to be uncataloged when the limit is reached. NOEMPTY can be abbreviated NEMP.

SCRATCH | NOSCRATCH
specifies whether a generation data set is to be deleted from the VTOC when it is uncataloged. If no value is coded, NOSCRATCH is the default.

SCRATCH
specifies that the generation data set is to be removed from the VTOC of the volume on which it resides when it is uncataloged. SCRATCH can be abbreviated SCR.

NOSCRATCH
specifies that the generation data set is not to be removed from the VTOC of the volume on which it resides when it is uncataloged. NOSCRATCH can be abbreviated NSCR.

OWNER(*ownerid* **)**
specifies the identification of the owner of the generation data group being defined. The ownerid may contain one to eight EBCDIC characters. The ownerid must be enclosed in single quotation marks if it contains a comma, semicolon, blank, parenthesis, or slash. If a single quotation mark appears within ownerid, it must be coded as two single quotation marks when the ownerid is enclosed in single quotation marks. Ownerid may be coded in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. If ownerid is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks. If Access Method Services is being used interactively with TSO to define a generation data group, and OWNER isn't specified, the TSO userid is the default ownerid.

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the geration data group being defined.

TO(*date* **)**
specifies the date, in the form yyddd, where yy is the year and ddd is the number (001 through 365) of the day, through which the generation data group being defined is to be kept.

FOR(*days* **)**
specifies the number of days for which the generation data group being defined is to be kept. This value can be expressed in decimal, hexadecimal, or binary. If it is expressed in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. The maximum number that can be specified is 9999. If the number specified is 0 through 1830, the generation data group is retained for the number of days specified; if the number is between 1831 and 9999, the generation data group is retained through the year 1999. If neither TO

nor FOR is specified, the generation data group can be deleted at any time.

CATALOG(*catname* [**/** *password*] **)**
identifies the catalog in which the generation data group is to be defined. CATALOG can be abbreviated CAT.

catname
specifies the name of the catalog in which the entry is to be defined.

password
specifies a password. If the catalog is password protected, you must supply the update or higher level password.

DEFINE GENERATIONDATAGROUP Example

Define a Generation Data Group and a Generation Data Set Within It: Example

In this example, a generation data group is defined in the master catalog. Next, a generation data set is defined within the generation data group by using JCL statements.

```
//DEFGDG    JOB     ...
//STEP1     EXEC    PGM=IDCAMS
//GDGMOD    DD      DSNAME=GDG01,DISP=(,KEEP),
//          SPACE=(TRK,(0)),UNIT=2314,VOL=SER=VSER03,
//          DCB=(RECFM=FB,BLKSIZE=2000,LRECL=100)
//SYSPRINT  DD      SYSOUT=A
//SYSIN     DD      *
     DEFINE GENERATIONDATAGROUP -
          (NAME(GDG01) -
          EMPTY -
          NOSCRATCH -
          LIMIT(255) )
/*
//STEP2     EXEC    PGM=IEFBR14
//GDGDD1    DD      DSNAME=GDG01(+1),DISP=(NEW,CATLG),
//          SPACE=(TRK,(10,5)),VOL=SER=VSER03,
//          UNIT=2314
//SYSPRINT  DD      SYSOUT=A
//SYSIN     DD      *
/*
```

The job control statements are:

- GDGMOD DD, which describes the generation data group. When the scheduler processes the DD statement, no space is allocated to GDG01.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

Because no catalog was specified in a CATALOG parameter or with a JOBCAT or STEPCAT DD statement, VSAM assumes all entries built for this command sequence are to be cataloged in the master catalog.

The DEFINE GENERATIONDATAGROUP command defines a generation data group base catalog entry, GDG01. Its parameters are:

- NAME, which specifies the name of the generation data group, GDG01. Each generation data set in the group will have the name GDG01.GxxxxVyy, where "xxxx" is the generation number and "yy" is the version number.
- EMPTY, which specifies that all data sets in the group are to be uncataloged by VSAM when the group contains the maximum number of data sets (as specified by the LIMIT parameter and one more generation data set is added to the group).
- NOSCRATCH, which specifies that, when a data set is uncataloged, its DSCB is not to be removed from its volume's VTOC. Therefore, even if a data set is uncataloged, its records can be accessed when it is allocated to a job step with the appropriate JCL DD statement.
- LIMIT, which specifies that the maximum number of generation data sets in the group is 255. The LIMIT parameter is required.

The second step, STEP2, is used to allocate space and catalog a generation data set in the newly defined generation data group. Its job control statements are:

- GDGDD1 DD, which specifies a generation data set in the generation data group. When the Scheduler processes the DD statement, it catalogs the data set and allocates space to it on volume VSER03.
- SYSPRINT DD, which is required in all job steps. The SYSPRINT DD statement identifies the output device to which messages to the programmer are sent.

Defining a Page Space

The DEFINE command can be used to define an entry for a page space—a VS2 system data set that supports the Auxiliary Storage Manager. A page space is a nonindexed data set that is preformatted in its entirety. A page space must reside on a single volume. A page space cannot be opened as a user data set.

A page space is made known to the system as a system data set at system-generation time or through members of a partioned data set: SYS1.PARMLIB. To be used as a page space, a page space must have been defined in a catalog that is currently being used as the master catalog.

DEFINE (Page Space)

The format of the DEFINE command when it is used to define a page space is:

DEFINE	**PAGESPACE** (**NAME(***entryname***)** **VOLUMES(***volser***)** **[FILE(***dname***)]** **{TRACKS(***primary***) |** **CYLINDERS(***primary***) |** **RECORDS(***primary***)}** **[UNIQUE | SUBALLOCATION]** **[MASTERPW(***password***)]** **[CONTROLPW(***password***)]** **[UPDATEPW(***password***)]** **[READPW(***password***)]** **[CODE(***code***)]** **[ATTEMPTS(***number***)]** **[AUTHORIZATION(***entrypoint* [b *string*])]** **[OWNER(***ownerid***)]** **[MODEL(***entryname* [/ *password*] [b *catname* [/ *password*]])]** **[TO(***date***) | FOR(***days***)]**) **[CATALOG(***catname* [/ *password*])]

DEFINE can be abbreviated DEF. The parameters of this command are described in the following groups:

- Entry type, which describes the PAGESPACE parameter.
- Name, which describes the NAME parameter.
- Allocation, which describes the VOLUMES, FILE, TRACKS, CYLINDERS, RECORDS, UNIQUE, and SUBALLOCATION parameters.

- Protection and integrity, which describes the MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, AUTHORIZATION, OWNER, TO, and FOR parameters. These parameters are used to:
  - Associate passwords with the page space.
  - Provide a mechanism by which the console operator can be prompted to supply a password without disclosing the name of the entry.
  - Identify a user-written routine for additional authorization verification.
  - Identify the owner of the page space.
  - Model, which describes the MODEL parameter. This parameter is used to identify an existing page space entry from which attributes are to be copied.
  - Catalog, which describes the CATALOG parameter. This parameter is used to provide the name and password of the catalog in which the page space is to be defined.

Entry Type (Page Space)

The entry-type parameter specifies that a page space entry is to be created as a result of the DEFINE command.

PAGESPACE
specifies that a page space entry is to be created. PAGESPACE can be abbreviated to PGSPC.

Name (Page Space)

The name parameter is used to name the page space that is being defined.

NAME(*entryname* **)**
specifies the name of the entry being defined. The name may contain from 1 through 44 alphanumeric characters, national characters (@, #, and $), and two special characters (the hyphen and the 12-0 overpunch). Names containing more than eight characters must be segmented by periods; one to eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or national character.

Allocation (Page Space)

The allocation parameters are used to specify:

- The amount of space to be allocated.
- Whether the page space is to reside in a data space alone.

VOLUMES(*volser* **)**
specifies the volume that contains the page space. VOLUMES must be specified. It can be specified as a parameter for the page space as a whole. Alternatively, it can be specified as a subparameter of DATA. A volume serial number, volser, can contain one to six alphanumeric, national characters (@, #, and $), and special characters (comma, blank, semicolon, parenthesis, slash, asterisk, period, quotation mark, ampersand, plus sign, hyphen, equal sign). A volume serial number must be enclosed in single quotation marks if it contains a special character (for example, 'VOL/D1'). Single quotation marks within a volume serial number are

coded as two single quotation marks (for example, 'VOL' 'D2'). VOLUMES can be abbreviated VOL.

FILE(*dname* **)**
specifies the name of the DD statement that identifies the device and volume to be allocated to the page space. If UNIQUE is specified, FILE is required. If FILE is omitted, the volume is allocated dynamically. The volume must be mounted as permanently RESIDENT or RESERVED.

TRACKS(*primary* **) |**
CYLINDERS(*primary* **) |**
RECORDS(*primary* **)**
specifies the amount of space that is to be allocated by tracks, cylinders, or number of records. If RECORDS is specified, the space required is calculated in terms of the number of records, but the space is allocated by tracks; if the space is allocated for a page space with the UNIQUE attribute, the space is rounded up to the nearest cylinder. One of these parameters must be specified. TRACKS, CYLINDERS, and RECORDS can be abbreviated TRK, CYL, and REC, respectively.

primary
specifies the size of the extent to be allocated. This value can be expressed in decimal form (0 through 9), in hexadecimal form (up to eight hexadecimal digits, 0 through 9 and A through F), or in binary form (up to 32 binary digits, 0 and 1). If the amount is specified in hexadecimal or binary, it must be preceded by X or B, be enclosed in single quotation marks, and cannot be longer than one fullword.

UNIQUE | SUBALLOCATION
specifies whether this page space is allocated space of its own or whether a portion of previously defined VSAM data space is suballocated for it. If UNIQUE is specified, the page space is allocated space of its own. VSAM generates a name for the page space and builds a DSCB in the volume's table of contents (VTOC) to describe the page space's space. If SUBALLOCATION is specified, the name of the data space, not of the page space, appears in the VTOC. If no value is coded, SUBALLOCATION is the default. If SUBALLOCATION is coded, a data space must be defined on the volume on which the page space is defined. UNIQUE and SUBALLOCATION can be abbreviated to UNQ and SUBAL, respectively.

Protection and Integrity (Page Space)

The protection and integrity parameters permit you to:

- Specify passwords to be associated with the page space.
- Specify a prompting code and number of attempts allowed to provide the correct password in response to prompting at the operator's console.
- Specify a user-supplied authorization verification routine.
- Identify the owner of the page space.
- Specify a retention period for the page space.

The passwords and protection attributes are cataloged in both the page space entry and its data component's entry. You should specify control, update, and

read level passwords to prevent the page space from being accessed if its volume is moved to a VS1 system. A VS2 system automatically prevents users from accessing page spaces and their data components.

The format of the protection and integrity parameters is:

MASTERPW(*password* **)**
specifies a master level password for the entry being defined. The AUTHORIZATION, CODE, and ATTEMPTS parameters have no effect unless the entry has a master password associated with it. If MASTERPW is not specified, the highest level password specified becomes the password for all higher levels. Because the page space is a system data set, it cannot be opened or used by a user's program. MASTERPW can be abbreviated MRPW.

CONTROLPW(*password* **)**
specifies a control level password for the entry being defined. Because the page space is a system data set, it cannot be opened or used by a user's program. CONTROLPW can be abbreviated CTLPW.

UPDATEPW(*password* **)**
specifies an update level password for the entry being defined. Because the page space is a system data set, it cannot be opened or used by a user's program. UPDATEPW can be abbreviated UPDPW.

READPW(*password* **)**
specifies a read level password for the entry being defined. Because the page space is a system data set, it cannot be opened or used by a user's program. READPW can be abbreviated RDPW.

password
is a one-to-eight EBCDIC character password. Passwords can be coded in hexadecimal form, where two hexadecimal characters represent an EBCDIC character. If password is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks. If the password contains commas, semicolons, blanks, parentheses, or slashes. the password must be enclosed in single quotation marks. Single quotation marks within a password must be coded as two single quotation marks if the password is enclosed in single quotation marks.

CODE(*code* **)**
specifies a code name for the entry being defined. If an attempt is made to access a password-protected entry without a password, the code name is used in a prompting message; the code enables the operator to be prompted for the password without disclosing the name of the entry. The code may contain one to eight EBCDIC characters. The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. The code can be specified hexadecimal form, where two hexadecimal characters represent an EBCDIC character. If code is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks. If code is not specified and an attempt is made to access a cluster or component that is password protected without supplying a password, the operator is prompted with the name of the entry.

ATTEMPTS(*number* **)**
specifies the maximum number of times the operator can try to enter a correct password in response to a prompting message. The number can be any number, 0 through 7. When you define a page space, you should specify ATTEMPTS(0), so that the operator is not prompted and is not allowed to enter a password from the console. This value can be expressed in decimal form, in hexadecimal form, or in binary form. If ATTEMPTS is not specified, 2 is the default. ATTEMPTS can be abbreviated ATT.

AUTHORIZATION(*entrypoint* [b *string*])
specifies that a user-security-verification routine (USVR) is available for additional security verification. When a protected page space is accessed and the user supplies a correct password other than the page space's master password, the user security-verification routine (USVR) receives control. AUTHORIZATION can be abbreviated AUTH.

entrypoint
specifies the name of the user's security verification routine. The name can contain one to eight alphanumeric, national (@, #, and $), or special (the hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or national character.

string
specifies information to be passed to the user-security-verification routine when it receives control to verify authorization. The string must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a string must be coded as two single quotation marks if the string is enclosed in single quotation marks. If string is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks; it may contain up to 255 hexadecimal characters.

OWNER(*ownerid* **)**
specifies the identification of the owner of the entry being defined. The ownerid may contain one to eight EBCDIC characters. The ownerid must be enclosed in single quotation marks if it contains a comma, semicolon, blank, parenthesis, or slash. If a single quotation mark appears within ownerid, it must be coded as two single quotation marks when the ownerid is enclosed in single quotation marks. Ownerid can be coded in hexadecimal form, where two hexadecimal characters represent an EBCDIC character. If ownerid is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks. If Access Method Services is being used interactively with TSO to define a catalog, cluster, or page space and OWNER is not specified, the TSO user's userid is the default ownerid.

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the cluster being defined. This parameter cannot be specified as a DATA or INDEX parameter.

TO(*date* **)**
specifies the date, in the form yyddd, where yy is the year and ddd is the number (001 through 365) of the day, through which the cluster being defined is to be kept.

FOR(*days*)
specifies the number of days for which the cluster being defined is to be kept. The number of days can be expressed in decimal form (0 through 9), in hexadecimal digits, form (up to eight hexadecimal 0 through 9 and A through F), or in binary form (up to 32 binary digits, 0 and 1). The maximum number that can be specified is 9999. If the number specified is 0 through 1830, the cluster is retained for the number of days specified; if the number is between 1831 and 9999, the cluster is retained through the year 1999. If the number is specified in hexadecimal or binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks. If neither TO nor FOR is specified, the cluster can be deleted at any time.

Model (Page Space)

It is possible to use an already defined page space as a model for another page space. When one entry is used as a model for another, its attributes are copied as the new entry is defined.

You may use some attributes of the model and override others by explicitly specifying them in the definition of the page space. If you do not want to add or change any attributes, you need specify only the entry-type (page space) of the model to be used and the name of the entry to be defined. If the model includes the UNIQUE attribute, you must specify FILE.

MODEL
specifies that an existing entry is to be used as a model for the entry being defined.

entryname
specifies the name of the entry to be used as a model. The entry to be used as a model must be of the same entry type as the entry being defined.

password
specifies a password. If the entry to be used as a model is password protected and is cataloged in a password-protected catalog, a password is required. If the protection attributes are to be copied, substitute the master password of either the entry being used as a model (following entryname) or the catalog in which the entry being used as a model is defined (following catname). If protection attributes are not to be copied, any password can be used.

catname
specifies the name of the catalog in which the entry to be used as a model is defined. This parameter is required if (1) you are going to specify the password of the catalog that contains the entry instead of specifying the password of the entry itself, or (2) the catalog is not the master catalog or a catalog identified by a JOBCAT or STEPCAT DD statement.

If you use the MODEL parameter and explicitly change from UNIQUE or SUBALLOCATION, or vice versa, or change the unit of allocation through the TRACKS, CYLINDERS, or RECORDS parameters, your job may be terminated because of allocation problems.

Catalog (Page Space)

The catalog parameter is used to supply the name and password, when required, of the catalog in which the page space is to be defined.

CATALOG(*catname* [*/ password*] **)**
specifies the name and password of the catalog in which the page space is to be defined. CATALOG can be abbreviated CAT.

catname
specifies the name of the catalog in which the page space is to be defined.

password
specifies a password. If the catalog is password protected, you must supply the catalog's update or higher level password. If no password is specified, VSAM asks the operator or TSO terminal user for the correct password.

DEFINE PAGESPACE Examples

Define a Suballocated Page Space: Example 1

In this example, a page space is defined.

```
//DEFPGSP1 JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//VOLUME   DD      VOL=SER=VSER05,UNIT=2305,DISP=OLD
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE SPACE -
          (VOLUMES(VSER05) -
          CYLINDERS(20 10) -
          FILE(VOLUME) )
     DEFINE PAGESPACE -
          (NAME(SYS1.PAGE2) -
          CYLINDERS(10) -
          VOLUMES(VSER05) )
/*
```

The job control statements are:

- VOLUME DD, which describes the volume on which the data space is to be deleted.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

Because no catalog is explicitly specified with a CATALOG parameter or with JOBCAT or STEPCAT DD statements, VSAM assumes that the data space and page space are to be defined in the master catalog.

The DEFINE SPACE command defines a data space on volume VSER05. Its parameters are:

- VOLUMES, which specifies that the data space is to reside on volume VSER05.
- CYLINDERS, which specifies that the data space is to be allocated 20 cylinders initially. When the data space is extended, additional space is allocated to it in increments of 10 cylinders.
- FILE, which identifies the VOLUME DD statement. The VOLUME DD statement specifies the volume serial number and device type of the volume that is to contain the data space.

The DEFINE PAGESPACE command defines a page space. The data space occupies a portion of a previously-defined data space on volume VSER05 (see the DEFINE SPACE command above). It is called a suballocated page space. Its parameters are:

- NAME, which specifies the name of the page space, SYS1.PAGE2.
- CYLINDERS, which specifies that the page space is to occupy ten cylinders. The page spaces are never extended.
- VOLUMES, which specifies that the page space is to reside on volume VSER05.

Define a Unique Page Space: Example 2

In this example, a unique page space is defined. A unique page space differs from a suballocated page space in that a unique page space occupies an entire VSAM data space. When a unique page space is defined, a VSAM data space is automatically built for the page space. When space is allocated to a unique page space, the volume's VTOC identifies the space with the name of the page space.

```
//DEFPGSP2 JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     DEFINE PAGESPACE -
          (NAME(SYS1.PAGE1) -
          CYLINDERS(10) -
          VOLUMES(VSER05) -
          UNIQUE)
/*
```

The DEFINE PAGESPACE command defines a page space. Its parameters are:

- NAME, which specifies the name for the page space: SYS1.PAGE1. Because the page space is unique, the page space's name is put into the DSCB (in the volume's VTOC) that describes the space allocated to the page space.
- CYLINDERS, which specifies that the page space occupies 10 cylinders and cannot be extended.
- VOLUMES, which identifies the volume on which the page space is to reside. Because no DD statement describes the volume, the volume is dynamically allocated. Volume VSER05 must be mounted as permanently RESIDENT or RESERVED.
- UNIQUE, which specifies that the page space is to be a unique page space.

Because neither a CATALOG parameter nor a JOBCAT or STEPCAT DD statement was specified, VSAM assumes that the page space is to be defined in the master catalog.

ALTERING ENTRIES

The ALTER command is used to alter attributes in catalog entries. To alter an entry, you need to supply its name and the attributes to be altered.

To alter qualified-named entries—for example, PAYROLL.YR.MO—you may supply all qualifiers or all but one qualifier. If you supply all qualifiers, only one entry is altered. If you supply all but one qualifier, all entries that match the qualifiers supplied may be altered. The unspecified qualifier is indicated by an asterisk (*). If A.* is specified, all two-qualifier entries that have A for a first qualifier may be altered. If A.*.B is specified, all three-qualifier entries that have A for a first qualifier and B for a third qualifier may be altered. This kind of shorthand name is referred to as a generic name.

When a generic name is used to alter catalog entries and a catalog is specified in the ALTER command, only the specified catalog is searched. If no catalog is specified, the STEPCAT, JOBCAT, and master catalog are searched for entries to be altered.

Altering an entry doesn't normally require that the entry's volume be mounted, because the entry's use of space and the availability of space in the volume's data spaces can be determined by examining the catalog. The entry's volume must be mounted whenever the volume's VTOC must be consulted or modified, such as when a data space, a unique component's space, or a catalog's space is to be altered.

A data set or volume can be dynamically allocated by specifying the data-set name or volume serial number if no DD statement is supplied. The volume must be mounted as permanently RESIDENT or RESERVED.

ALTER Command

The format of the ALTER command is:

ALTER	*entryname* [/ *password*] [**NEWNAME(***newname***)**] [**NULLIFY([MASTERPW]** **[CONTROLPW]** **[UPDATEPW]** **[READPW]** **[OWNER]** **[AUTHORIZATION(MODULE | STRING)]** **[RETENTION]** **[CODE])**] [**MASTERPW(***password***)**] [**CONTROLPW(***password***)**] [**UPDATEPW(***password***)**] [**READPW(***password***)**] [**UNINHIBIT | INHIBIT**] [**CODE(***code***)**] [**ATTEMPTS(***number***)**] [**AUTHORIZATION(***entrypoint* [ƀ *string*])] [**OWNER(***ownerid***)**] [**TO(***date***) | FOR(***days***)**] [**SHAREOPTIONS(***crossregion* [ƀ *crosssystem*])] [**ERASE | NOERASE**] [**WRITECHECK | NOWRITECHECK**] [**DESTAGEWAIT | NODESTAGEWAIT**] [**FILE(***dname***)**] [**FREESPACE(***cipercent* [ƀ *capercent*])] [**BUFFERSPACE(***size***)**] [**ADDVOLUMES(***volser* [ƀ *volser*...])] [**REMOVEVOLUMES(***volser* [ƀ *volser*...])] [**STAGE | BIND | CYLINDERFAULT**] [**EMPTY | NOEMPTY**] [**SCRATCH | NOSCRATCH**] [**CATALOG(***catname* [/ *password*][ƀ *dname*])]

The parameters of the ALTER command are described in the following groups:

- Name, which describes the entryname and NEWNAME parameters. These parameters are used to name the entry to be altered and, optionally, to rename the entry.
- Protection and integrity, which describes the NULLIFY, MASTERPW, CONTROLPW, UPDATEPW, READPW, UNINHIBIT, INHIBIT, CODE, ATTEMPTS, AUTHORIZATION, OWNER, TO, FOR, SHAREOPTIONS, ERASE, NOERASE, WRITECHECK, NOWRITECHECK, DESTAGEWAIT, and NODESTAGEWAIT parameters. These parameters are used to alter protection and integrity attributes.
- Allocation, which describes the FILE, FREESPACE, BUFFERSPACE, ADDVOLUMES, REMOVEVOLUMES, STAGE, BIND, and CYLINDERFAULT parameters. These parameters are used to modify the amount of free space to be left in control intervals and control areas, to

modify the amount of buffer space to be provided, to add and remove volumes from the list of volumes to be used as overflow volumes, and to modify the indication how a data or index component that is stored on a mass storage volume is to be staged.

- Generation-data-group attributes, which describes the EMPTY, NOEMPTY, SCRATCH, and NOSCRATCH parameters. These parameters are used to alter what is to happen when the maximum number of generation data sets isets is reached.
- Catalog, which describes the CATALOG parameter. This parameter is used to name the catalog in which the entry to be altered is defined.

Order of Catalog Use: ALTER

The order in which catalogs are searched when an existing entry is to be located in order to alter it is:

- If a catalog is specified in the CATALOG parameter, only that catalog is searched.
- Any user catalog(s) specified in the current job step (STEPCAT) or, if none is specified for the job step, any user catalog(s) specified for the current job (JOBCAT). If more than one catalog is specified for the job step or job, the job-step or job catalogs are searched in order of concatenation.
- If the entry is not found, and the entry's name is a qualified name, and the first qualifier (that is, the first one to eight characters before a period) is the same as:
  - the name of a user catalog, or
  - the alias of a user catalog, or
  - the alias of a control volume,

  that user catalog or control volume is searched.
- The master catalog is searched.

Name (ALTER)

Name parameters are used to name and rename catalog entries.

If generic names are specified, both the entry name and the new name must be generic names. The generic name of both the entries to be altered and the new name consist of leading qualifiers, an asterisk (*), and trailing qualifiers. The leading and trailing qualifiers of the old name are replaced by the leading and trailing qualifiers of the new name. For example, if the generic name of the entries to be altered is A.*.C and the new name is C.*.A, all entries that have A as the first qualifier and C as the third and last qualifier are to be renamed. The new name will have C as a first qualifier and A as the third and last qualifier, as follows:

Old Name	New Name
A.1.C	C.1.A
A.2.C	C.2.A
A.3.C	C.3.A

If the generic name to be altered is A.B.*.D, all entries that have A and B as first and second qualifiers and D as a fourth and last qualifier are to be renamed. If the new name is C.*.DATA, names are changed, as follows:

Old Name	New Name
A.B.1.D	C.1.DATA
A.B.2.D	C.2.DATA
A.B.Z.D	C.Z.DATA

entryname [/ *password*]
is a required parameter that names the entry to be altered and supplies a password. If a member of a partitioned data set is to be renamed, the entryname is specified in the form: pdsname(membername). If you are altering a password-protected entry in a password-protected catalog, you must specify a password. The password can be specified with entryname or in the CATALOG parameter. The password must be the master password for the entry or for the catalog that contains the entry. If a data or index component entry is to be altered, the master password of the cluster, component, or catalog can be supplied.

NEWNAME(*newname* **)**
specifies that the entry to be altered is to be given a new name. The new name may contain 1 to 44 alphanumeric characters, national characters (@, #, and $), and two special characters (the hyphen and 12-0 overpunch). Names that contain more than eight characters must be segmented by periods; one to eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic character or a national character. NEWNAME cannot be specified for a generation data group base, alias, data space, catalog, or component of a catalog. If a member of a partitioned data set is to be renamed, its name and new name are specified in the form: pdsname(membername). NEWNAME can be abbreviated NEWNM.

Protection and Integrity (ALTER)

The protection and integrity parameters can be used to alter protection and integrity information in a catalog entry.

NULLIFY
specifies that the protection attributes identified by the keywords coded as subparameters of NULLIFY are to be nullified. Any attribute specified is nullified before any respecification of attributes is performed. If all levels of passwords are nullified and none are respecified, CODE, AUTHORIZATION, and ATTEMPTS have no effect. NULLIFY cannot be specified for a nonVSAM data set or for a component of a catalog. NULLIFY can be abbreviated NULL.

MASTERPW
specifies that the master password is to be nullified. If a new master password is not specified and if other passwords exist, the highest level password that exists automatically becomes the password for all higher levels, including the master level. MASTERPW can be abbreviated MRPW.

CONTROLPW
specifies that the control level password is to be nullified. CONTROLPW can be abbreviated CTLPW.

UPDATEPW
specifies that the update level password is to be nullified. UPDATEPW can be abbreviated UPDPW.

READPW
specifies that the read level password is to be nullified. READPW can be abbreviated RDPW.

OWNER
specifies that the owner identification is to be nullified.

AUTHORIZATION(MODULE | STRING)
specifies that either the user authorization routine or the user authorization record is to be nullified. When MODULE is specified, the module name is removed from the catalog record, but the module itself is not deleted. If you nullify the user authorization module, the user authorization record (character string) is also nullified. If, however, you nullify the authorization record, the corresponding module is not nullified. AUTHORIZATION, MODULE, and STRING can be abbreviated AUTH, MDLE, and STRG, respectively.

RETENTION
specifies that the retention period, specified in a TO or FOR parameter, is to be nullified. RETENTION cannot be specified for a component. RETENTION can be abbreviated RETN.

CODE
specifies that the code name used for prompting is to be nullified.

MASTERPW(*password* **)**
specifies a master level password for the entry being altered. The AUTHORIZATION, CODE, and ATTEMPTS parameters have no effect unless the entry has a master password associated with it. If MASTERPW is not specified, the highest level password specified becomes the password for all higher levels. The master password allows all operations. MASTERPW can be abbreviated MRPW.

CONTROLPW(*password* **)**
specifies a control level password for the entry being altered.

UPDATEPW(*password* **)**
specifies an update level password for the entry being altered.

READPW(*password* **)**
specifies a read level password for the entry being altered.

password
is a one-to-eight EBCDIC character password. If the password contains commas, semicolons, blanks, parentheses, slashes, or asterisks, the password must be enclosed in single quotation marks. Single quotation marks within a password must be coded as two single quotation marks if the password is enclosed in single quotation marks. Passwords can be coded in hexadecimal form, where two hexadecimal characters represent an EBCDIC character. If password is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks.

UNINHIBIT | INHIBIT
specifies whether the entry being altered can be accessed for any operation or for only read operations. UNINHIBIT specifies that the read-only restriction set by a previous ALTER or EXPORT command is to be removed. INHIBIT specifies that the entry being altered is only to be read. This parameter can only be specified for data and index components of clusters. UNINHIBIT and INHIBIT can be abbreviated UNINH and INH, respectively.

CODE(*code* **)**
specifies a code name for the entry being altered. If an attempt is made to access a password-protected entry without a password, the code name is used in a prompting message; the code enables the operator to be prompted for the password without disclosing the name of the entry. The code may contain one to eight EBCDIC characters. The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. Code can be specified in hexadecimal, where two hexadecimal characters represent an EBCDIC character. If code is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks. CODE cannot be specified for a nonVSAM data set, an alias, a generation data group base, or a catalog's data or index component. If CODE is not specified and an attempt is made to access a cluster or component that is password protected without supplying a password, the operator is prompted with the name of the entry.

ATTEMPTS(*number* **)**
specifies the maximum number of times the operator can try to enter a correct password in response to a prompting message. The number can be any number, 0 through 7. If 0 is specified, the operator is not prompted and is not allowed to enter a password from the console. This value can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. ATTEMPTS can be abbreviated ATT.

AUTHORIZATION(*entrypoint* [b *string*])
specifies that a user-security-verification routine (USVR) is available for additional security verification. AUTHORIZATION can be abbreviated AUTH.

entrypoint
specifies the name of the user's security-verification routine. The name can contain one to eight alphanumeric, national (@, #, and $), or special (the hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or national character.

string
specifies information to be passed to the user-security-verification routine when it receives control to verify authorization. The string must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a string must be coded as two single quotation marks if the string is enclosed in single quotation marks. If string is specified in hexadecimal, it must be preceded by X and enclosed in single quotation marks; it may contain up to 255 hexadecimal characters.

OWNER(*ownerid* **)**
specifies the identification of the owner of the entry being altered. The ownerid may contain one to eight EBCDIC characters. The ownerid must be enclosed in single quotation marks if it contains a comma, semicolon, blank, parenthesis, or slash. If a single quotation mark appears within ownerid, it must be coded as two single quotation marks when the ownerid is enclosed in single quotation marks. Ownerid can be expressed in hexadecimal form, where two hexadecimal characters represent an EBCDIC character. If ownerid is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks.

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the entry being altered. This cannot be specified for aliases or for the data or index components of clusters or catalogs.

TO(*date* **)**
specifies the date, in the form yyddd, where yy is the year and ddd is the number (001 through 365) of the day, through which the entry is to be kept.

FOR(*days* **)**
specifies the number of days for which the entry is to be kept. This value can be expressed in decimal, hexadecimal, or binary. If it is expressed in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. The maximum number that can be specified is 9999. If the number specified is 0 through 1830, the cluster is retained for the number of days specified; if the number is between 1831 and 9999, the cluster is retained through the year 1999.

SHAREOPTIONS(*crossregion* [ð *crosssystem*])
specifies how a data or index component of a cluster can be shared. This can be specified only for the data or index components of clusters.

crossregion
specifies the amount of sharing allowed among regions. The values that can be specified are:

1
specifies that any number of users can share the component, cluster, or catalog if only read operations are being performed. If a write operation is being performed, sharing is not allowed.

2
specifies that any number of users can use the component, cluster, or catalog for read operations even if one user is using it for a write operation.

3
specifies that any number of users can share the component, cluster, or catalog for both read and write operations; VSAM does not monitor accesses to ensure data integrity. When a data record is updated, VSAM holds its control interval in exclusive control until the update operation completes and the control interval has been written to the direct-access device.

4
specifies that any number of users can share the component, cluster, or catalog for both read and write operations; VSAM provides some assistance to ensure data integrity.

crosssystem
specifies the amount of sharing allowed among systems. The values that can be specified are:

3
specifies that any number of users can share the component, cluster, or catalog for both read and write operations; VSAM does not monitor accesses to ensure data integrity.

4
specifies that any number of users can share the component, cluster, or catalog for both read and write operations; VSAM provides some assistance to ensure data integrity.

ERASE | NOERASE
specifies whether the data component is to be erased when its entry in the catalog is deleted. If ERASE is specified, the component is overwritten with binary zeros when its catalog entry is deleted. This parameter is applicable only when a cluster's data component is to be altered. ERASE and NOERASE can be abbreviated to ERAS and NERAS, respectively.

WRITECHECK | NOWRITECHECK
specifies whether a cluster's data or index component is to be checked by a machine action called *write-check* when a record is written into it. If WRITECHECK is specified, a record is written and then read, without data transfer, to test for the data check condition. This parameter can be specified only for data and index components of a cluster. WRITECHECK and NOWRITECHECK can be abbreviated WCK and NWCK, respectively.

DESTAGEWAIT | NODESTAGEWAIT
specifies whether a data or index component that is stored on a mass storage volume is to be destaged synchronously or asynchronously with respect to the program that closes it. Can be specified only for the data or index component of a cluster.

DESTAGEWAIT indicates that destaging is to be completed before control is returned from VSAM to the program that issues the CLOSE macro. VSAM can thus notify the program whether destaging was successful. DESTAGEWAIT can be abbreviated DSTGW.

NODESTAGEWAIT indicates that notification of unsuccessful destaging is to be made only by a message to the operator and to the messages (SYSPRINT) data set. NODESTAGEWAIT can be abbreviated NDSTGW.

Allocation (ALTER)

The allocation parameters permit you to respecify the amount of distributed free space, whether write-check operations are to be performed, and the amount of buffer space to be provided. In addition, you can add or remove volumes from the list of candidate volumes associated with the entry.

FILE(*dname* **)**
specifies the name of a DD statement that identifies the entry to be altered. If FILE is not specified and the volume is physically mounted, the volume specified with the VOLUME parameter is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED.

FREESPACE(*cipercent* [ƀ *capercent*])
specifies the amount of space that is to be left free after any allocation and after any split of control intervals (cipercent) and control areas (capercent). The amounts are specified as percentages. The percentages, which must be equal to or less than 100, may be expressed in decimal, hexadecimal, or binary. If you specify 100 percent of freespace, one record is placed in each control interval and one control interval is placed in each control area. If the percentage is expressed in hexadecimal or binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks. FREESPACE applies only to the data component of a key-sequenced cluster. FREESPACE can be abbreviated FSPC.

BUFFERSPACE(*size* **)**
specifies the minimum space to be provided for buffers. The amount specified should be greater than or equal to the amount specified in the original definition. If the amount is less than was specified when the entry was defined, VSAM attempts to get enough space to contain two data component control intervals and, if the data is key sequenced, one index component control interval. BUFFERSPACE can be specified only for a catalog or the cluster's data component. BUFFERSPACE can be abbreviated BUFSPC or BUFSP.

size
is the amount of space to be provided for buffers. This value can be expressed in decimal, hexadecimal, or binary. If size is specified in hexadecimal or binary, it must be preceded by X or B, respectively, be enclosed in single quotation marks, and cannot be longer than one fullword. If the size specified is less than the amount VSAM requires, VSAM gets the amount it requires when the data set is opened.

ADDVOLUMES(*volser* [ƀ *volser*] **)**
specifies volumes to be added to the list of overflow (candidate) volumes. A volume serial number, volser, may contain one to six alphameric, national (@, #, and $), and special characters; the special characters include commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs. A volume serial number must be enclosed in single quotation marks if it contains a special character. Single quotation marks within a volume serial number must be coded as two single quotation marks. The volumes to be added as candidate volumes must already be owned by the catalog that contains the entry being altered; that is, space must have been defined on a volume to be added or the volume must have been identified as a candidate volume. This parameter can only be specified for the data and index components of clusters. ADDVOLUMES can be abbreviated AVOL.

REMOVEVOLUMES(*volser* [ƀ *volser*] **)**
has two uses:

(1) REMOVEVOLUMES specifies volume(s) to be removed from the list of candidate volumes associated with the entry being altered. Volumes specified are removed after any new volumes are added to the candidate list. If a volume to be removed contains data that belongs to the entry being altered, the volume is not removed.

(2) REMOVEVOLUMES specifies volume(s) from which all VSAM data spaces are to be removed and VSAM ownership is to be taken away—*without* access to the user catalog that owns the volume(s). For this use of REMOVEVOLUMES, the name of the master catalog and its

master password (if any) must be specified in the *entryname* parameter, and the FILE parameter is required. (You can have only one DD statement with this use of ALTER REMOVE VOLUMES—only volumes of the same device type can be processed.) See the section "VSAM Volume Cleanup" in the chapter "Data Security and Integrity" for important information and cautions about this use of ALTER REMOVEVOLUMES.

A volume serial number, volser, may contain one to six alphameric, national (@, #, and $), and special characters; the special characters include commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs. A volume serial number must be enclosed in single quotation marks if it contains a special character. Single quotation marks within a volume serial number must be coded as two single quotation marks. This parameter can only be specified for the data and index components of clusters. REMOVEVOLUMES can be abbreviated RVOL.

STAGE | BIND | CYLINDERFAULT
specifies how a data or index component that is stored on a mass storage volume is to be staged. Can be specified only for the data or index component of a cluster.

STAGE
indicates that the component is to be staged from mass storage to a direct-access storage staging drive when the cluster or component is opened.

BIND
indicates that the component is not only to be staged, but also to be bound—that is, retained on the direct-access storage staging drive until it is closed.

CYLINDERFAULT
indicates that the component is not to be staged when it is opened, but that data from it is to be staged as a processing program needs it. CYLINDERFAULT can be abbreviated CYLF.

Generation-Data-Group Attributes (ALTER)

The generation-data-group-attribute parameters are used to modify the attributes of a previously defined generation data group.

EMPTY | NOEMPTY
specifies what is to happen when the maximum number of generation data sets has been cataloged. EMPTY specfies that all of the generation data sets are to be uncataloged; NOEMPTY specifies that only the oldest generation data set is to be uncataloged. EMPTY and NOEMPTY can be abbreviated EMP and NEMP, respectively.

SCRATCH | NOSCRATCH
specifies whether generation data sets are to be removed from the VTOC of the volume on which they reside when they are uncataloged. SCRATCH and NOSCRATCH can be abbreviated SCR and NSCR, respectively.

Ownerid and retention period can also be nullified or modified for a generation data group. The maximum number of generation data groups cannot be changed.

Catalog (ALTER)

The catalog parameter is used to name the catalog and its password, when required, of the catalog in which the entry to be altered resides.

CATALOG
specifies the catalog location of the entry to be altered. CATALOG can be abbreviated CAT. See "Order of Catalog Use: ALTER" for information about the order in which catalogs are searched.

catname
specifies the name of the catalog that contains the entry.

password
specifies the master password of the catalog that contains the entry to be altered. If the entry to be altered is password protected and the catalog is also password protected, a password must be entered either through this parameter or through the parameter that specifies the entry to be altered.

dname
specifies the name of the DD statement that identifies the catalog that contains the entry to be altered when that catalog is not the catalog obtained by default. The dname specification is ignored. See "Order of Catalog Use: ALTER" for information about the order in which catalogs are searched.

ALTER Examples

Alter a Cluster's Entry: Example 1

In this example, an ALTER command is used to specify passwords for a nonindexed (entry-sequenced) cluster, EXAMPLE.ESDS1. No password for the cluster is required, because the cluster was defined without passwords.

```
//ALTER1   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT2,DISP=SHR
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     ALTER -
           EXAMPLE.ESDS1 -
           MASTERPW(DEPT26M) -
           CONTROLPW(DEPT26C) -
           UPDATEPW(DEPT26U) -
           READPW(DEPT26R) -
           AUTHORIZATION(D26AUTH)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT2.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The ALTER command adds passwords to the entry-sequenced cluster's cluster catalog entry. The passwords are not added to the cluster's data entry, however. If a user's program supplies the cluster's data-entry entryname and opens the data component, the unauthorized user can access the cluster's data records even though the cluster itself is password protected. The ALTER command's parameters are:

- EXAMPLE.ESDS1, the name of the entry-sequenced cluster.
- MASTERPW, CONTROLPW, UPDATEPW, READPW, and AUTHORIZATION, which specify passwords and the entryname of the user's security-verification routine.

Alter the Entryname's of Generically Named Clusters: Example 2

In this example, several clusters with similar names, GENERIC.*.BAKER (where "*" is any 1 to 8 character simple name), are renamed so that their entrynames are GENERIC.*.ABLE. The name "GENERIC.*.BAKER" is called a generic name.

```
//ALTER2   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT1,DISP=SHR
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     ALTER -
          GENERIC.*.BAKER -
          NEWNAME(GENERIC.*.ABLE)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The ALTER command changes each generic entryname, GENERIC.*.BAKER, to GENERIC.*.ABLE. Its parameters are:

- GENERIC.*.BAKER, which identifies the objects to be modified.
- NEWNAME, which specifies that each generic entryname GENERIC.*.BAKER (where "*" is any 1 to 8 character simple name) is changed to GENERIC.*.ABLE.

Alter the Attributes of a Generation Data Group: Example 3

In this example, the attributes of a generation data group are modified. Because the attributes of the group are cataloged in the generation data group's base catalog entry, only this entry is modified.

```
//ALTER3   JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     ALTER -
           GDG01 -
           NOEMPTY -
           SCRATCH
/*
```

The ALTER command modifies some of the attributes of generation data group GDG01. Its parameters are:

- GDG01, which identifies the object to be modified.
- NOEMPTY, which specifies that only the oldest generation data set is to be uncataloged when the maximum number of cataloged generation data sets is exceeded.
- SCRATCH, which specifies that the generation data set's DSCB is to be removed from the volume's VTOC when the data set is uncataloged.

The attributes specified for the generation data group with the ALTER command override any attributes previously specified for the GDG.

LISTING CATALOG ENTRIES

The LISTCAT command is used to list entries from a catalog. The entries listed can be selected by name or entry type, and the fields to be listed for each entry can additionally be selected.

If entries to be listed are selected by name, the name(s) can be indicated in its entirety, by generic-name or by level.

Entries are specified by generic name by supplying all but one qualifier of the name. The qualifier omitted is indicated by an asterisk (*). If you specify A.*, all two-qualifier entries that have A for a first qualifier may be listed. If you specify A.*.B, all three-qualifier entries that have A for a first qualifier and B for a third qualifier may be listed.

The actions for entries specified by level are as follows:

- If A.*.B is specified, all entries, regardless of the number of levels in their names, that have A as a first level and B as a third level may be listed.
- If A is specified, all entries that have A as the first level, regardless of the number of levels in their names, may be listed.
- For a level name, the * may not be used as the last qualifier; if A.* is specified, it is considered to be an error.

See "Appendix B: Interpreting LISTCAT Output" for an explanation of the output produced as a result of the LISTCAT command.

Order of Catalog Use: LISTCAT

When the ENTRIES or LEVEL parameter is not specified, or when the command is not executed through TSO, the order in which catalogs are searched when entries are to be listed using the LISTCAT command is:

- If a catalog is specified in the CATALOG parameter, only that catalog is listed.
- The first user catalog specified in the current job step (STEPCAT) or, if none is specified, the first user catalog specified in the current job (JOBCAT) is listed.
- If no user catalog is specified in the current job step or job, the master catalog is listed.

When the ENTRIES or LEVEL parameter is specified, or when the command is executed through TSO, the order in which catalogs are searched when entries are to be listed using the LISTCAT command is:

- If a catalog is specified in the CATALOG parameter, only that catalog is listed.
- Any user catalog(s) specified in the current job step (STEPCAT) or, if none is specified for the job step, any user catalog(s) specified for the current job (JOBCAT). If more than one catalog is specified for the job step or job, the job-step or job catalogs are listed in order of concatenation.

- If the entry is not found, and the entry's name is a qualified name, and the first qualifier (that is, the first one to eight characters before a period) is the same as:
  - the name of a user catalog, or
  - the alias of a user catalog, or
  - the alias of a control volume,

  that user catalog or control volume is listed.
- The master catalog is listed.

LISTCAT Command

The format of the LISTCAT command is:

LISTCAT	**[CATALOG(***catname* **[/** *password* **][ƀ** *dname* **])]** **[OUTFILE(***dname* **)]** **[ENTRIES(***entryname* **[/** *password* **]** **[ƀ** *entryname* **[/** *password* **]...]) |** **LEVEL(***level* **)]** **[CLUSTER][ƀDATA][ƀINDEX][ƀUSERCATALOG]** **[ƀSPACE][ƀPAGESPACE][ƀALIAS]** **[ƀNONVSAM][ƀGENERATIONDATAGROUP]** **[CREATION(***days* **)]** **[EXPIRATION(***days* **)]** **[ALL | NAME | HISTORY | VOLUME | ALLOCATION]**

where:

LISTCAT
specifies that catalog entries are to be listed. LISTCAT can be abbreviated LISTC.

CATALOG(*catname* **[/** *password* **][ƀ** *dname***])**
specifies the name of the catalog that contains the entries that are to be listed. If CATALOG is coded, only entries from that catalog are listed. See "Order of Catalog Use: LISTCAT" above for information about the order in which catalogs are searched. CATALOG can be abbreviated CAT.

catname
is the name of the catalog.

password
specifies the read level or higher level password of the catalog that contains entries to be listed. If the entries to be listed contain information about password-protected data sets, a password must be supplied either through this parameter or through the ENTRIES parameter. If passwords are to be listed, you must specify the master password.

dname
specifies the name of the DD statement that identifies the catalog to be listed if the desired catalog is not the first one found. The catalog identified by dname cannot be a concatenated catalog. The dname specification is ignored.

OUTFILE(*dname* **)**
specifies a data set other than the SYSPRINT data set to be used as an output data set. The dname identifies a DD statement that in turn identifies the alternate output data set. If OUTFILE is not specified, the entries are listed in the SYSPRINT data set. If an alternate data set is specified, it must meet the requirements shown under "Output Data Sets" in the "Introduction." OUTFILE can be abbreviated OFILE.

ENTRIES(*entryname* [*/ password*][ƀ *entryname* [*/ password*]...]) |
LEVEL(*level* **)**
specifies the names of the entries to be listed. If neither ENTRIES nor LEVEL is coded, no entry-type, creation, or expiration restriction is coded, and TSO is not being used, the entire catalog is listed. For TSO, only the entries associated with the user's prefix are listed.

ENTRIES(*entryname* [*/ password*]
[ƀ *entryname* [*/ password*]...])
specifies the names or generic names of entries to be listed. If you want information about a catalog, the catalog's volume must be physically mounted. Specify the catalog's name as the entry name and also specify ALL. If you want data space information, you must specify the volume serial number (as the entryname) of the volume containing the data space; you must also specify SPACE and ALL. Under TSO, a user is prompted to complete an unqualified name. ENTRIES can be abbreviated ENT.

password
specifies a password when the entry to be listed is password protected and a password was not specified through the CATALOG parameter. The password must be the read or higher level password. If protection attributes are to be listed, you must supply the master password; if no password is supplied, the operator is prompted for each entry's password. Passwords cannot be specified for these types of entry: nonVSAM data set, generation data group base, alias, user-catalog connector, and data space.

LEVEL(*level* **)**
specifies the level of entry names to be listed. LEVEL can be abbreviated LVL.

CLUSTER
specifies that cluster entries are to be listed. If the only entry type specified is CLUSTER, entries for data and index components associated with the clusters are not listed. CLUSTER can be abbreviated CL.

DATA
specifies that entries for data components, excluding the data component of the catalog, are to be listed. If a cluster's name is specified and DATA is coded, only the data-component entry is listed.

INDEX
specifies that entries for index components, excluding the index component of the catalog, are to be listed. If a cluster's name is specified and INDEX is coded, only the index-component entry is listed. INDEX can be abbreviated IX.

USERCATALOG
specifies that catalog connectors are to be listed. USERCATALOG can be abbreviated UCAT.

SPACE
specifies that entries for volumes containing data spaces defined in this catalog are to be listed. Candidate volumes are included. If entries are identified by entryname or level, SPACE can be coded only when no other entry-type restriction is coded. SPACE can be abbreviated SPC.

PAGESPACE
specifies that entries for page spaces are to be listed. PAGESPACE can be abbreviated PGSPC.

ALIAS
specifies that alias entries are to be listed.

NONVSAM
specifies that entries for nonVSAM data sets are to be listed. If a generation data group's name and NONVSAM are specified, the generation data sets associated with the generation data group are listed. NONVSAM can be abbreviated NVSAM.

GENERATIONDATAGROUP
specifies, in a VS2 system only, that entries for generation data groups are to be listed. GENERATIONDATAGROUP can be abbreviated GDG.

CREATION(*days* **)**
specifies that entries of the indicated type (CLUSTER, DATA, etc.) are to be listed only if they were created the specified number of days ago or earlier. CREATION can be abbreviated CREAT.

days
specifies the number of days ago. This value can be expressed in decimal, hexadecimal, or binary. If it is expressed in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. The maximum number that can be specified is 9999; the default is 0, which indicates that all entries are to be listed.

EXPIRATION(*days* **)**
specifies that entries of the indicated type (CLUSTER, DATA, etc.) are to be listed only if they will expire the specified number of days from now or earlier. EXPIRATION can be abbreviated EXPIR.

days
specifies the number of days from now. This value can be expressed in decimal, hexadecimal, or binary. If it is expressed in hexadecimal or binary, it must be preceded by X or B and be enclosed in single quotation marks. The maximum number that can be specified is 9999, which is the default and indicates that all entries are to be listed. 0 indicates that only entries that have already expired are to be listed.

ALL | NAME | HISTORY | VOLUME | ALLOCATION
specifies the fields to be included for each entry listed. If no value is coded, NAME is the default.

ALL
specifies that all fields are to be listed.

NAME
specifies that the name and entry type of the entries are to be listed. With TSO, only the name of each entry associated with the prefix of the TSO user is listed.

HISTORY
specifies that only the following information is to be listed for each entry: name, ownerid, creation date, and expiration date. HISTORY can be abbreviated HIST. It can be specified for CLUSTER, DATA, INDEX, GENERATIONDATAGROUP, NONVSAM, and PAGESPACE.

VOLUME
specifies that the information provided by specifying HISTORY and the volume serial numbers and device types allocated to the entries are to be listed. Volume information is not listed for clusters, aliases, page spaces, or generation data groups. VOLUME can be abbreviated VOL.

ALLOCATION
specifies that the information provided by specifying VOLUME and detailed information about the allocation are to be listed. The information about allocation is listed only for data and index component entries. ALLOCATION can be abbreviated ALLOC.

Note: When LISTCAT is invoked from a TSO terminal and no parameters (besides CATALOG) are specified, the prefix of the TSO user becomes the highest level of entryname qualification. Only the names of those entries associated with the prefix are listed.

LISTCAT Examples

Listing a Key-Sequenced Cluster's Entry: Example 1

In this example, a key-sequenced cluster entry is listed.

```
//LISTCAT1 JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     LISTCAT -
          ENTRIES(EXAMPLE.KSDS1) -
          CLUSTER -
          ALL -
          CATALOG(D27UCAT2/MRPWD27)
/*
```

The LISTCAT command lists the cluster's catalog entry. Its parameters are:

- ENTRIES, which identifies the entry to be listed.
- CLUSTER, which specifies that only the cluster entry is to be listed. If CLUSTER hadn't been specified, the cluster's data and index entries would also be listed.
- ALL, which specifies that all fields of the cluster entry are to be listed.
- CATALOG, which identifies the catalog that contains the cluster entry, D27UCAT2, and specifies its update (or higher level) password, MRPWD27.

Alter a Catalog Entry, Then List the Modified Entry: Example 2

In this example, the freespace attributes for the data component of cluster MYDATA are modified. Next, the cluster entry, data entry, and index entry of MYDATA are listed to determine the effect, if any, the modification has on the cluster's other attributes and specifications.

```
//LISTCAT2 JOB  ...
//JOBCAT   DD   DSNAME=D27UCAT1,DISP=SHR
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     ALTER -
          KSDATA -
          FREESPACE(10 10)
     IF LASTCC = 0 -
          THEN -
          LISTCAT -
                ENTRIES(MYDATA) -
                ALL
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The ALTER command modifies the freespace specifications of the key-sequenced VSAM cluster MYDATA. The command's parameters are:

- KSDATA, which is the entryname of the data component being altered. KSDATA identifies the data component of a key-sequenced VSAM cluster, MYDATA. In order to alter a value that applies only to the cluster's data component, such as FREESPACE does, you must specify the data component's entryname.
- FREESPACE, which specifies the new freespace percentages for the data component's control intervals and control areas.

The IF ... THEN command sequence verifies that the ALTER command completed successfully before the LISTCAT command executes.

The LISTCAT command lists the cluster's entry and its data and index entries. Its parameters are:

- ENTRIES, which specifies the entryname of the object being listed. Because MYDATA is a key-sequenced cluster, the cluster entry, its data entry, and its index entry are listed.
- ALL, which specifies that all fields of each entry are to be listed.

List Catalog Entries: Example 3

In this example, each catalog entry with the name "GENERIC.*.ABLE"is listed, where "*" is any 1 to 8 character simple name. The name "GENERIC.*.ABLE" is a generic name, and this example illustrates how all catalog entries with the same generic name are listed.

```
//LISTCAT3 JOB  ...
//JOBCAT   DD   DSNAME=D27UCAT1,DISP=SHR
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     LISTCAT -
          ENTRIES(GENERIC.*.ABLE) -
          ALL
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The LISTCAT command lists each catalog entry with the name GENERIC.*.ABLE, where "*" is any 1 to 8 character simple name. Its parameters are:

- ENTRIES, whcih specifies the entryname of the object to be listed. Because GENERIC.*.ABLE is a generic name, more than one entry might be listed.
- ALL, which specifies that all fields of each entry are to be listed.

DELETING CATALOG ENTRIES

The DELETE command is used to delete entries from a catalog. If a cluster is to be deleted, its associated components are automatically deleted. If a data space or catalog is to be deleted, it must be empty. If a generation-data-group entry is to be deleted, any generation data sets that belong to it must have been deleted.

When an alias entry is to be deleted, only that entry is deleted. However, when a user catalog connector or nonVSAM data set which also has aliases is deleted, all alias entries for the catalog connector or data set are deleted in addition to the catalog connector's or data set's entry.

Entries can be deleted from more than one catalog with a single DELETE command.

You can do this by specifying many entrynames and omitting the CATALOG parameter. Use JOBCAT or STEPCAT DD statements to identify the catalogs that contain the entries.

Deleting an entry doesn't normally require that the entry's volume be mounted, because the entry's use of space in the volume's data spaces can be determined by examining the catalog. The entry's volume must be mounted whenever the volume's VTOC must be consulted or modified, such as when:

- A data space, a unique component, or a user catalog is to be deleted.
- A cluster is to be deleted and erased.
- A nonVSAM data set is to be deleted and scratched.

A data set or volume can be dynamically allocated by specifying the data-set name or volume serial number if no DD statement is supplied. The volume must be mounted as permanently RESIDENT or RESERVED.

Order of Catalog Use: DELETE

The order in which catalogs are searched when an existing entry is to be located in order to delete it is:

- If a catalog is specified in the CATALOG parameter, only that catalog is searched.
- Any user catalog(s) specified in the current job step (STEPCAT) or if none is specified for the job step, any user catalog(s) specified for the current job (JOBCAT). If more than one catalog is specified for the job step or job, the job-step or job catalogs are searched in order of concatenation.
- If the entry is not found, and the entry's name is a qualified name, and the first qualifier (that is, the first one to eight characters before a period) is the same as:
  - the name of a user catalog, or
  - the alias of a user catalog, or
  - the alias of a control volume,

  that user catalog or control volume is searched.

Restriction: Control volumes are searched when a nonVSAM data set is being deleted.

DELETE Command

The format of the DELETE command is:

DELETE	(*entryname* [/ *password*][ъ *entryname* [/ *password*]...]) **[CATALOG(***catname* [/ *password*][ъ *dname*])] **[FILE(***dname* **)]** **[PURGE | NOPURGE]** **[ERASE | NOERASE]** **[SCRATCH | NOSCRATCH]** **[CLUSTER | SPACE | USERCATALOG |** **NONVSAM | ALIAS |** **GENERATIONDATAGROUP | PAGESPACE]**

where:

DELETE
specifies that an entry is to be removed from a catalog. DELETE can be abbreviated DEL.

(*entryname* [/ *password*][ъ *entryname* [/ *password*]...])
is a required parameter that names the entries to be deleted. If more than one entry is to be deleted, the list of entry names must be enclosed in parentheses. This parameter must be the first parameter following DELETE.

A generic name can identify a group of entries to be deleted. The generic name consists of leading qualifiers, an asterisk (*), and (optionally) trailing qualifiers. For example, if the generic name of the entries to be deleted is A.*.C, all entries that have A as the first qualifier and C as the *third and last* qualifier are deleted. If the catalog contains the following entries:

A.1.C
A.2.C.1
A.2.C
A.3.C

entries A.1.C, A.2.C, and A.3.C will be deleted. The entry for A.2.C.1 will not be deleted.

entryname
is the name of the entry to be deleted, or the volume serial number of the volume that contains a data space to be deleted. If a volume is indicated that contains more than one data space, all empty data spaces are deleted. Data and index components cannot be named for deletion. If a component is to be deleted, its cluster name must be specified; the cluster and component entries are deleted. User catalogs, generation data groups, and data spaces can be deleted only when they are empty. With the exception of catalogs, data spaces, and page spaces, entry types can be intermixed in a list. A catalog, data space, or page space, however, cannot be in a list that contains an entry type other than itself. If a member of a partitioned data set is to be deleted, its name is specified in the form: dsname(membername).

password
specifies a password for a password-protected entry. Passwords may be specified for each entry name or the catalog's password may be specified through the CATALOG parameter for the catalog that contains the entries to be deleted. If you are deleting data-space entries, specify the update or higher level password of the catalog that contains the entry. If you are deleting a cluster and are specifying a password, specify the master password of the entry. If you are deleting a user catalog, the password must be specified with the entry name; the password must be the master password.

CATALOG(*catname* [/ *password*][b *dname*])
specifies the name of the catalog that contains the entries to be deleted. See "Order of Catalog Use: DELETE" for information about the order in which catalogs are searched. This parameter cannot be used when a user catalog is to be deleted. CATALOG can be abbreviated CAT.

catname
identifies the catalog that contains the entry to be deleted.

password
specifies the master password of the catalog that contains the entries to be deleted. If entries to be deleted are password protected and the catalog is also password protected, a password must be supplied either through CATALOG or with the name of each entry to be deleted. If a user catalog is to be deleted, a password, if required, must be supplied in the ENTRIES parameter. If you are deleting a nonVSAM data set or an alias, you must supply the catalog's update or higher level password.

dname
specifies the name of a DD statement that identifies the catalog that contains the entries to be deleted. The dname specification is ignored.

FILE(*dname* **)**
specifies the name of the DD statement that identifies the volume that contains the data set to be deleted or identifies the entry to be deleted. If FILE is not coded and the entry to be deleted is also to be erased, the entry is a data space or a unique data set, or the entry is a nonVSAM data set that is to be scratched, the object is dynamically allocated. The volume must be mounted as permanently RESIDENT or RESERVED. FILE is not applicable when a catalog is to be deleted, however, a JOBCAT or STEPCAT DD statement is required. Concatenated DD statements are not allowed.

If the FILE parameter is omitted, the entryname is dynamically allocated in the following cases:

- A nonVSAM entry is to be deleted and scratched.
- An entry is to be deleted and erased.
- An entry that resides in a data space of its own is to be deleted.

When the last data space on a volume is deleted and the volume is not a candidate volume, the volume's entry is also deleted from the catalog. The effect of deleting the volume entry is that the volume is no longer owned by the catalog, that is, another catalog is free to allocate space on the volume.

PURGE | NOPURGE
specifies whether the entry is to be deleted regardless of the retention period specified. This parameter cannot be used if a data-space entry is to be deleted. If neither PURGE nor NOPURGE is coded, NOPURGE is the default.

PURGE
specifies that the entry is to be deleted even if the retention period, specified in the TO or FOR parameter, has not expired. PURGE can be abbreviated PRG.

NOPURGE
specifies that the entry is not to be deleted if the retention period has not expired. If NOPURGE is coded and the retention period has not expired, the entry is not deleted. NOPURGE can be abbreviated NPRG.

ERASE | NOERASE
specifies whether the data component of the cluster to be deleted is to be erased, that is, overwritten with binary zeros. This parameter will override whatever was coded when the cluster was defined or last altered. This parameter should be specified only when a cluster entry is to be deleted. When you specify ERASE, you must identify the to-be-deleted entry's catalog with a JOBCAT or STEPCAT DD statement unless:

- the entry is in the master catalog, or
- the first simple name qualifier in the entry's qualified name identifies the catalog (that is, the first simple name is the catalog's name or alias).

ERASE
specifies that the data component is to be overwritten with binary zeros when the cluster is deleted. If ERASE is specified, the volume that contains the data component must be mounted. The data component can be dynamically allocated. ERASE can be abbreviated ERAS.

NOERASE
specifies that the data component is not to be overwritten with binary zeros when the cluster is deleted. NOERASE can be abbreviated NERAS.

SCRATCH | NOSCRATCH
has two uses:

(1) With NONVSAM, SCRATCH | NOSCRATCH specifies whether a nonVSAM data set is to be scratched—removed from the VTOC—of the volume on which it resides.

(2) With CLUSTER, SPACE, or PAGESPACE, NOSCRATCH specifies that the entry is to be deleted from the catalog *without* access to the volume that contains the object defined by the entry. See the section "VSAM Catalog Cleanup" in the chapter "Data Security and Integrity" for important information about the use of DELETE NOSCRATCH.

When you specify SCRATCH, you must identify the to-be-deleted entry's catalog with a JOBCAT or STEPCAT DD statement, unless:

- the entry is in the master catalog, or
- the first simple name in the entry's qualified dsname identifies the catalog (that is, the first simple name is the catalog's name or alias).

If neither SCRATCH nor NOSCRATCH is specified, SCRATCH is the default. SCRATCH and NOSCRATCH can be abbreviated SCR and NSCR, respectively.

CLUSTER | USERCATALOG | SPACE | NONVSAM | ALIAS | GENERATIONDATAGROUP | PAGESPACE

specifies the type of the entry to be deleted. If the entry to be deleted is a catalog or data space entry, USERCATALOG or SPACE is required. If one of these values is coded and the named entry is not of the specified type, the command is terminated. The master catalog cannot be deleted.

CLUSTER

specifies that the entry to be deleted is a cluster entry. CLUSTER can be abbreviated CL.

USERCATALOG

specifies that the entry to be deleted is a user-catalog entry. The catalog connector entry in the master catalog is deleted. If the user catalog has aliases, all of the catalog's alias entries in the master catalog are deleted. This parameter must be specified if a user catalog is to be deleted. You must also include a single STEPCAT or JOBCAT DD statement that identifies the user catalog to be deleted. A user catalog can be deleted only if it is empty. USERCATALOG can be abbreviated UCAT.

SPACE

specifies that the entry to be deleted is a volume entry. This parameter is required if all empty data spaces on a volume are to be deleted. A data space can be deleted only if it is empty. If all data spaces have been deleted and the volume is not a candidate volume, its volume entry is also deleted. SPACE can be abbreviated SPC.

NONVSAM

specifies that the entry to be deleted is a nonVSAM data set entry. If the nonVSAM data set has aliases, all of its alias entries are deleted. If the NONVSAM data set is partitioned, you can delete one of its members by specifying pdsname (member name). NONVSAM can be abbreviated NVSAM. If the NONVSAM data set is partitioned, you can delete one of its members by specifying pdsname (member name).

ALIAS

specifies that the entry to be deleted is an alias entry.

GENERATIONDATAGROUP

specifies that the entry to be deleted is a generation-data-group entry. A generation data group can be deleted only if it is empty. GENERATIONDATAGROUP can be abbreviated GDG.

PAGESPACE

specifies that an inactive page space is to be deleted. A page space is identified as "active" during the operator's IPL procedure. PAGESPACE can be abbreviated PGSPC.

DELETE Examples

Deleting an Alias Entry: Example 1

In this example, an alias entry, EXAMPLE.NONVSAM1, is removed from catalog D27UCAT1.

```
//DELET1   JOB  ...
//JOBCAT   DD   DSNAME=D27UCAT1,DISP=OLD
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     DELETE -
          EXAMPLE.NONVSAM1 -
          ALIAS -
          CATALOG(D27UCAT1/MRPWD27)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DELETE command removes an alias entry from catalog D27UCAT1. Its parameters are:

- EXAMPLE.NONVSAM1, the entryname of the object to be deleted. EXAMPLE.NONVSAM1 identifies an alias entry.
- ALIAS, which specifies the type of entry to be deleted. VSAM verifies that EXAMPLE.NONVSAM1 is an alias entry, then deletes it. If EXAMPLE.NONVSAM1 incorrectly identified another entry by mistake, VSAM would not delete the entry, but would note the discrepancy with a message to the programmer.
- CATALOG, which identifies the catalog that contains the entry, D27UCAT1, and its master password, MRPWD27.

Deleting a Key-Sequenced VSAM Cluster: Example 2

In this example, a key-sequenced cluster is deleted.

```
//DELET2   JOB  ...
//JOBCAT   DD   DSNAME=D27UCAT2,DISP=SHR
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     DELETE -
          EXAMPLE.KSDS1 -
          PURGE -
          ERASE -
          CATALOG(D27UCAT2/MRPWD27)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for the job: D27UCAT2.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DELETE command deletes the key-sequenced VSAM cluster from the D27UCAT2 catalog. Its parameters are:

- EXAMPLE.KSDS1, which is the entryname of the object being deleted. EXAMPLE.KSDS1 is a key-sequenced VSAM cluster.
- PURGE, which specifies that the cluster is to be deleted regardless of its retention period or date.
- ERASE, which specifies that the cluster's data component is to be overwritten with binary zeros, regardless of a NOERASE attribute that might have been specified when the cluster was defined or altered.
- CATALOG, which identifies the catalog that contains the cluster's entries, D27UCAT2. The catalog's master password is required, unless the entry's master password is specified with the entryname.

Deleting an Entry-Sequenced VSAM Cluster and All Empty Data Spaces on a Volume: Example 3

In this example, an entry-sequenced VSAM cluster is deleted. Next, all empty VSAM data spaces on volume VSER03 are deleted and the volume's volume entry is deleted. The volume is dynamically allocated. Volume VSER03 must be mounted as permanently RESIDENT or RESERVED.

```
//DELET3   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT2,DISP=OLD
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DELETE -
          EXAMPLE.ESDS2/DEPT26M -
          PURGE -
          CLUSTER
     DELETE -
          VSER03 -
          SPACE
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for the job: D27UCAT2.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The first DELETE command deletes the only VSAM cluster on volume VSER03. The cluster, EXAMPLE.ESDS2, is identified with a DD statement to ensure that its volume is mounted when the cluster is deleted. The DELETE command's parameters are:

- EXAMPLE.ESDS2, which is the entryname of the cluster to be deleted, and DEPT26M, which is the cluster's master password. VSAM assumes the cluster is cataloged in the D27UCAT2 user catalog.
- PURGE, which specifies that the cluster is to be deleted regardless of its retention period or date.
- CLUSTER, which specifies that the entryname EXAMPLE.ESDS2 identifies a VSAM cluster (that is, the entryname identifies a cluster entry.)

The second DELETE command deletes all empty VSAM data spaces on volume VSER03. The VSDSET1 DD statement ensured that volume VSER03 is mounted when its data spaces are deleted. The DELETE command's parameters are:

- VSER03, which identifies the volume that contains empty VSAM data spaces. VSAM assumes the volume is cataloged in the D27UCAT2 user catalog.
- SPACE, which specifies that the entryname identifies a volume entry. When a volume's data spaces are to be deleted, the SPACE parameter is required.

Deleting Two Key-Sequenced Clusters: Example 4

In this example, two key-sequenced clusters, MYDATA and ENTRY, are deleted. This example illustrates how more than one cataloged object is deleted with a single DELETE command.

```
//DELET4   JOB   ...
//JOBCAT   DD    DSNAME=D27UCAT1,DISP=SHR
//         DD    DSNAME=COPYUCAT,DISP=SHR
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DELETE -
          (MYDATA -
           ENTRY) -
          PURGE -
          CLUSTER
/*
```

The job control statements are:

- JOBCAT DD, which makes two catalogs available for the job: D27UCAT1 and COPYUCAT. Concatenated JOBCAT DD statements are used to identify both catalogs.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DELETE command deletes the key-sequenced clusters MYDATA and ENTRY. Its parameters are:

- MYDATA and ENTRY, which identify the objects to be deleted. MYDATA and ENTRY are the entrynames of two key-sequenced clusters. VSAM assumes that MYDATA and ENTRY are cataloged in either the D27UCAT1 or COPYUCAT user catalogs.
- PURGE, which specifies that the cluster is to be deleted regardless of its retention period or date.
- CLUSTER, which specifies that MYDATA and ENTRY are clusters (that is, the names identify cluster catalog records).

Deleting Generic-Named Entries: Example 5

In this example, each catalog entry with the name "GENERIC.*.ABLE" is deleted, where "*" is any 1 to 8 character simple name. The name "GENERIC.*.ABLE" is a generic name, and this example shows how all catalog entries with the same generic name are deleted.

```
//DELET5   JOB   ...
//JOBCAT   DD    DSNAME=D27UCAT1,DISP=OLD
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DELETE -
          GENERIC.*.ABLE -
          PURGE -
          CATALOG(D27UCAT1/MRPWD27)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DELETE command removes all entries (and their associated entries) with the generic name "GENERIC.*.ABLE" from catalog D27UCAT1. Its parameters are:

- GENERIC.*.ABLE, a generic name, identifies all catalog entries with the name GENERIC.*.ABLE, where "*" is any 1 to 8 character simple name.
- PURGE, which specifies that each entry is to be purged regardless of the retention period or date specified when it was defined.
- CATALOG, which identifies the catalog that contains the entries, D27UCAT1, and its master password, MRPWD27.

List a Generation Data Group's Entries, Then Delete the Group and Its Data Sets: Example 6

In this example, a generation data group, GDG01, and its associated (generation data set) entries are listed. Next, the only generation data set in the group is deleted. Finally, the generation data group base catalog entry is deleted.

```
//DELET6   JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     LISTCAT -
          ENTRIES(GDG01) -
          ALL
     DELETE -
          GDG01.G0001V00 -
          PURGE
     DELETE -
          GDG01 -
          GENERATIONDATAGROUP -
          PURGE
/*
```

The job control statement, SYSPRINT DD, which is required in all Access Method Services job steps, identifies the output device to which Access Method Services messages to the programmer are sent.

Because no catalog was specified in a CATALOG parameter or with a JOBCAT or STEPCAT DD statement, VSAM assumes all entries required for this command sequence are in the master catalog.

The LISTCAT command lists the generation data group GDG01 and its associated generation-data-set entries. Its parameters are:

- ENTRIES, which specifies that the entry GDG01 is to be listed. Since the entry GDG01 is a generation data group entry, its associated generation data set's (nonVSAM) entries are also listed. In addition, if one of the generation data sets has aliases, the alias entries associated with the generation data set's entry are listed.
- ALL, which specifies that all fields are to be listed.

The first DELETE command removes the nonVSAM data set entry for the only generation data set in the generation data group: GDG01.G0001V00. Its parameters are:

- GDG01.G0001V00, the entryname of the object being deleted. GDG01.G0001V00 identifies the only generation data set in the generation data group GDG01.
- PURGE, which specifies that the generation data set's retention period or date is to be ignored. If PURGE is not specified and the generation data set's retention period has not yet expired, VSAM wouldn't delete its entry.

The second DELETE command removes the generation data group base catalog entry from the catalog. Its parameters are:

- GDG01, the entryname of the object being deleted. GDG01 identifies the generation data group base entry.
- GENERATIONDATAGROUP, which specifies the type of entry to be deleted. VSAM verifies that GDG01 is a generation data group entry, then deletes it. If GDG01 incorrectly specified another type of entry by mistake, VSAM would not delete the entry, but would note the discrepancy with a message to the programmer.
- PURGE, which specifies that the generation data group's retention period or date is to be ignored. If PURGE is not specified and the generation data group's retention period has not yet expired, VSAM wouldn't delete its entry.

Deleting a NonVSAM Data Set's Entry: Example 7

In this example, a nonVSAM data set's entry (cataloged in a VSAM user catalog) is deleted. The SCRATCH parameter is implied (that is, it is the default).

```
//DELET7   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT1,DISP=SHR
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DELETE -
          EXAMPLE.NONVSAM -
          PURGE -
          CATALOG(D27UCAT1/MRPWD27)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The DELETE command deletes the nonVSAM data set EXAMPLE.NONVSAM. Because the catalog in which the entry resides is not password protected, the CATALOG parameter is not required. The DELETE command's parameters are:

- EXAMPLE.NONVSAM, which is the entryname of the object to be deleted.
- PURGE, which specifies that the nonVSAM data set's retention period or date is to be ignored. If PURGE is not specified and the nonVSAM data set's retention period has not yet expired, VSAM wouldn't delete its entry.
- CATALOG, which identifies the catalog that contains the entries, D27UCAT1, and its master password, MRPWD27.

When the data set is deleted, its DSCB entry in the volume's VTOC is removed, dynamic allocation is used to allocate EXAMPLE.NONVSAM's volume.

Deleting a Member of a Partitioned (NonVSAM) Data Set: Example 8

In this example, the MEM1 member of partitioned data set EXAMPLE.NONVSAM2 is deleted. Next, the nonVSAM data set itself is deleted.

```
//DELET8   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT1,DISP=SHR
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DELETE -
          EXAMPLE.NONVSAM2(MEM1) -
          PURGE -
          CATALOG(D27UCAT1/MRPWD27)
     DELETE -
          EXAMPLE.NONVSAM2 -
          PURGE -
          CATALOG(D27UCAT1/MRPWD27)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT1.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The first DELETE command deletes one of the members of a partitioned data set, EXAMPLE.NONVSAM2(MEM1), from the user catalog D27UCAT1. Its parameters are:

- EXAMPLE.NONVSAM2(MEM1), which is the entryname of one of the members of the partitioned (nonVSAM) data set, EXAMPLE.NONVSAM2. The entryname identifies the object to be deleted.

- PURGE, which specifies that the nonVSAM data set's retention period or date is to be ignored. If PURGE is not specified and the nonVSAM data set's retention period has not yet expired, VSAM wouldn't delete its entry.
- CATALOG, which identifies the catalog that contains the entry, D27UCAT1, and its master password, MRPWD27.

The second DELETE command deletes all remaining members, as well as the data itself, of the partitioned nonVSAM data set EXAMPLE.NONVSAM2. Its parameters are:

- EXAMPLE.NONVSAM2, which is the entryname of the object to be deleted.
- PURGE, which specifies that the nonVSAM data set's retention period or date is to be ignored. If PURGE had not been specified and the nonVSAM data set's retention period has not yet expired, VSAM wouldn't delete its entry.
- CATALOG, which identifies the catalog that contains the entry, D27UCAT1, and its master password, MRPWD27.

In both parts of this example, the DSCB entry in the volume's VTOC is removed. Dynamic allocation is used to allocate the data set's volume.

Deleting a Page Space: Example 9

In this example, page space SYS1.PAGE2 is deleted.

```
//DELET3   JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DELETE -
          SYS1.PAGE2 -
          PURGE -
          PAGESPACE
/*
```

The job control statement SYSPRINT DD, which is required in all Access Method Services job steps, identifies the output device to which Access Method Services messages to the programmer are sent.

The DELETE command removes a page space entry, SYS1.PAGE2, from the master catalog. Its parameters are:

- SYS1.PAGE2, the entryname of the object to be deleted. SYS1.PAGE2 identifies a page space entry.
- PURGE, which specifies that the page space entry is to be deleted regardless of the retention period or date specified when it was defined.
- PAGESPACE, which specifies the type of entry to be deleted. VSAM verifies that SYS1.PAGE2 is a page space entry, then deletes it. If SYS1.PAGE2 incorrectly identified another type of entry, VSAM would not delete it, but would send an error message to the programmer.

Deleting All Empty VSAM Data Spaces on a Volume: Example 10

In this example, all of the empty data spaces on volume VSER05 are deleted. Because all VSAM data spaces on the volume are empty, the volume's catalog entry is also deleted. The volume is dynamically allocated. The volume, VSER05, must be mounted as permanently RESIDENT or RESERVED.

```
//DELET10  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DELETE -
          VSER05 -
          SPACE -
          CATALOG( AMASTCAT/MRCATPW2 )
/*
```

The DELETE command examines each data space cataloged in the volume's entry and, if the data space is empty, deletes the data space. The DELETE command deletes the volume entry when there are no data spaces cataloged in it and when there are no data sets identified that specify the volume as a candidate volume. Access Method Services also updates the volume's VTOC to reflect the deleted data spaces. The DELETE command's parameters are:

- VSER05, which identifies the volume with its serial number.
- SPACE, which specifies that a volume entry is to be modified or deleted. When a volume's data space are to be deleted the SPACE parameter is required.
- CATALOG, which identifies the catalog that owns the volume, AMASTCAT. The catalog's update (or higher level) password is required.

Deleting a User Catalog: Example 11

In this example, a user catalog is deleted. A user catalog can be deleted when it is empty—that is, when there are no objects except the catalog's volume cataloged in it. If the catalog is not empty, it cannot be deleted.

```
//DELET11  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DELETE -
          D27UCAT1/MRPWD27
          PURGE -
          USERCATALOG
/*
```

The DELETE command deletes the catalog. The volume that contained the catalog is now available for ownership by another VSAM catalog. The DELETE command also deletes the catalog's user catalog connector entry and alias entries, if any, in the master catalog. The DELETE command's parameters are:

- D27UCAT1, the name of the user catalog. The catalog's update (or higher level) password is required.
- PURGE, which specifies that the user catalog's retention period or date is to be ignored. If PURGE is not specified and the catalog's retention period has not yet expired, it won't be deleted.

- USERCATALOG, which identifies D27UCAT1 as a user catalog.
- CATALOG, which specifies the catalog (itself) in which the object to be deleted is cataloged. The catalog's update (or higher level) password is required.

MOVING ENTRIES

The EXPORT and IMPORT commands allow you to create backup copies, to transport user catalogs and clusters from one system to another, and to prevent (and subsequently, with the IMPORT command, to allow) anyone usage of a user catalog.

When a user catalog is exported, its catalog connector entry is removed from the master catalog. When the user catalog is subsequently imported to a new system, an entry is created for it in the new system's master-catalog volume and the user catalog is physically transported to the new system. When a user catalog is transported, it is not copied; the user catalog remains on its original volume in its original form. When a user catalog is disconnected but not exported to another system, its catalog connector entry is removed from the master catalog. The user catalog and its cataloged objects are unavailable to any user until the user catalog is connected (with the IMPORT CONNECT command) to a master catalog.

You don't have to issue the EXPORT command to allow a user catalog to be moved to another system. If the catalog is on a portable volume, you can physically move the volume to another system. You can also build a copy of the catalog for use on another system (see "Copying a Catalog"). You must use an IMPORT CONNECT command to build a catalog connector entry for the user catalog in the new system's master catalog. When you issue the EXPORT DISCONNECT command, VSAM removes the user catalog's connector entry and any alias entries in the master catalog that point to the user catalog.

When a cluster is exported, its catalog entry is copied to a moveable volume along with the cluster's user records. The entries and components are subsequently copied into a new system.

Exportation of a cluster is either permanent or temporary. In permanent exportation, the catalog record is deleted and storage space is freed in the original system. In temporary exportation, both the sending and receiving systems retain a copy of the cluster, but the copy in the original system is marked to indicate that there is a copy elsewhere.

When a cluster is exported, attributes specified in the DEFINE command or in subsequent ALTER commands are moved with the cluster, with the following exceptions:

- Volume serial numbers are moved only when the entry is to be kept in the original system.
- Catalog name.

Some attributes are moved with a cluster in addition to the attributes that can be specified in a DEFINE command. The additional attributes are:

- Control-area size.
- System time-stamp value.

The portable copy of a cluster is a variable-blocked, sequential data set.

Exporting an Entry

The EXPORT command is used in conjunction with the IMPORT command to move clusters and user catalogs from one system to another, to provide backup copies of clusters, and to sever the relationship between a user catalog and the master catalog. The EXPORT command cannot be used to move or provide a backup copy of a master catalog, a nonVSAM data set, data space, generation data group, or page space.

If a user catalog is to be exported, the catalog is merely disconnected from the master catalog. The user catalog can then be physically (or logically) moved to another system. Even if the catalog is not moved to another system, it is temporarily unavailable to the system's users and its aliases are deleted. The person who is responsible for the catalog can issue IMPORT CONNECT to make the catalog available again.

EXPORT Command

The format of the EXPORT command is:

EXPORT	*entryname* [*/ password*] {**DISCONNECT** | **INFILE(***dname***)** ъ [**OUTFILE(***dname***)** | **OUTDATASET(***entryname***)**]} [**TEMPORARY** | **PERMANENT**] [**INHIBITSHOURCE** | **NOINHIBITSOURCE**] [**INHIBITTARGET** | **NOINHIBITTARGET**] [**ERASE** | **NOERASE**] [**PURGE** | **NOPURGE**]

where:

EXPORT
specifies that a cluster or user catalog is to be moved from the system in which the command is executed. EXPORT can be abbreviated EXP.

entryname [*/ password*]
is a required parameter that names the cluster or user catalog to be exported. If you are exporting a user catalog, you must supply the update or higher level password of the master catalog. If you are exporting a cluster, you must supply the cluster's master password.

DISCONNECT
specifies that a user catalog is to be exported. The entry for the user catalog will be deleted from the master catalog. DISCONNECT is a required parameter if EXPORT is issued for a user catalog. In VS2, all of the user catalog's aliases are also deleted from the master catalog. The volume that contains the user catalog must be physically moved to the system to which the catalog will be imported. If DISCONNECT is coded, the entry name and the master catalog's update or higher-level password, and no other parameters, must be coded with it. To make a user catalog available in other systems and in the original system, code the IMPORT command to import the user catalog to each system to which it is to be available, but do not EXPORT the user catalog. DISCONNECT can be abbreviated DCON.

INFILE(*dname* **)**
specifies the name of the DD statement that identifies the location of the cluster to be exported. This parameter cannot be used when a catalog is to be exported.

You must identify the entry's catalog with a JOBCAT or STEPCAT DD statement, unless:

- the entry is the master catalog, or
- the first simple name in the entry's qualified dsname identifies a catalog (that is, the first simple name is the catalog's entryname or alias).

If INFILE is not specified and a cluster is to be exported, the entryname is dynamically allocated. INFILE can be abbreviated IFILE.

OUTFILE(*dname* **)**
specifies the name of the DD statement that identifies the output data set to be created as a result of the EXPORT command. The data-set characteristics of the output data set that is to contain the cluster to be exported should not be specified. The desired characteristics—physical sequential organization; variable, spanned, blocked record format; and a record length of the greater of 256 or the maximum record length plus 4—are established by the EXPORT command. This parameter cannot be specified when a catalog is to be exported.

You must identify the entry's catalog with a JOBCAT or STEPCAT DD statement, unless:

- the entry is the master catalog, or
- the first simple name in the entry's qualified dsname identifies a catalog (that is, the first simple name is the catalog's entryname or alias).

OUTFILE can be abbreviated OFILE.

OUTDATASET(*entryname* **)**
specifies the name of the data set that is to receive the data that is to be exported. If OUTDATASET is specified, the entryname is dynamically allocated. This parameter cannot be specified when a catalog is to be exported. OUTDATASET can be abbreviated ODS.

TEMPORARY | PERMANENT
specifies whether the cluster to be exported is to be deleted from the original system. This parameter cannot be used when the entry to be exported is a user catalog. If neither TEMPORARY nor PERMANENT is coded, PERMANENT is the default.

TEMPORARY
specifies that the cluster is not to be deleted from the original system. If the entry is moved to another system, volume information must be specified. The cluster in the original system is marked as temporary to indicate that another copy exists and that the original copy can be replaced. To replace the original copy, a portable copy created by an EXPORT command must be imported to the original system. The original copy is deleted, the new cluster is defined, and the data from the portable copy is copied into the newly defined cluster. TEMPORARY can be abbreviated TEMP.

PERMANENT
specifies that the cluster is to be deleted from the original system. Volume information must be specified to subsequently import the cluster. The storage space used by the cluster is freed. If a cluster is to be deleted from the orignal system and the cluster's retention period has not expired, PURGE must be coded. PERMANENT can be abbreviated PERM.

INHIBITSOURCE | NOINHIBITSOURCE
specifies whether the original cluster can be accessed for any operation other than retrieval. This parameter is not applicable if a user catalog is to be exported. This specification can later be altered through the ALTER command.

INHIBITSOURCE
specifies that the cluster in the original system cannot be accessed for any operation other than retrieval. This parameter cannot be used when a user catalog is to be exported. When INHIBITSOURCE is coded, TEMPORARY must also be coded. INHIBITSOURCE can be abbreviated INHS.

NOINHIBITSOURCE
specifies that the cluster in the original system can be accessed for any kind of operation. If neither INHIBITSOURCE nor NOINHIBITSOURCE is coded, NOINHIBITSOURCE is the default. NOINHIBITSOURCE can be abbreviated NINHS.

INHIBITTARGET | NOINHIBITTARGET
specifies whether the copy's records can be accessed for any operation other than retrieval after it has been imported to another system. This parameter cannot be used when a user catalog is to be exported. This specification can be altered through the ALTER command.

INHIBITTARGET
specifies that the exported copy cannot be accessed for any operation other than retrieval after it has been imported into another system. INHIBITTARGET can be abbreviated INHT.

NOINHIBITTARGET
specifies that the exported copy can be accessed for any type of operation after it has been imported into another system. If neither INHIBITTARGET nor NOINHIBITTARGET is coded, NOINHIBITTARGET is the default. NOINHIBITTARGET can be abbreviated NINHT.

ERASE | NOERASE
specifies whether the data component of the cluster to be exported is to be erased—overwritten with binary zeros. This parameter overrides whatever was specified when the cluster was defined or last altered. This parameter can be specified only if the cluster is to be permanently exported, that is, deleted from the original system. This parameter cannot be used when a user catalog is to be exported. ERASE and NOERASE can be abbreviated ERAS and NERAS, respectively.

PURGE | NOPURGE
specifies whether the cluster to be exported is to be deleted from the original system regardless of the retention period, specified in a TO or FROM parameter when the cluster was defined. This parameter can be specified only if the cluster is to be permanently exported, that is, deleted from the original system. This parameter cannot be specified when a user catalog is to be exported.

PURGE
specifies that the cluster is to be deleted even if the retention period has not expired. PURGE can be abbreviated PRG.

NOPURGE
specifies that the cluster is not to be deleted if the retention period has not expired. If neither PURGE nor NOPURGE is coded, NOPURGE is the default. NOPURGE can be abbreviated NPRG.

Note: When an entry is exported, the statistics kept in the catalog entry are lost, that is, the statistics are not available when the entry is subsequently imported.

EXPORT Examples

Exporting a User Catalog: Example 1

In this example, the user catalog D27UCAT1 is exported—that is, it is disconnected from the system. Its cataloged objects are no longer available to users of the system.

```
//EXPORT1  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     EXPORT -
          D27UCAT1/MRPWD27 -
          DISCONNECT
/*
```

The job control statement, SYSPRINT DD, which is required in all Access Method Services job steps, identifies the output device to which Access Method Services messages to the programmer are sent.

The EXPORT command removes the user catalog connector entry for D27UCAT1 from the master catalog. Also, alias entries for D27UCAT1 are removed. The catalog becomes unavailable to system users until the system programmer reconnects it to the system, using an IMPORT CONNECT command. The EXPORT command's parameters are:

- D27UCAT1, which identifies the object to be exported. When a user catalog is exported, the master catalog's master password is required. (See IMPORT example 1, "Import a User Catalog.")
- DISCONNECT, which identifies the exported object as a user catalog. When a user catalog is exported, DISCONNECT is required.

Exporting a Key-Sequenced VSAM Cluster: Example 2

In this example, a key-sequenced cluster, EXAMPLE.KSDS1, is exported from a user catalog, D27UCAT2. The cluster is copied to a portable file, TAPE2, and its catalog entries are modified to prevent the cluster's data records from being updated, added to, or erased.

```
//EXPORT2  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//STEPCAT  DD     DSNAME=D27UCAT2,DISP=SHR
//RECEIVE  DD     DSNAME=TAPE2,UNIT=(2400-3,,DEFER),
//                DISP=OLD
//          VOL=SER=003030,DCB=(DEN=3),LABEL=(1,SL)
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     EXPORT -
          EXAMPLE.KSDS1 -
          OUTFILE(RECEIVE) -
          TEMPORARY -
          INHIBITSOURCE
/*
```

The job control statements are:

- STEPCAT DD, which makes a catalog available for this job step: D27UCAT2.
- RECEIVE DD, which describes the portable file, a magnetic tape file, that is to receive a copy of the cluster's records.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The EXPORT command copies the key-sequenced cluster, EXAMPLE.KSDS1, to a portable file, TAPE2 and modifies the cluster's catalog entries (its cluster, data, and index entries). The EXPORT command's parameters are:

- EXAMPLE.KSDS1, which identifies the cluster to be exported. (Because DISCONNECT was not specified, Access Method Services assumes that EXAMPLE.KSDS1 is a cluster.) Because the cluster is not password protected, no password is provided with the cluster's entryname. EXAMPLE.KSDS1 is dynamically allocated to the job step.
- OUTFILE, which points to the RECEIVE DD statement. The RECEIVE DD statement describes the portable file, TAPE2, that is to contain a copy of the cluster.
- TEMPORARY, which specifies that the cluster is not to be deleted. The cluster's catalog entry is marked "temporary" to indicate that another copy of the cluster exists and that the original copy can be replaced (see the IMPORT example 2, "Import a Key-Sequenced Cluster.")
- INHIBITSOURCE, which specifies the cluster (that is, the copy of it that remains in the original system, as a result of TEMPORARY) cannot be modified. User programs are allowed only to read the cluster's records.

Exporting an Entry-Sequenced VSAM Cluster: Example 3

In this example, an entry-sequenced cluster is exported to a portable file and then it is deleted from the system.

```
//EXPORT3  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//STEPCAT  DD     DSNAME=D27UCAT2,DISP=OLD
//SOURCE   DD     DSNAME=EXAMPLE.ESDS1,DISP=OLD
//RECEIVE  DD     DSNAME=TAPE1,UNIT=(2400,,DEFER),DISP=NEW,
//         VOL=SER=001147,LABEL=(1,SL)
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     EXPORT -
          EXAMPLE.ESDS1/DEPT26M -
          INFILE(SOURCE) -
          OUTFILE(RECEIVE) -
          PURGE
/*
```

The job control statements are:

- STEPCAT DD, which makes a catalog available for this job step: D27UCAT2.
- SOURCE DD, which identifies the entry-sequenced cluster, EXAMPLE.ESDS1, that is to be exported.
- RECEIVE DD, which describes the portable file, TAPE1, that is to contain a copy of the exported entry-sequenced cluster.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The EXPORT command copies the entry-sequenced cluster, EXAMPLE.ESDS1, to a portable file, TAPE1. The cluster is deleted from the system after it is copied into the portable file. The EXPORT command's parameters are:

- EXAMPLE.ESDS1, which identifies the entry-sequenced cluster to be exported, and which specifies the cluster's master password, DEPT26M. (Because DISCONNECT was not specified, Access Method Services assumes that EXAMPLE.ESDS1 is a cluster.) When a cluster is exported, its master password is required if it is password protected.
- INFILE, which points to the SOURCE DD statement. The SOURCE DD statement identifies EXAMPLE.ESDS1, the cluster to be exported.
- OUTFILE, which points to the RECEIVE DD statement. The RECEIVE DD statement describes the portable data set, TAPE1, that is to receive a copy of the cluster.
- PURGE, which allows the cluster to be deleted regardless of its retention period or date.

Because TEMPORARY is not specified, Access Method Services assumes the cluster is to be deleted once TAPE1 contains a copy of it.

Because INHIBITUPDATE is not specified, Access Method Services assumes the cluster can be updated (by users of the other system) when it is imported to another system.

Importing an Entry

The IMPORT command is used in conjunction with the EXPORT command to move clusters and user catalogs from one system to another. When a cluster is imported, it is automatically reorganized. In the process of moving an entry, you may choose to change some of its attributes.

When a user catalog is moved into a system, you may connect it to the master catalog in the new system, thereby making the user catalog available in the new system. The user catalog might have been disconnected to make it temporarily unavailable to users even though its volume isn't moved. The person who is responsible for the catalog can issue IMPORT CONNECT to make it available again.

A cluster or user catalog cannot be moved into a system if its name or the names of any of its components is the same as any name that already exists in the receiving catalog. The only exception is when the existing cluster entry is temporary (that is, TEMPORARY was coded when the cluster was exported). A cluster's temporary entry is deleted when it is imported; a new entry is built and imported in place of the temporary entry. If same-named entries exist and the entry in the receiving catalog is not marked as temporary, the command is terminated.

The first record of the copy to be imported causes an entry to be defined in the receiving system. A check is made for a duplicate name; if a duplicate name exists and is marked temporary, the duplicated name is deleted from the receiving system so that the new entry can be defined. The deletion takes place even if the request for import cannot subsequently be completed.

Note: If you do not want the cluster reorganized, use OS/VS IEHDASDR. See *OS/VS Utilities* for a description of the IEHDASDR program.

IMPORT Command

The format of the IMPORT command is:

IMPORT	**[CONNECT]** **[{INFILE(***dname***) | INDATASET(***entryname***)}** ƀ **{OUTFILE(***dname***) | OUTDATASET(***entryname***)}]** **[CATALOG(***catname* **[/***password***])]** **[ERASE | NOERASE]** **[PURGE | NOPURGE]** **[OBJECTS((***name* **[ƀNEWNAME(***newname***)]** **[ƀVOLUMES(***volser* **[ƀ** *volser* ....])]** **[ƀFILE(***dname***)]** **[ƀKEYRANGES((***lowkey* **ƀ** ***highkey***)** **[ƀ(***lowkey* **ƀ** ***highkey***)....])]** **[ƀORDERED | UNORDERED]** **[ƀDEVICETYPE(***devtype***)])** **[ƀ(***name* ...)])]**

where:

IMPORT
specifies that a cluster or user catalog entry is to be moved into the system in which this command is executed. IMPORT can be abbreviated IMP.

CONNECT
specifies that the entry to be imported is a user catalog. The user catalog is connected to the master catalog in the receiving system. If CONNECT is coded, OBJECTS must also be coded to name the catalog and to identify the volume serial number and device type of the volume that contains the user catalog. CONNECT can be coded only when a user catalog is to be imported. CONNECT can be abbreviated CON.

INFILE(*dname* **)**
specifies the name of a DD statement that identifies the portable copy of the cluster to be imported. If a non-labeled tape contains the copy, the following DCB parameters must be specified on the referenced DD statement: BLKSIZE must equal 2048; LRECL must be the larger of 260 or the maximum record size plus 4; and RECFM must be VBS. INFILE cannot be used to identify a catalog to be imported. INFILE can be abbreviated IFILE.

INDATASET(*entryname* **)**
specifies the name of the entry to be imported. This parameter cannot be specified when a catalog is to be imported.

You must identify the entry's catalog with a JOBCAT or STEPCAT DD statement, unless:

- the entry is the master catalog, or
- the first simple name in the entry's qualified dsname identifies a catalog (that is, the first simple name is the catalog's entryname or alias).

If INDATASET is specified, the entryname is dynamically allocated. The entryname must be cataloged in a catalog that is accessible by the system into which the entry is to be imported. INDATASET can be abbreviated IDS.

OUTFILE(*dname* **)**
specifies the name of a DD statement that identifies the volume that is to receive the cluster that is to be imported. You must use concatenated DD statements when:

- a key-sequenced cluster is being imported, and
- its data and index components are on different device types, and
- either UNIQUE or ERASE was specified for the cluster (with a previous DEFINE or ALTER command.)

The first DD statement specifies the name of the cluster as the DSNAME, the volume serial numbers and device types of the data component, and AMP='AMORG'. The second DD statement specifies the name of the index component as the DSNAME, the volume serial numbers and device types of the index component, and AMP='AMORG'. If NEWNAME is specified, the data-set name on the DD statement must be the same as the new name. The volume that is to receive the imported cluster must be owned by a VSAM catalog. OUTFILE is not applicable to user catalogs. OUTFILE can be abbreviated OFILE.

OUTDATASET(*entryname*)
specifies the name of the data set that is to receive the data that is to be imported. This parameter cannot be specified when a catalog is to be imported.

You must identify the entry's catalog with a JOBCAT or STEPCAT DD statement, unless:

- the entry is the master catalog, or
- the first simple name in the entry's qualified dsname identifies a catalog (that is, the first simple name is the catalog's entryname or alias).

If OUTDATASET is specified, the entryname is dynamically allocated. OUTDATASET can be abbreviated ODS.

CATALOG(*catname*[/ *password*])
specifies the name of the catalog in which the imported entry is to be defined. This parameter is required when the catalog is password protected or when you want to direct the entry for the imported data set to a particular catalog. CATALOG can be abbreviated CAT.

catname
is the name of the catalog in which the entry to be imported is to be defined. If the specified catalog is not the master catalog, the catalog must be identified by a JOBCAT or STEPCAT DD statement. If you are importing a user catalog, the specified catalog must be the master catalog.

password
specifies the update or higher level password of the catalog in which the imported entry is to be defined.

ERASE | NOERASE
specifies whether the data component of the cluster that was exported with the TEMPORARY option is to be erased, that is, overwritten with binary zeros. This parameter can be used only when you are importing a cluster into a system from which it was previously exported with the TEMPORARY option. This parameter overrides whatever was specified when the cluster was defined or last altered. This parameter cannot be used when importing a user catalog. ERASE and NOERASE can be abbreviated ERAS and NERAS, respectively.

PURGE | NOPURGE
specifies whether the original cluster is to be deleted and replaced regardless of the retention time specified in the TO or FOR parameter. This parameter can be used only when you are importing a cluster into the original system from which it was exported with the TEMPORARY option. This parameter cannot be used when importing a user catalog. PURGE and NOPURGE can be abbreviated PRG and NPRG, respectively.

OBJECTS((*name* [ƀ **NEWNAME(***newname* **)]**
[ƀVOLUMES(*volser* [ƀ *volser*])]**
[ƀFILE(*dname* **)]**
[ƀKEYRANGES((*lowkey* ƀ*highkey* **)[ƀ(***lowkey* ƀ*highkey* **)...])]**
[ƀORDERED | UNORDERED]
[ƀDEVICETYPE(*devtype* **)])**
[ƀ(*name...*)])

specifies the attributes for the cluster or user catalog to be imported. If a user catalog is being imported, this parameter is required. Attributes may be specified for the cluster or user catalog and its components by repeating the parameter list.

name

specifies the name of the data component, index component, cluster, or user catalog whose attributes are being specified.

NEWNAME(*newname* **)**

specifies the new name of an imported cluster. You cannot change the name of an imported catalog. The new name can contain 1 to 44 alphanumeric, national (@, #, and $), and special (the hyphen and 12-0 overpunch) characters. Names that contain more than eight characters must be segmented by periods; one to eight characters may be specified between periods. The first character of the name or name segment must be either an alphabetic character or national character. If you are specifying a new name for an entry that was exported with the TEMPORARY option and the entry is being imported back into the original system, rename the cluster and each of its components. NEWNAME can be abbreviated NEWNM.

VOLUMES(*volser* [ƀ *volser...*] **)**

specifies the volumes on which the cluster is to reside or the volume on which the user catalog resides. If TEMPORARY was specified in the EXPORT command and the original volume from which the cluster or catalog was exported is to be the receiving volume, VOLUMES is not required. The volume serial number may contain one to six alphameric, national (@, #, and $), hyphens, and special (commas, semicolons, blanks, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, and equal signs) characters. The volume serial numbers must be enclosed in single quotation marks if they contain special characters. Single quotation marks within a volume serial number must be coded as two single quotation marks. If VOLUMES is not coded and the exported cluster was marked TEMPORARY, the original volume is used as the receiving volume. This parameter is required when a user catalog is to be imported; when importing a user catalog, specify only one volume. VOLUMES can be abbreviated VOL.

FILE(*dname* **)**

specifies the name of a DD statement that identifies the volumes allocated to the data and index components of a key-sequenced data set. This parameter is required when the components are defined as unique and when the data and index components reside on different device types. When components reside on different device types, FILE must be coded twice within the OBJECTS parameter, once in the parameter set for the index component and once in a second parameter set for the data component.

KEYRANGES((*lowkey* ђ *highkey*)
[ђ *lowkey* ђ *highkey* ...])
specifies portions of key-sequenced data to be placed on separate volumes. The data is divided, by key, among the volumes specified in VOLUMES. If a volume serial number was duplicated in VOLUMES, multiple key ranges of a cluster or component with the SUBALLOCATION attribute are placed on that volume. If the number of volumes is greater than the number of key ranges, the excess volumes are used for overflow records from any key range without consideration of range boundaries. If there are fewer volumes than key ranges, the excess key ranges are placed on the last volume specified. The maximum number of key-range pairs is 123. Key ranges may not overlap; gaps may exist within a specified set of ranges, but records cannot be inserted within a gap. Keys can contain 1 to 64 characters; if coded in hexadecimal, they may contain 1 to 128 hexadecimal characters. All EBCDIC characters are allowed. Keys consisting of characters must be enclosed in single quotation marks if they contain commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a key must be coded as two single quotation marks if the key is enclosed in single quotation marks. If the key is specified in hexadecimal, it must be preceded by X, be enclosed in single quotation marks. If both UNIQUE and KEYRANGES are specified for the same entry, each key range must reside on a separate volume. KEYRANGES may be abbreviated KRNG.

lowkey
specifies the low key of the key range. If lowkey is shorter than the actual keys, it will be padded with binary zeros.

highkey
specifies the high key of the key range. If highkey is shorter than the actual keys, it will be padded with binary ones.

ORDERED | UNORDERED
specifies whether volumes are to be used in the order in which they were listed in the VOLUMES parameter. If KEYRANGES is also specified, all of the records within the range specified by the first low-key/high-key pair are placed on the first volume specified in VOLUMES; all of the records within the second range are placed on the second volume; etc. If it is impossible to allocate volumes in the given order and ORDERED is specified, the command is terminated. If no value is specified, UNORDERED is the default. ORDERED and UNORDERED can be abbreviated ORD and UNORD, respectively.

DEVICETYPE(*devtype*)
specifies the device type of the volume that contains a user catalog that is to be imported. You can specify a device type for any direct-access device that is supported by your VS2 system. DEVICETYPE can be abbreviated DEVT.

By repeating the OBJECTS parameter set different volumes. Although the index and data components may reside on different device types, each volume of a multivolume component must be of the same type.

If the receiving volume is of a type different from that that originally contained the cluster, the job may be terminated because of allocation problems. Space allocation quantities are recorded in catalog entries in

cylinders and tracks even when RECORDS was specified in the DEFINE command. When a cluster is imported, the number of cylinders or tracks in the catalog entry is not modified, even though the cluster may be imported to reside on a device type other than that it was exported from. An attempt to import a cluster that previously resided on a 3330 may fail if it is imported to a 2314. Conversely, if a cluster is exported from a 2314 and imported to a 3330, more space is allocated than the cluster needs.

These space allocation problems can be avoided when moving clusters between systems or between catalogs by:

1. Using the REPRO command to copy the cluster to be exported to tape.
2. Using the DEFINE command to define a new entry for the cluster in the catalog to which the cluster is to be moved. Specify all the parameters used when the cluster was originally defined. If space was allocated in RECORDS, you may specify the same quantity; if it was allocated in TRACKS or CYLINDERS, you must adjust the quantity for the new device type. If an entry already exists on the catalog for this data set, you must delete that entry or use a different name in the DEFINE command.
3. Loading the data set into the newly defined cluster by issuing a REPRO command to copy it from the tape.

IMPORT Examples

Import a User Catalog: Example 1

In this example, a user catalog, D27UCAT1, is imported and connected to the new system's master catalog, AMASTCAT. This example reconnects the user catalog, D27UCAT1, that was disconnected with EXPORT example 1.

```
//IMPORT1  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     IMPORT -
          OBJECTS( -
                (D27UCAT1 -
                VOLUME(VSER02) -
                DEVICETYPE(2314) ) -
                ) -
          CONNECT -
          CATALOG(AMASTCAT/MRCATPW2)
/*
```

The job control statement, SYSPRINT DD, which is required in all Access Method Services job steps, identifies the output device to which Access Method Services messages to the programmer are sent.

The IMPORT command builds a user catalog connector entry that identifies the user catalog D27UCAT1 in the master catalog AMASTCAT. Its parameters are:

- OBJECTS, which is required when a user catalog is being imported. The subparameters of OBJECTS identify the user catalog, D27UCAT1, the user catalog's volume, VSER02, and the device type of the user catalog's volume, 2314.
- CONNECT, whcih specifies that the user catalog connector entry is to be built and put in the master catalog to connect the user catalog to the

master catalog. CONNECT is required when a user catalog is being imported.

- CATALOG, which identifies the master catalog, AMASTCAT, and specifies its update (or higher level) password, MRCATPW2.

Import a Key-Sequenced Cluster: Example 2

In this example, a key-sequenced cluster, EXAMPLE.KSDS1, that was previously exported, is imported (see the previous EXPORT example, "Exporting a Key-Sequenced VSAM Cluster.") The original copy of EXAMPLE.KSDS1 is replaced with the imported copy, TAPE2. Access Method Services finds and deletes the duplicate name, EXAMPLE.KSDS1, in the catalog D27UCAT2. (A duplicate name exists because TEMPORARY was specified when the cluster was exported.) Access Method Services then redefines EXAMPLE.KSDS1 using the catalog information from the portable file, TAPE2. Because the cluster's EXPORT command specified TEMPORARY, this example's IMPORT command doesn't specify volume information.

```
//IMPORT2  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//STEPCAT  DD     DSNAME=D27UCAT2,DISP=SHR
//SOURCE   DD     DSNAME=TAPE2,UNIT=(2400-3,,DEFER),
//         VOL=SER=003030,DISP=OLD,DCB=(DEN=3),
//         LABEL=(1,SL)
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(EXAMPLE.KSDS1) -
          CATALOG(D27UCAT2/MRPWD27)
/*
```

The job control statements are:

- STEPCAT DD, which makes a catalog available for this job step: D27UCAT2.
- SOURCE DD, which describes the portable data set, TAPE2. TAPE2 resides on a magnetic tape file, which will not be mounted by the operator until Access Method Services opens TAPE2 for processing.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The IMPORT command copies the portable data set, TAPE2, into the system and assigns it the name EXAMPLE.KSDS1. When TAPE2 is copied, Access Method Services reorganizes the data records so that deleted records are removed and control intervals and control areas contain the specified freespace percentages. The original copy of the cluster is deleted and replaced with the data records from the TAPE2 portable file. The IMPORT command's parameters are:

- INFILE, which points to the SOURCE DD statement. The SOURCE DD statement describes the portable file, TAPE2, to be imported.
- OUTDATASET, which identifies the receiving cluster, EXAMPLE.KSDS1. Because OUTDATASET, rather than OUTFILE, was specified, VSAM dynamically allocates the cluster.

- CATALOG, which identifies the catalog, D27UCAT2, in which the imported cluster is to be defined. The catalog's update (or higher level) password is required.

Import an Entry-Sequenced VSAM Cluster: Example 3

In this example, an entry-sequenced cluster, EXAMPLE.ESDS1, is imported from a portable file, TAPE1. This example is associated with IMPORT example 3, "Exporting an Entry-Sequenced VSAM Cluster."

```
//IMPORT3  JOB  ...
//STEP1    EXEC PGM=IDCAMS
//STEPCAT  DD   DSNAME=D27UCAT2,DISP=SHR
//SOURCE   DD   DSNAME=TAPE1,UNIT=(2400,,DEFER),DISP=OLD,
//          VOL=SER=001147,LABEL=(1,SL)
//RECEIVE  DD   DSNAME=EXAMPLE.ESDS2,DISP=OLD,UNIT=2314,
//          VOL=SER=VSER03,AMP='AMORG'
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     IMPORT -
          INFILE(SOURCE) -
          OUTFILE(RECEIVE) -
          OBJECTS( -
                (EXAMPLE.ESDS1 -
                   NEWNAME(EXAMPLE.ESDS2) -
                   VOLUMES(VSER03) ) -
                 ) -
          CATALOG(D27UCAT2/MRCATPW1)
/*
```

The job control statements are:

- STEPCAT DD, which makes a catalog available for this job step: D27UCAT2.
- SOURCE DD, which describes the portable file, TAPE1. TAPE1 resides on a magnetic tape file, which will not be mounted by the operator until Access Method Services opens TAPE1 for processing.
- RECEIVE DD, which describes the entry-sequenced cluster that is to contain TAPE1's records, EXAMPLE.ESDS2. DISP=OLD is specified because the IMPORT command causes space to be allocated for the cluster and its data component. AMP='AMORG' is required and identifies EXAMPLE.ESDS2 as a VSAM cluster. The DSNAME is the new name assigned to the cluster as a result of the IMPORT command.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The IMPORT command moves the contents of the portable file, TAPE1, into the system. When TAPE1 is moved, Access Method Services reorganizes the data records, so that deleted records are not copied. The IMPORT command's parameters are:

- INFILE, which points to the SOURCE DD statement. The SOURCE DD statement describes the data set to be imported, TAPE1.
- OUTFILE, which points to the RECEIVE DD statement. The RECEIVE DD statement describes the entry-sequenced VSAM cluster, EXAMPLE.ESDS2, that is to receive the imported copy of TAPE1.

- OBJECTS, which specifies some of the attributes for the object being imported:
  - EXAMPLE.ESDS1, which identifies the entry-sequenced cluster as it is currently named on TAPE1.
  - NEWNAME, which specifies that the cluster's entryname is to be changed to EXAMPLE.ESDS2.
  - VOLUMES, which identifies the volume on which the cluster is to reside.
- CATALOG, which identifies the catalog, D27UCAT2, that is to contain the cluster's catalog entry. The catalog's update (or higher level) password is required.

CONVERTING CATALOG ENTRIES

CNVCAT causes entries in an OS catalog to be converted to entries in the master catalog or a user catalog.

Figure 10 shows OS catalog entry types, abbreviations, descriptions, and the VSAM entry types into which OS catalog entries are converted.

Entry Name	Abbreviation	Description	VSAM Entry Type
Alias entry	AE	Contains an alternate name for the high-level qualifier of the data set name	Alias entry.
Control volume pointer entry	CVPE	Connects another control volume (CVOL) to this CVOL	Alias entry for a user catalog in master catalog.
Data-set entry pointer	DSPE	Contains the name and location of a data set	NonVSAM entry.
Generation index pointer entry	GIPE	Points to the lowest index for a generation data group	Generation data group.
Generation data set	G0000V00	Points to a generation data set	NonVSAM entry that is attached to a generation data group.

Figure 10. OS Catalog Entry Types and VSAM Equivalents

When you use the CNVTCAT command, you can cause all the entry types shown in Figure 10 to be converted or all but the control volume pointer entries (CVPEs) to be converted. CVPEs are converted to alias entries in the master catalog. If, for example, ABC was the name of a control volume (CVOL) that resided on volume 123456, ABC becomes the alias in the master catalog; you provide the name of the VSAM catalog for which ABC is an alias.

Because CNVTCAT creates aliases for CVPEs, converted OS catalog entries can be found without a STEPCAT or JOBCAT DD statement that identifies the VSAM catalog to be searched. If, for example, ABC.DEF, which was previously contained in an OS catalog, is to be found:

1. Master catalog is searched because no STEPCAT or JOBCAT is specified.
2. ABC is found in the master catalog; it is an alias that is related to a user catalog entry.
3. User catalog entry points to the user catalog.
4. ABC.DEF is found as a nonVSAM entry on the user catalog.

Figure 11 shows an OS catalog, SYSCTLG, that contains two CVPEs and shows the CVOLs that they point to. Before you can convert the CVPEs, you must be able to provide their volume serial numbers. To learn the volume serial numbers, list the OS catalog to be converted. If SYSCTLG, shown in the top half of Figure 11, is converted, aliases SYSA, and SYSB are created in the master catalog. In a separate conversion, the data sets cataloged in SYSA and SYSB are converted into entries in existing VSAM catalog. The bottom half of Figure 11 shows the master catalog and two user catalogs after SYSCTLG and SYSA and SYSB have been converted.

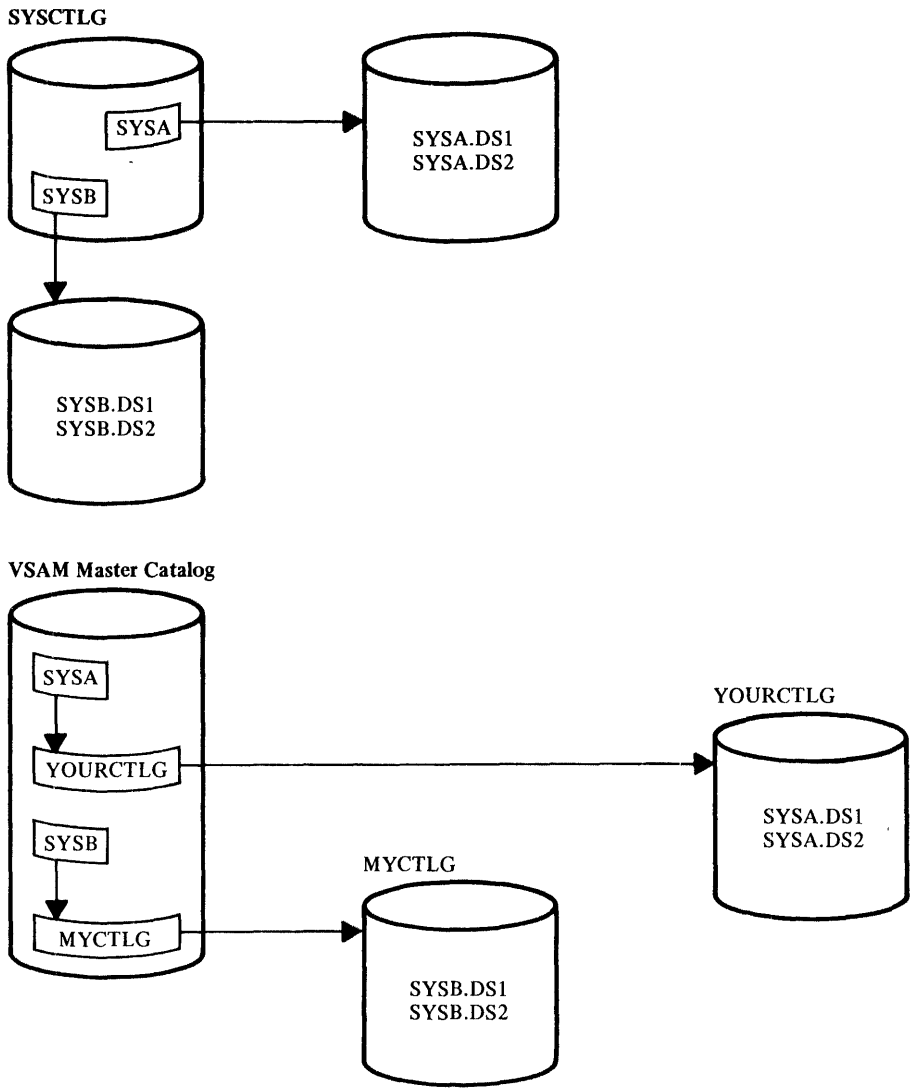


Figure 11. Converting Control Volume Entries

The amount of space required in the VSAM catalog depends upon the number and kind of records converted. Each OS catalog entry converts to a nonVSAM entry in a VS catalog. The exception is alias entries. One alias entry in an OS catalog can apply to many data-set names. When the alias is converted, one alias entry is created in the VSAM catalog for each data set that has the high-level qualifier to which the alias applies. For example, an alias, FAKE, exists for REAL. If REAL.A, REAL.B, and REAL.C are

converted, a total of six records are required in the VSAM catalog: one record is required for each of the three data sets and one record is required for each of three aliases, one per data set. See "Catalog Space Estimates" to help you estimate the VSAM catalog's size.

Note: Control volumes can be used without converting entries; control volumes cannot, however, be extended or maintained. See *OS/VS2 Using OS Catalog Management with the Master Catalog: CVOL Processor* for information about the use of control volumes.

CNVTCAT Command

The CNVTCAT command can be used to convert OS catalog entries into VSAM catalog entries.

The format of the CNVTCAT command is:

CNVTCAT	{**INFILE(***dname***)** | **INDATASET(***entryname***)**} [**CATALOG(***catname* [/ *password*])] [**MASTERCATALOG(***catname* [/ *password*])] [**CVOLEQUATES((***catname* (*volser* [ъ *volser*...])) [ъ(*catname*...)])] [**LIST** | **NOLIST**]

where:

INFILE(*dname* **)**
specifies the name of a DD statement that identifies the OS catalog that is to be converted. If INFILE is omitted, the entryname is dynamically allocated. INFILE can be abbreviated IFILE.

INDATASET(*entryname* **)**
specifies the control volume pointer (CVPE) name (not the data set name) of the OS catalog that is to be converted. If INDATASET is specified, the entryname is dynamically allocated. The OS system catalog entryname, SYSCTLG, cannot be specified. If entries in the system catalog are being converted, INFILE must be used to specify a DD statement that identifies the OS system catalog. INDATASET can be abbreviated IDS.

CATALOG(*catname* [/ *password*])
specifies the name of a catalog that is to receive the converted entries. If CATALOG is omitted, the entries are put into the catalog identified by a STEPCAT or JOBCAT DD statement. If neither the CATALOG parameter nor a JOBCAT or STEPCAT DD statement is specified, the entries are put in the master catalog. CATALOG can be abbreviated CAT.

password
specifies for a password-protected catalog the update or higher level password of the catalog that is to receive the converted entries.

MASTERCATALOG(*catname* [/ *password*])
specifies the name of the master catalog into which any aliases for user catalogs are to be placed. MASTERCATALOG is required when CVOLEQUATES is specified. MASTERCATALOG can be abbreviated MCAT and MRCAT.

password
specifies the update or higher level password of the master catalog if it is password protected.

[CVOLEQUATES((*catname* (*volser* [ђ *volser* ...]))
[(*catname* ...)])]
specifies the name of an existing VSAM catalog and the volume serial numbers of one or more control volumes (CVOLs) for which control volume pointer entries (CVPEs) are to be converted. If CVOLEQUATES is specified, MASTERCATALOG is required. CVOLEQUATES can be abbreviated CVEQU.

catname
specifies the name of an existing VSAM catalog. The catalog named contains or is to contain converted OS catalog entries that were cataloged in the CVOL pointed to by the CVPE being converted.

volser
specifies the volume serial number(s) of one or more CVOLS for which entries have been or are to be converted.

LIST | NOLIST
specifies whether entries are to be listed after they are converted. If neither LIST nor NOLIST is specified, LIST is the default. NOLIST can be abbreviated NLIST.

CNVTCAT Examples

The two CNVTCAT examples are related, and show how the entries of two catalogs in an OS system can be converted to entries in a VSAM catalog. The catalog on volume VSER09 contains control volume pointer entries (CVPEs) that point to the catalog on volume VSER08. Therefore, the two catalogs are chained together with VSER08's catalog at the end of the chain. When two or more OS catalogs are chained together, the catalog at the end of the chain should be converted first.

Convert an OS Catalog's Entries to Entries in a VSAM Catalog: Example 1

In this example, the entries in the catalog on volume VSER08 are converted to VSAM catalog entries and written into the USER11 catalog.

```
//CNVTCAT1 JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//OSCAT1   DD     VOL=SER=VSER08,DISP=OLD,DSN=SYSCATLG
//SYSIN    DD     *
     CNVTCAT -
        INFILE(OSCAT1) -
        CATALOG(USER11/MR)
/*
```

The job control statements are:

- OSCAT1 DD, which describes and allocates the OS catalog that is to be converted.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The CNVTCAT command converts the entries in the OS catalog on volume VSER08 to entries in a user catalog, USER11. The command's parameters are:

- INFILE, which points to the OSCAT1 DD statement. The OSCAT1 DD statement describes and allocates the OS catalog that is to be converted.
- CATALOG, which specifies the name of an existing VSAM catalog, the user catalog USER11. USER11 is to receive the converted entries. The user catalog's update (or higher level) password, MR, is required.

Convert an OS Catalog's Entries to Entries in a VSAM Catalog: Example 2

In this example, the entries in the OS catalog on volume VSER09 are converted to VSAM catalog entries and written into the USER11 catalog. Some of the entries in the OS catalog are control volume pointer entries, and point to the OS catalog on volume VSER08 (which was converted in the previous example).

```
//CNVTCAT2 JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//OSCAT2   DD     VOL=SER=VSER09,DISP=OLD,DSN=SYSCTLG
//SYSIN    DD     *
     CNVTCAT -
        INFILE(OSCAT2) -
        CATALOG(USER12/MR) -
        CVOLEQUATES( -
           (USER11(VSER08))   ) -
        NOLIST -
        MASTERCATALOG(AMASTCAT)
/*
```

The job control language statements are:

- OSCAT2, which describes and allocates the OS catalog that is to be converted.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The CNVTCAT command converts the entries in the OS catalog on volume VSER09 to entries in a user catalog, USER12. Because CVOLEQUATES is specified, an alias entry is built and put in the master catalog to relate the name VSER08 to the user catalog USER11. The command's parameters are:

- INFILE, which points to the OSCAT2 DD statement. The OSCAT2 DD statement describes and allocates the OS catalog that is to be converted.
- CATALOG, which specifies the name of an existing VSAM catalog, the user catalog USER12. USER12 is to receive the converted entries. The user catalog's update (or higher level) password, MR, is required.
- CVOLEQUATES, which specifies that control volume pointer entries (CVPEs) are to be converted to alias entries. Each CVPE in VSER09's catalog points to an entry in VSER08's catalog. Each new alias entry points to the USER11 user catalog connector entry in the master catalog. The alias entries are written into the master catalog.

- NOLIST, which specifies that the entries are not to be listed after they are converted.
- MASTERCATALOG, which identifies the master catalog. This parameter must be specified because CVOLEQUATES is also specified. If the master catalog is password protected, the catalog's update (or higher level) password must be specified.

VERIFYING END-OF-FILE

When a data set is closed, its end-of-file (EOD) and end-of-key-range (EOKR) information is used to upgrade EOD and EOKR information in the data set's cataloged information. If an OS/VS system failure occurs before the data set is closed (that is, before the user's program issues CLOSE or CLOSE(TYPE=T)), the data set's cataloged information is not upgraded. This means that the data set's cataloged information contains possibly obsolete EOD and EOKR information. The data set's real EOD and EOKR indicators are written in the data set, but are not shown in the data set's cataloged information. When the data set is subsequently opened and the user's program attempts to process records EOD or EOKR, a "no record found" error results on a read operation, and a write operation might write records over previously written records.

VERIFY Command

The VERIFY command is used to compare the end-of-data-set and end-of-key-range information as it is stored in a VSAM catalog with the true end-of-file and end-of-key-range. If the information in the catalog does not agree with the true end-of-file or end-of-key-range, the catalog information is corrected. The VERIFY command can be used following a system failure that caused a component opened for update processing to be improperly closed. Clusters (or their components) and catalogs can be verified.

The format of the VERIFY command is:

VERIFY	{**FILE(***dname* [*/password*]) | **DATASET(***entryname* [*/password*])}

where:

VERIFY
specifies that the end-of-data-set information is to be verified. VERIFY can be abbreviated VFY.

FILE(*dname* [*/ password*] **)**
specifies the name of a DD statement that identifies the cluster or component to be verified.

DATASET(*entryname* [*/ password*] **)**
specifies the name and password of the data set to be verified. If DATASET is coded, the receiving data set will be dynamically allocated. DATASET can be abbreviated DS.

password
is the control or master password of a password-protected cluster or component or the master password of a password-protected catalog.

Upgrading a Data Set's End-of-File Information

When a data set that was improperly closed (that is, closed as a result of system failure) is opened, the VSAM Open routines set a return code to indicate that the data set's cataloged information might not be accurate. The user can upgrade the EOD and EOKR information (so that it is accurate when the data set is next opened) by closing the data set[1] and issuing the VERIFY command:

```
//FIXEOD  EXEC  PGM=IDCAMS
//ACCNTS   DD   DSN=FAROUT,DISP=OLD
//SYSIN    DD   *
           LISTCAT   ENTRIES(FAROUT) -
                     CLUSTER -
                     ALL
           VERIFY    FILE(ACCNTS)
           LISTCAT   ENTRIES(FAROUT) -
                     CLUSTER -
                     ALL
/*
//
```

ACCNTS is a DD statement that describes the FAROUT data set.

The first LISTCAT command lists the data set's cataloged information, showing the data set's parameters as they were when the data set was last closed properly.

The VERIFY command updates the data set's cataloged information to show the data set's real EOD and EOKR values.

The second LISTCAT command lists the data set's cataloged information again. This time, the EOD and EOKR information show the point at which processing stopped due to system failure. This information should help the user determine how much of his data was added correctly before the system failed.

[1] When the data set is opened (first OPEN after system failure), VSAM Open sets a "data set improperly closed" return code. When the data set is closed properly, VSAM Close resets the "data set improperly closed" indicator but does not upgrade erroneous catalog information that resulted from the system failure. Subsequently, when the data set is next opened, its EOD and EOKR information might still be erroneous (until VERIFY is issued to correct it), but VSAM Open sets the "data set opened correctly" return code.

Interpreting LISTCAT Output Listings

See "Appendix B: Interpreting LISTCAT Output Listings" for details on the order in which catalog records are listed and the meanings of the listed fields.

VERIFY Example

Rectify the End-of-File and End-of-Key-Range Values: Example 1

In this example, the end-of-file value of MYDATA is checked and, if necessary, corrected so that the cataloged end-of-file value indicates the data set's actual end-of-file relative byte address (RBA).

```
//VERIF1   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT1,DISP=SHR
//STEP1    EXEC   PGM=IDCAMS
//FILE1    DD     DSNAME=MYDATA,DISP=OLD
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     VERIFY -
          FILE(FILE1)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT1.
- FILE1 DD, which identifies the cluster to be verified. The DISP=OLD parameter is required to ensure that the cluster is not updated by another user when it is being verified.

The VERIFY command first opens the cluster, then verifies that the cataloged end-of-file RBA (relative byte address) is the cluster's end-of-file. If not, Access Method Services finds the cluster's end-of-file indicator and rectifies the cataloged end-of-file RBA. The cluster is then closed.

The VERIFY command's FILE parameter points to the FILE1 DD statement. The FILE1 DD statement identifies the cluster, MYDATA. VSAM assumes that MYDATA is cataloged in either D27UCAT1 or the master catalog.

COPYING AND PRINTING

The REPRO and PRINT commands are used to copy and print VSAM and nonVSAM data sets, and to copy VSAM catalogs.

Copying Data Sets

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

- Copy a VSAM data set into another VSAM data set.
- Copy a sequential data set into another sequential data set.
- Convert a sequential or indexed-sequential data set into a VSAM data set.
- Convert a VSAM or indexed-sequential data set into a sequential data set.
- Copy a data set—other than a catalog—to reorganize it. This is an automatic feature over which you have no control.
- Merge two VSAM data sets.
- Copy a catalog for conversion from a volume of one device to another—for example, from a 2314 volume to a 3330 volume.
- Make a backup copy of a catalog.

Throughout the remainder of the REPRO discussion, all of these functions will be referred to as copying.

VSAM data sets used as either input or output must be cataloged. Sequential and indexed-sequential data sets need not be cataloged. For a key-sequenced data set, the records to be copied must be in ascending order when the records are initally loaded.

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

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

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

Figure 12 describes how the records from the input data set are added to the output data set when the output data set is an empty or non-empty entry-sequenced, key-sequenced, or sequential data set.

	Empty	Non-Empty
Entry-Sequenced/ Sequential	Creates new data set in sequential order.	Adds records in sequential order to the end of the data set.
Key-Sequenced	Creates new data set in key sequence and builds an index.	Merges records by key and updates the index. Records whose key duplicates a key in the output data set are lost.

Figure 12. Adding Records to Various Types of Output Data Sets

Note: If four recoverable errors are encountered in the copy operation, the command is terminated.

To use your own program to load a key-sequenced data set, first sort the records (or build them) in key sequence, and store them with sequential access (the PUT macro). If you have defined the data set with free space, sequential storage leaves the indicated free space in control intervals and control areas. See *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* to find out how to use the VSAM macros to write your own program to load records into a data set.

To use the REPRO command to load a key-sequenced data set, the records must be in key sequence. With an entry-sequenced data set, the records to be loaded can be in any order.

REPRO causes Access Method Services to retrieve records from a sequential, indexed-sequential, or VSAM data set and store them in VSAM format in key sequence or entry sequence, or store them in a sequential data set. When records are stored in key sequence, index entries are created and loaded into an index component as control intervals and control areas are filled up. Free space, as indicated in the data-set definition in the catalog, is left and records are stored on particular volumes according to key ranges, if indicated in the definition.

You can load all of the records in one job or load them in several jobs. In subsequent jobs VSAM continues to store records as before, extending the data set as required.

Loading Records into a Data Set

In a REPRO job, your VSAM data sets are already cataloged. JCL can be used to allocate the required data set(s) and indicate user catalog(s), if any, so that VSAM can look at the definition of a data set to find out what volume(s) it's stored on. A data set can be dynamically allocated if JCL is not used. See "JCL and Dynamic Allocation" in "Introduction" for additional information about dynamic allocation.

To name a data set, using JCL, code:

// *dname* **DD DISP=OLD,DSNAME=** *dsname*

Copying a Catalog

You can use the REPRO command to copy a VSAM user catalog for conversion from one volume to another. For example, you might convert from a 2314 to a 3330 for faster I/O and improved space allocation.

The catalog is copied into a newly defined, empty user catalog. Copying a catalog for conversion is *not the same* as making a backup copy of a catalog, which is described below under "Backing up a Catalog."

To use the REPRO command to copy a catalog, Access Method Services must be authorized. See *OS/VS2 System Programming Library: Supervisor* for information about the authorized program facility.

Copy-Catalog Preparation

In preparation for copying a catalog into a newly defined catalog, you should determine the amount of space to be allocated in the receiving catalog. Part of the process of determining how to allocate space for the receiving catalog includes reviewing how space is allocated in the catalog to be copied.

Two methods can be used to estimate the size of the receiving catalog:

- Use

  LISTCAT–
  CATALOG(*catname/password*)

  to list the source catalog's entries. Use the summary table at the end of the LISTCAT output to determine the number of each type of entry. Use the worksheet in the "Catalog Space Estimates" section to determine the number of records in the source catalog. Add 20 to the total number of records to determine the *minimum* number of records required for the receiving catalog.

- Use

  LISTCAT–
  ENTRIES (*catname/password*)–
  CATALOG(*catname/password*)–
  ALL

  to list the catalog entry (that is, the cataloged information that describes the catalog itself). Interpret the LISTCAT output to determine the high-allocated RBA value for the catalog's data and index components. Add the two RBA values and divide the sum by 512 to determine the number of records that are occupied by the source catalog. Add 20 to the number of records to determine the *minimum* number of records required for the receiving catalog.

The first method results in the smallest possible receiving catalog, since it doesn't account for any of the source catalog's free records (that is, records that have been used, then marked deleted and are available for reuse). However, the first method will probably require a secondary space allocation when an entry is added to it.

The second method results in a larger receiving catalog, since the high-allocated RBA values include free records in the source catalog. However, the receiving catalog's primary allocation quantity derived by this

method is usually large enough to contain additional entries in the receiving catalog.

Copy-Catalog Procedure

The following steps should be followed to copy a catalog:

1. Use the DEFINE command to define a catalog into which the source user catalog is to be copied. The receiving user catalog must be empty. That is, the receiving catalog cannot contain any entries other than the entries that describe the catalog and its data space.
2. Use the REPRO command to copy the source catalog to the receiving catalog. Use a concatenated JOBCAT or STEPCAT DD statement to describe and allocate both the source and receiving catalogs. The receiving catalog must not contain any other entries before the source catalog is copied.
3. Use the DELETE command to remove the source user catalog's entries, which appear in the receiving catalog as a result of the copy operation, from the receiving catalog.
4. Use the LISTCAT command to list the source user catalog's aliases:

   ```
   LISTCAT -
        CATALOG( master catname/password ) -
        ENTRIES( user catname ) -
        ALL
   ```

   The aliases are listed under the heading "ASSOCIATIONS."
5. Use the EXPORT command to disconnect the source user catalog from the master catalog.
6. Use the DEFINE ALIAS command to establish for the new user catalog each alias that belonged to the old catalog (see step 3).

The copy-catalog procedure should be performed with caution. Until the source user catalog is disconnected from the master catalog, two user catalogs are available for use.

At the completion of the copy-catalog procedure:

- The DSCB on the source-catalog volume is modified so that it no longer indicates that the volume contains a catalog.
- Space used by the source catalog is made available for suballocation.
- A volume entry is created in the receiving catalog for the source-catalog volume; the source-catalog volume is owned by the receiving catalog.

Backing up a Catalog

You can use the REPRO command to unload a VSAM catalog into a key-sequenced, entry-sequenced, or sequential (SAM) data set. If the catalog becomes inaccessible, you can use REPRO to reload the backup to replace it.

Using REPRO to unload or reload a password-protected catalog requires the catalog's master password.

The FROM and TO parameters are invalid for unload/reload.

Use a STEPCAT DD statement to identify the catalog that you are loading or reloading.

Do not allow the catalog to be updated during the operation. (REPRO does not itself prevent other processing of the catalog.)

Unloading a Catalog

A sequential, key-sequenced, or entry-sequenced backup copy of a catalog is inaccessible as a catalog. Because there is no advantage in having the backup on a direct-access volume, you can most conveniently use magnetic tape to copy a catalog in a sequential data set.

To unload a catalog into a key-sequenced or entry-sequenced data set, first define the data set in *another* catalog (for protection).

To continually have a recent backup copy available, you should unload a catalog periodically. With tape, you can easily alternate two or more volumes for several levels of backup.

Use LISTCAT before unloading a catalog. You can compare the listing with the one you obtain after reloading.

Reloading a Catalog

You can use REPRO to reload the backup copy into a catalog (the "target") with the same name, volume serial number, and device type as the original catalog. The target catalog can be a version of the original catalog (which might have been obtained by way of IEHDASDR RESTORE). Reloading the master catalog has special requirements, because a VS2 system must have a master catalog operational. (See the section "Backing up the Master Catalog" in the chapter "Data Security and Integrity" for information about the special requirements.)

To reload a user catalog, the target can also be a new user catalog that you defined after using EXPORT DISCONNECT to get rid of the original catalog. The newly defined catalog must be able to hold at least as many entries as the original catalog could hold at the time the backup copy was made.

Whether you reload a version of the original catalog or a newly defined catalog, reloading results in a catalog equivalent to the original one at the time the backup was made. Reloading *replaces* entries in the target catalog with entries of the same name in the backup. It *inserts* into the target entries that exist only in the backup. It *deletes* from the target entries that exist only in the target. During reloading, Access Method Services issues a maximum of one hundred messages to indicate entries that exist only in the target or only in the backup.

After you reload a catalog, use LISTCAT to list its contents. Run LISTCAT in a *separate step,* so that the catalog will be closed after it is reloaded (to update its self-defining information). Compare the listing with the one you obtained before unloading the original catalog to ensure that you have used the right backup.

See the section "Updating a Backup Catalog" in the chapter "Data Security and Integrity" for a discussion of making the contents of the backup catalog agree with the contents of the original catalog at the time it became inaccessible.

Optimizing the Performance of Unload/Reload

You can specify additional I/O buffers for unloading and reloading by using:

- The AMP parameter in the STEPCAT DD statement that identifies the catalog—AMP= 'BUFND=x, BUFNI=2', where x equals 2 times the number of 512-byte control intervals per track of the device used for the catalog.
- The AMP parameter in the DD statement that identifies a key-sequenced or entry-sequenced backup copy—AMP= 'BUFND=x, BUFNI=2', where x equals 2 times the number of 512-byte control intervals per track of the device used for the backup.
- The DCB parameter in the DD statement that identifies a sequential backup copy—DCB=BUFNO=x, where x equals either:
  - 2 times the number of 512-byte control intervals per track of the device used for the catalog (when the backup is on magnetic tape) or
  - 2 times the number of physical records per track of the device used for the backup (when the backup is on a direct-access volume).

  Block the records in a sequential backup data set. Some catalog records are 47 bytes long; the rest are 505 bytes long. Use DCB=RECFM=VB.

Example of Unload/Reload

The following example shows unloading a catalog into a sequential tape data set, exporting the catalog to get rid of it (the catalog), redefining it, and reloading it with the backup.

```
//JOB1         JOB
//STEP1        EXEC  PGM=IDCAMS,REGION=256K
//STEPCAT      DD    DSN=MINCAT,DISP=OLD
//SYSPRINT     DD    SYSOUT=A
//SYSABEND     DD    SYSOUT=A
//AMSDUMP      DD    SYSOUT=A
//CATDS        DD    DSN=MINCAT,DISP=OLD,AMP='BUFND=40,
//             BUFNI=3'
//TAPE         DD    DSN=CATURTP,UNIT=2400,VOL=SER=003535,
//             DISP=NEW,DCB=(RECFM=VB,LRECL=516,
//             BLKSIZE=5164,BUFNO=40),
//             LABEL=(1,SL)
//SYSIN        DD    *
     LISTCAT CATALOG (MINCAT) ALL
     REPRO-
         IFILE(CATDS) -
         OFILE(TAPE)
/*
//JOB2         JOB
//STEP1        EXEC  PGM=IDCAMS,REGION=256K
//SYSPRINT     DD    SYSOUT=A
//USERCAT      DD    UNIT=3330,VOL=SER=MVSPAG,DISP=OLD
//SYSIN        DD    *
     EXPORT MINCAT DISCONNECT
     DEFINE USERCATALOG (NAME (MINCAT) FILE (USERCAT) -
         VOLUMES (MVSPAG) CYL (28) DATA -
         (TRACKS (100 3)   ) -
         INDEX (TRACKS (5)   )
//STEP2        EXEC  PGM=ICDAMS,REGION=256K
//STEPCAT      DD    DSN=MINCAT,DISP=OLD
//SYSPRINT     DD    SYSOUT=A
//SYSABEND     DD    SYSOUT=A
//AMSDUMP      DD    SYSOUT=A
//CATDS        DD    DSN=MINCAT,DISP=OLD,AMP='BUFND=40,
//             BUFNI=3'
//TAPE         DD    DSN=CATURTP,UNIT=2400,VOL=SER=003535,
//             DCB=BUFNO=40,DISP=OLD
//SYSIN        DD    *
     REPRO-
         IFILE (TAPE) -
         OFILE (CATDS)
/*
//STEP3        EXEC  PGM=ICDAMS,REGION=256K
//STEPCAT      DD    DSN=MINCAT,DISP=OLD
//SYSPRINT     DD    SYSOUT=A
//SYSABEND     DD    SYSOUT=A
//AMSDUMP      DD    SYSOUT=A
//SYSIN        DD    *
     LISTCAT CATALOG (MINCAT) ALL
/*
```

REPRO Command

The format of the REPRO command is:

REPRO	{**INFILE(***dname* [*/password*] [ƀ**ENVIRONMENT(DUMMY)**]) | **INDATASET(***entryname* [*/password*] [ƀ**ENVIRONMENT(DUMMY)**])} {**OUTFILE(***dname* [*/password*]) | **OUTDATASET(***entryname* [*/password*])} [**FROMKEY(***key*) | **FROMADDRESS(***address*) | **SKIP(***count*)] [**TOKEY(***key*) | **TOADDRESS(***address*) | **COUNT(***count*)]

where:

INFILE(*dname* [*/ password*][ƀ**ENVIRONMENT(DUMMY)**]) \|
INDATASET(*entryname* [*/ password*][ƀ**ENVIRONMENT(DUMMY)**])
identifies the data set to be copied.

INFILE(*dname* [*/ password*])
specifies the name of the DD statement that identifies the data set to be copied. INFILE can be abbreviated IFILE.

INDATASET(*entryname* [*/ password*])
specifies the name of the entry to be copied. If INDATASET is specified, the entryname is dynamically allocated. INDATASET can be abbreviated IDS.

password
is the read or higher level password of the data to be copied. If the data is password protected, a password must be supplied. If a catalog is to be copied, the master password is required.

ENVIRONMENT (DUMMY)
specifies that dummy ISAM records are to be copied. Dummy records are records with hexadecimal 'FF' in the first byte. If you do not include this parameter, dummy records will be ignored during the copy operation. See *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* for further information. If ENVIRONMENT is not coded, dummy records are not copied. ENVIRONMENT and DUMMY can be abbreviated ENV and DUM, respectively.

OUTFILE(*dname* [*/ password*]) \|
OUTDATASET(*entryname* [*/ password*])
identifies the output data set. ISAM data sets cannot be specified as output data sets.

OUTFILE(*dname* [*/ password*])
specifies the name of a DD statement that identifies the output data set. OUTFILE can be abbreviated OFILE.

OUTDATASET(*entryname* [*/ password*])
specifies the name of the output data set. If OUTDATASET is specified, the entryname is dynamically allocated. OUTDATASET can be abbreviated ODS.

password
specifies the update or higher level password for a password-protected output data set.

FROMKEY(*key* **) | FROMADDRESS(** *address* **) | SKIP(** *count* **)**
specifies the location in the input data set from which copying is to start. You can use only one of the three possible choices. If no value is coded, the listing begins with the first logical record in the data set. The only parameter that can be coded for a SAM data set is SKIP. If you are copying a catalog, the entire catalog must be copied; FROMKEY, FROMADDRESS, or SKIP cannot be specified.

FROMKEY
specifies the key of the first record you want copied. You can specify generic keys—that is, keys shorter than that defined for the data set. If you specify generic keys, copying begins at the first record whose key matches that portion of the key you specified. (You cannot specify a key longer than that defined for the data set. If you do, the data set is not copied.) If the specified key is not found, the next higher key is used as the starting point for copying. FROMKEY can be coded only for key-sequenced or ISAM data sets; FROMKEY cannot be specified if a catalog is to be copied. The key must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a key must be coded as two single quotation marks; it may contain up to 255 hexadecimal characters. The address can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks. FROMKEY can be abbreviated FKEY.

FROMADDRESS
specifies the relative byte address (RBA) of the first record you want copied. The RBA value must be the beginning of a logical record. If you specify this parameter for key-sequenced data, the records will be copied in a physical sequential order instead of in a logical sequential order. FROMADDRESS can be coded only for VSAM data sets. If it is specified in hexadecimal or binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks. TOADDRESS can be abbreviated TADDR. FROMADDRESS can be abbreviated FADDR.

SKIP
specifies the number of logical records you want to skip before beginning to copy records. For example, if you want to copy beginning with record number 500, you specify SKIP(499). The *count* can be expressed in decimal, hexadecimal, or binary. If the amount is specified in hexadecimal or binary, it must be preceded by X or B, be enclosed in single quotation marks, and cannot be longer than one fullword.

TOKEY(*key* **) | TOADDRESS(** *address* **) | COUNT(** *count* **)**
specifies the location in the data set being copied at which copying is to stop. You can use only one of the three choices. If no value is coded, the copying ends with the logical end of the data set or component. The only parameter that can be specified for a sequential data set is COUNT. If you are copying a catalog, the entire catalog must be copied; TOKEY, TOADDRESS, or COUNT cannot be specified. The location at which the copying is to end must follow the location at which it is to begin.

TOKEY
specifies the key of the last record you want copied. You can specify generic keys—that is, keys shorter than that defined for the data set. If you specify generic keys, copying stops after the last record is copied whose key matches that portion of the key you specified. (You cannot specify a key longer than that defined for the data set. If you do, the data set is not copied.) If the specified key is not found, the next lower key is used as the stopping point for copying. TOKEY cannot be specified as the ending location if FROMADDRESS was specified as the starting location. TOKEY can be used only with indexed-sequential data sets or with key-sequenced data when the cluster name is specified; TOKEY cannot be specified if a catalog is to be copied. The key must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a key must be coded as two single quotation marks; it may contain up to 255 hexadecimal characters.

TOADDRESS
specifies the relative byte address (RBA) of the last record you want copied. Unlike FROMADDRESS, the RBA value does not need to be the beginning of a logical record. The entire record containing the specified RBA is copied. If you specify this parameter for a key-sequenced data set, the listing will be in a physical sequential order instead of in a logical sequential order. TOADDRESS cannot be specified as the ending location if FROMKEY was specified as the starting address. TOADDRESS can be used only with VSAM data sets or components. The address can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks. TOADDRESS can be abbreviated TADDR.

COUNT
specifies the number of logical records you want copied. The *count* can be expressed in decimal, hexadecimal, or binary. (0-9). If the amount is specified in hexadecimal or binary, it must be preceded by X or B, be enclosed in single quotation marks, and cannot be longer than one fullword.

REPRO Examples

Copy Records Into a VSAM Cluster: Example 1

In this example, records from an indexed-sequential data set, ISAMDSET, are copied into an key-sequenced VSAM cluster, EXAMPLE.KSDS1.

```
//REPRO1 JOB     ...
//JOBCAT   DD    DSNAME=D27UCAT2,DISP=SHR
//STEP1  EXEC    PGM=IDCAMS
//INDSET1  DD    DSNAME=ISAMDSET,DISP=OLD,
//                DCB=(DSORG=IS,BUFNO=6)
//VSDSET1  DD    DSNAME=EXAMPLE.KSDS1,DISP=OLD
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     REPRO -
          INFILE(INDSET1) -
          OUTFILE(VSDSET1)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT2.
- INDSET1 DD, which describes the indexed-sequential data set, ISAMDSET, that contains the input records. The BUFNO parameter specifies the number of buffers assigned to the ISAM data set. This improves performance when the ISAM data set's records are accessed.
- VSDSET1 DD, which describes the key-sequenced VSAM cluster, EXAMPLE.KSDS1, that is to contain the copied records.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The REPRO command copies all records from the input data set, ISAMDSET, to the output data set, EXAMPLE.KSDS1. Its parameters are:

- INFILE, which points to the INDSET1 DD statement. The INDSET1 DD statement identifies the source, or input, data set: ISAMDSET. ISAMDSET is an indexed-sequential data set.
- OUTFILE, which points to the VSDSET1 DD statement. The VSDSET1 DD statement identifies the key-sequenced VSAM cluster into which the input records are to be copied.

Copy Records Into a VSAM Cluster: Example 2

In this two-part example, data records are copied from the nonVSAM data set SEQ.D27V, a sequential data set, into an entry-sequenced VSAM cluster, MYDATA. Next, records are copied from the entry-sequenced cluster, MYDATA, into a key-sequenced cluster, ENTRY.

```
//REPRO2   JOB  ...
//JOBCAT   DD   DSNAME=D27UCAT1,DISP=SHR
//         DD   DSNAME=MYCAT,DISP=SHR
//STEP1    EXEC PGM=IDCAMS
//INPUT    DD   DSNAME=SEQ.D27V,DISP=SHR,DCB=(BUFNO=6)
//OUTPUT   DD   DSNAME=MYDATA,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     REPRO -
          INFILE(INPUT) -
          OUTFILE(OUTPUT)
/*
//STEP2    EXEC PGM=IDCAMS
//INPUT    DD   DSNAME=MYDATA,DISP=OLD
//OUTPUT   DD   DSNAME=ENTRY,DISP=OLD
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     REPRO -
          INFILE(INPUT) -
          OUTFILE(OUTPUT) -
          FROMKEY(DEAN) -
          TOKEY(JOHNSON)
/*
```

STEP1:
Access Method Services copies records from a sequential data set, SEQ.D27V, into a key-sequenced VSAM cluster, MYDATA. STEP1's job control statements are:

- JOBCAT DD, which makes two catalogs available for this job: D27UCAT1 and MYCAT. Concatenated JOBCAT DD statements are used to identify both catalogs.
- INPUT DD, which identifies the sequential data set, SEQ.D27V, that contains the input records. The BUFNO parameter specifies the number of buffers assigned to the sequential data set. This improves performance when the data set's records are accessed.
- OUTPUT DD, which identifies the entry-sequenced VSAM cluster, MYDATA, that the records are copied into.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

STEP1's REPRO command copies all records from the input data set, SEQ.D27V, to the output data set, MYDATA. Its parameters are:

- INFILE, which points to the INPUT DD statement. The INPUT DD statement identifies the source, or input, data set.
- OUTFILE, which points to the OUTPUT DD statement. The OUTPUT DD statement identifies the key-sequenced cluster into which the input records are to be copied.

STEP2:
Access Method Services copies some of the records of the entry-sequenced cluster MYDATA into another key-sequenced cluster, ENTRY. STEP2's job control statements are:

- JOBCAT DD, which applies to this job step as well as to STEP1.
- INPUT DD, which identifies the entry-sequenced cluster, MYDATA, that contains the input records.
- OUTPUT DD, which identifies the key-sequenced cluster, ENTRY, that the records are copied into.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

STEP2's REPRO command copies records from the input data set, MYDATA, to the output data set, ENTRY. Only those records with key values from DEAN to, and including, JOHNSON are copied. The REPRO command's parameters are:

- INFILE, which points to the INPUT DD statement. The INPUT DD statement identifies the source, or input, entry-sequenced cluster.
- OUTFILE, which points to the OUTPUT DD statement. The OUTPUT DD statement identifies the key-sequenced cluster into which the input records are to be copied.
- FROMKEY and TOKEY, which specify the lower and upper key boundaries. Only those records with key values from DEAN to, and including, JOHNSON are copied.

If ENTRY already contains records, VSAM merges the copied records with ENTRY's records. A subsequent job step could resume copying the records into ENTRY, beginning with the records with key greater than JOHNSON. If you subsequently copied records with key values less than DEAN into ENTRY, VSAM merges them with ENTRY's records (data records are added to the end of the cluster, but not to the beginning.)

Copy a Catalog: Example 3

In this example, a catalog is copied to illustrate the copy-catalog procedure.

```
//COPYCAT  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
          DEFINE USERCATALOG -
          (NAME(COPYUCAT) -
                MASTERPW(UCATMPW) -
                UPDATEPW(UCATUPW) -
                FOR(365) -
                VOLUME(VSER06) -
                CYLINDERS(100 10) ) -
          DATA -
                (CYLINDERS(20 10) ) -
          INDEX -
                (CYLINDERS(10) ) -
          CATALOG -
                (AMASTCAT/MRCATPW2)
/*
//STEP2    EXEC   PGM=IDCAMS
//STEPCAT  DD     DSNAME=COPYUCAT,DISP=OLD
//         DD     DSNAME=MYCAT,DISP=OLD
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     REPRO -
          INDATASET(MYCAT) -
          OUTDATASET(COPYUCAT/UCATMPW)
     EXPORT -
          MYCAT -
          DISCONNECT
/*
//STEP3    EXEC   PGM=IDCAMS
//STEPCAT  DD     DSNAME=COPYUCAT,DISP=SHR
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     LISTCAT
/*
//STEP4    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DELETE -
          MYCAT -
          CLUSTER -
          PURGE -
          CATALOG(COPYUCAT/UCATMPW)
/*
//STEP5    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DEFINE ALIAS -
          (NAME(MYCAT) -
          RELATE(COPYUCAT) )
/*
```

STEP1:
A user catalog, COPYUCAT, is defined on volume VSER06. The catalog entries in MYCAT will be copied into COPYUCAT. STEP1's job control statement, SYSPRINT DD, which is required in all Access Method Services job steps, identifies the output device to which Access Method Services messages to the programmer are sent.

The DEFINE USERCATALOG command defines the user catalog, COPYUCAT. Its parameters are:

- NAME, which specifies the name of the new catalog, COPYUCAT.
- MASTERPW and UPDATEPW, which specify the master and update level passwords for the catalog.
- FOR, which specifies that the catalog is to be retained for 365 days.
- VOLUME, which specifies that the catalog is to reside on volume VSER06.
- CYLINDERS, which specifies that the catalog's data space is initially to be 100 cylinders. When the data space is extended, it is to be extended in increments of 10 cylinders. Part of the data space contains the catalog, COPYUCAT. The rest of the data space is available for suballocated VSAM clusters.
- DATA(CYLINDERS) and INDEX(CYLINDERS) specifies that the catalog itself is initially to occupy 30 cylinders. VSAM determines the proportion of space to allocate to the catalog's data and index components. When the catalog's data component is extended, it is to be extended in increments of 10 cylinders.
- CATALOG, which specifies that a user catalog connector entry is to be built and written into the AMASTCAT catalog. The user catalog connector entry points to COPYUCAT.

STEP2:
Access Method Services copies the contents of MYCAT into COPYUCAT. STEP2's job control statements are:

- STEPCAT DD, which makes two catalogs available for this job step: COPYUCAT and MYCAT. The DD statements that identify the catalogs are concatenated.
- SYSPRINT DD (see STEP1 above).

The REPRO command copies all records from MYCAT into COPYUCAT. Access Method Services treats each catalog as a key-sequenced VSAM cluster and copies each catalog record. Consequently, the first thirteen catalog records of MYCAT, which describe MYCAT as a key-sequenced cluster, are also copied into COPYUCAT. Entries from MYCAT are written into COPYUCAT beginning with catalog record 14 (that is, after the thirteen self-describing records of COPYUCAT). The REPRO command's parameters are:

- INDATASET, which identifies the source, or input, data set: MYCAT. VSAM assumes that MYCAT is cataloged in either MYCAT or COPYUCAT.
- OUTDATASET, which identifies the receiving data set: COPYUCAT. VSAM assumes that COPYUCAT is cataloged in either COPYUCAT or MYCAT.

The EXPORT command removes MYCAT's user catalog connector entry from the master catalog. MYCAT's cataloged objects are now not available to the system. (STEP5 builds an alias entry that relates MYCAT to COPYUCAT, making the cataloged objects available to the system again.)

STEP3:
Access Method Services lists the name of each entry in the new catalog, COPYUCAT. The STEPCAT DD statement identifies the catalog to be listed.

STEP4:
Access Method Services removes the entries in COPYUCAT that describe MYCAT as a key-sequenced cluster. (See STEP2 description above.) The DELETE command's parameters are:

- MYCAT, which identifies the object to be deleted.
- CLUSTER, which specifies that MYCAT is cataloged as a cluster in in COPYUCAT.MYCAT's cluster, data, and index entry are removed from COPYUCAT.
- PURGE, which specifies that MYCAT is to be deleted regardless of a specified retention period or date.
- CATALOG, which specifies that MYCAT is cataloged in COPYUCAT. COPYUCAT's master password, UCATMPW, is required to delete its catalog entries.

STEP5:
Access Method Services builds an alias entry that relates MYCAT to COPYUCAT. Because no CATALOG parameter or JOBCAT or STEPCAT DD statement identifies the catalog that is to contain the alias entry, VSAM assumes the entry is to be written into the master catalog.

When MYCAT was defined, it was defined on volume VSER04. Its catalog entries describe VSAM objects on that volume. COPYUCAT is defined on volume VSER06. Because MYCAT was copied into COPYUCAT (and MYCAT no longer exists as a user catalog), COPYUCAT entries now describe VSAM objects on volumes VSER04 and VSER06.

Printing Data Sets

You use the PRINT command to list part or all of an indexed-sequential, sequential, or VSAM data set. "Appendix A: Sample Output from PRINT" contains sample listings produced by this command.

Both key-sequenced and entry-sequenced data can be listed, and the components of a key-sequenced data set can be listed individually. To list a component of a key-sequenced data set, specify the component name as the data set name.

Sequential and entry-sequenced data sets are listed in physical sequential order. Indexed-sequential and key-sequenced data sets can be listed in key order or in physical sequential order.

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

- Its relative byte address (RBA) for entry-sequenced data sets.
- Its key for indexed-sequential and key-sequenced data sets.
- Its sequential record number for sequential data sets.

Note: If four recoverable errors are encountered while trying to read the input, the listing is terminated.

PRINT Command

The format of the PRINT command is:

PRINT	**INFILE(***dname* [*/ password*]) | **INDATASET(***entryname* [*/ password*]) **[OUTFILE(***dname***)]** **[FROMKEY(***key***) | FROMADDRESS(***address***) | SKIP(***count***)]** **[TOKEY(***key***) | TOADDRESS(***address***) | COUNT(***count***)]** **[HEX | CHARACTER | DUMP]**

where:

INFILE(*dname* [*/ password*]) \|
INDATASET(*entryname* [*/ password*])
identifies the data set or component to be listed. Either INFILE or INDATASET must be specified.

INFILE(*dname* [*/ password*])
is the name of the DD statement that identifies the data set or component. If you are listing a catalog, the referenced DD statement cannot be the DD statement for JOBCAT or STEPCAT. INFILE can be abbreviated IFILE.

INDATASET(*entryname* [*/ password*])
specifies the name of the entry to be printed. If INDATASET is specified, the entryname is dynamically allocated. INDATASET can be abbreviated IDS.

password
If a VSAM data set or component is password protected, a password must be supplied. The password to be supplied is the master level password of the catalog if you are listing a catalog, or the read or higher level password of the data set or component if the data set or component is not a catalog. You can specify the master level password of the cluster if you are listing a protected component of a password protected cluster. Passwords are applicable only to VSAM data sets and their components. If you are listing a catalog, the referenced DD statement cannot be the DD statement for JOBCAT or STEPCAT.

OUTFILE(*dname* **)**
identifies an alternate output data set—that is, an output data set other than SYSPRINT. For *dname* substitute the name of the JCL statement that identifies the alternate output data set. The standard Access Method Services output data set for listings, which is identified by the DD name SYSPRINT, is the default. The output data set must meet the requirements stated in "JCL and Dynamic Allocation" in the chapter "Introduction." OUTFILE can be abbreviated OFILE.

FROMKEY(*key* **) \| FROMADDRESS(** *address* **) \| SKIP(** *count* **)**
specifies the location in the data set being listed from which listing is to start. If no value is specified, the listing begins with the first logical record in the data set or component. The only value that can be specified for a SAM data set is SKIP.

FROMKEY
specifies the key of the first record you want listed. You can specify generic keys—that is, keys shorter than that defined for the data set. If you specify generic keys, listing begins at the first record whose key matches that portion of the key you specified. (You cannot specify a key longer than that defined for the data set. If you do, the listing is not performed.) If the specified key is not found, the next higher key is used as the starting point for the listing. FROMKEY can be specified only when the cluster name is specified for a key-sequenced data set or for an ISAM data set. The key must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a key must be coded as two single quotation marks; it may contain up to 255 hexadecimal characters. FROMKEY can be abbreviated FKEY.

FROMADDRESS
specifies the relative byte address (RBA) of the first record you want listed. The RBA value must be the beginning of a logical record. If you specify this parameter for a key-sequenced data set, the listing will be in a physical sequential order instead of in a logical sequential order. FROMADDRESS can be specified only for VSAM data sets or components. The address may be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks. FROMADDRESS can be abbreviated FADDR.

SKIP
specifies the number of logical records you want to skip before the listing of records begins. For example, if you want the listing to begin with record number 500, you specify SKIP(499). The value specified for *count* is a number that may contain one to ten numeric characters (0-9). If the amount is specified in hexadecimal or binary, it must be preceded by X or B, be enclosed in single quotation marks, and cannot be longer than one fullword.

TOKEY(*key* **) | TOADDRESS(** *address* **) | COUNT(** *count* **)**
specifies the location in the data set being listed at which listing is to stop. If no value is specified, the listing ends with the logical end of the data set or component. The only value that can be specified for a sequential data set is COUNT. The location at which the listing is to stop must follow the location at which the listing is to begin.

TOKEY
specifies the key of the last record to be listed. You can specify generic keys—that is, keys shorter than that defined for the data set. If you specify generic keys, listing stops after the last record is listed whose key matches that portion of the key you specified. (You cannot specify a key longer than that defined for the data set. If you do, the listing is not performed.) If the specified key is not found, the next lower key is used as the stopping point for the listing. TOKEY can be specified only when a cluster name is specified for a key-sequenced data set or for an ISAM data set. If FROMADDRESS was specified for the starting location, TOKEY cannot be specified for the ending location. The key must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a key must be coded as two single quotation marks; it may contain up to 255 hexadecimal characters.

TOADDRESS
specifies the relative byte address (RBA) of the last record you want listed. Unlike FROMADDRESS, the RBA value does not need to be the beginning of a logical record. The entire record containing the specified RBA is printed. If you specify this parameter for a key-sequenced data set, the listing will be in a physical sequential order instead of in a logical sequential order. TOADDRESS can be specified only for a VSAM data set or its data component. If FROMKEY was specified for the starting location, TOADDRESS cannot be specified for the ending location. The address can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B, respectively, and be enclosed in single quotation marks. TOADDRESS can be abbreviated TADDR.

COUNT
specifies the number of logical records to be listed. The value can be expressed in decimal, hexadecimal, or binary. If the amount is specified in hexadecimal or binary, the value must be preceded by X or B, be enclosed in single quotation marks, and cannot be longer than one fullword.

HEX | CHARACTER | DUMP
specifies the format of the listing. If no value is coded, DUMP is the default.

HEX
specifies that each byte in the logical record is to be printed as two hexadecimal digits. Key fields are listed in hexadecimal format.

CHARACTER
specifies that each byte in the logical record is to be printed as a character. Bit patterns not defining a character are printed as periods. Key fields are listed in character format. CHARACTER can be abbreviated CHAR.

DUMP
specifies that each byte in the logical record is to be printed in both hexadecimal and character format. In the character portion of the listing, bit patterns not defining a character are printed as periods. Key fields are listed in hexadecimal format.

PRINT Examples

Print a Key-Sequenced Cluster's Data Records: Example 1

In this example, the data records of a key-sequenced cluster, EXAMPLE.KSDS1, are printed in dump format. That is, each character of the record is printed in its hexadecimal and alphanumeric forms.

```
//PRINT1   JOB   ...
//JOBCAT   DD    DSNAME=D27UCAT2,DISP=SHR
//STEP1    EXEC  PGM=IDCAMS
//VSDSET1  DD    DSNAME=EXAMPLE.KSDS1,DISP=OLD
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     PRINT -
          INFILE(VSDSET1)
/*
```

The job control statements are:

- JOBCAT DD, which makes a catalog available for this job: D27UCAT2. Concatenated JOBCAT DD statements were used to identify both catalogs.
- VSDSET1 DD, which identifies the key-sequenced VSAM cluster, EXAMPLE.KSDS1, that contains the records to be printed.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The PRINT command prints data records of the key-sequenced cluster, EXAMPLE.KSDS1. Its parameter, INFILE, points to the VSDSET1 DD statement, which identifies the key-sequenced cluster. VSAM assumes that the cluster is cataloged either in the D27UCAT1 user catalog or the master catalog.

Because neither FROMADDRESS, FROMKEY, SKIP, TOKEY, TOADDRESS, or COUNT is specified, Access Method Services assumes that all of the cluster's data records are to be printed.

Because neither HEX nor CHAR was specified, Access Method Services prints each record in the DUMP format. An example of the printed record is shown in Figure 13.

```
KEY OF RECORD - 00F0F0F0F0F1C9E240C4C1405CC6C9
0000  00F0F0F0 F0F1C9E2 40C4C140 5CC6C9D3   C540C9F0 C6F8F05C 40F5F040 D9C5C3D6 *.00001IS DA *FILE I0D80* 50 RECO*
0020  D9C4E240 D6C640F6 F940C3C8 C1D9E240   E6C9E3C8 40D2C5E8 40C9D540 D7D6E240 *RDS OF 69 CHARS WITH KEY IN POS *
0040  F160F1F1 4B000000 00000000 00000000                                     *1-11............
```

Figure 13. An Example of the Printed Record in DUMP Format

Copy Records From a NonVSAM Data Set Into an Entry-Sequenced VSAM Cluster, Then Print the Records: Example 2

In this example, the first fifteen records from a nonVSAM data set, EXAMPLE.NONVSAM, are copied into an entry-sequenced cluster, EXAMPLE.ESDS1. If the records were copied correctly, the cluster's records are printed in hexadecimal format. Finally, even if the records were not copied correctly, the nonVSAM data set's first fifteen records are printed in character format.

```
//PRINT2   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT1,DISP=SHR
//         DD     DSNAME=D27UCAT2,DISP=SHR
//STEP1    EXEC   PGM=IDCAMS
//VSDSET2  DD     DSNAME=EXAMPLE.ESDS1,DISP=OLD
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     REPRO -
          INDATASET( EXAMPLE.NONVSAM ) -
          OUTFILE( VSDSET2/DEPT27U ) -
          COUNT( 15 )
     IF LASTCC = 0 -
          THEN -
          PRINT -
               INFILE( VSDSET2 ) -
               HEX
     PRINT -
          INDATASET( EXAMPLE.NONVSAM ) -
          COUNT( 15 ) -
          CHAR
/*
```

The job control statements are:

- JOBCAT DD, which makes two catalogs available for this job: D27UCAT1 and D27UCAT2. Concatenated JOBCAT DD statements were used to identify both catalogs.
- VSDSET2 DD, which identifies the entry-sequenced VSAM cluster, EXAMPLE.ESDS1, that the records are copied into.

  Note: If the DCB=(BUFNO=n) parameter was specified, performance would improve when the data set's records are accessed. BUFNO was allowed to default in this example, because only 15 records are being processed.
- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.

The REPRO command copies the first fifteen records from the input data set, EXAMPLE.NONVSAM, into the output entry-sequenced cluster, EXAMPLE.ESDS1. Its parameters are:

- INDATASET, which identifies the input data set, EXAMPLE.NONVSAM. Because a JOBCAT DD statement is included with the job, VSAM assumes that the catalog entry describing the nonVSAM data set is either in D27UCAT1 or D27UCAT2 user catalog, or it is in the master catalog.
- OUTFILE, which points to the VSDSET2 DD statement. The VSDSET2 DD statement identifies the output data set, EXAMPLE.ESDS1. VSAM

assumes that EXAMPLE.ESDS1 is cataloged either in D27UCAT1 or D27UCAT2 user catalog, or in the master catalog.

- COUNT, which specifies that fifteen records are to be copied. Because neither the FROMADDRESS, FROMKEY, or SKIP parameter was specified, Access Method Services assumes that the first fifteen records are to be copied. The records are always added after the last record in the output data set.

The IF ... THEN command sequence verifies that the REPRO command completed successfully before the first PRINT command executes.

The first PRINT command prints the records in the entry-sequenced cluster, EXAMPLE.ESDS1. Its parameters are:

- INFILE, which points to the VSDSET2 DD statement. The VSDSET2 DD statement identifies the cluster, EXAMPLE.ESDS1. VSAM assumes that the cluster is cataloged either in D27UCAT1 or D27UCAT2 user catalog, or in the master catalog.
- HEX, which specifies that each record is to be printed as a group of hexadecimal characters. An example of the printed record is shown in Figure 14.

Note: The IF ... THEN command sequence ends with the HEX parameter, because no continuation character follows this parameter. A DO ... END command sequence encloses two or more Access Method Services commands that should execute only if the IF specification is satisfied.

The second PRINT command, which executes even if the REPRO command failed, prints the first fifteen records of the nonVSAM data set, EXAMPLE.NONVSAM. Its parameters are:

- INDATASET, which identifies the nonVSAM data set EXAMPLE.NONVSAM. VSAM assumes that the data set is cataloged either in D27UCAT1 or D27UCAT2 user catalog or in the master catalog.
- COUNT, which specifies that fifteen records are to be printed. Because neither FROMaddress, FROMKEY, nor SKIP was specified, Access Method Services assumes that the first fifteen records are to be printed.
- CHAR, which specifies that each record is to be printed as a group of alpha-numeric characters. An example of the printed record is shown in Figure 15.

```
RBA OF RECORD - 0
C2D6E8C5D9404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040
4040404040404040404040404040404040404040
```

Figure 14. An Example of the Printed Record in Hexadecimal

```
   RECORD SEQUENCE NUMBER - 3
   CLARK
```

Figure 15. An Example of a Printed Alphanumeric Character Record

LISTING TAPE VOLUMES MOUNTED AT CHECKPOINT

OS/VS Checkpoint/Restart explains taking checkpoints and restarting programs. Here is a summary:

During processing, a program can issue the CHKPT macro to record various information for use in restarting the processing if an error prevents the program from continuing. Recording information by way of CHKPT is called *taking a checkpoint.* The records that contain the information make up a *checkpoint entry* in the *checkpoint data set,* which contains an entry for each checkpoint that is taken.

The checkpoint data set can be a sequential data set or a partitioned data set. In a partitioned data set, each checkpoint entry is a member of it.

Checkpoint information includes the volume serial numbers of tape data sets that were open at the checkpoint. The CHKLIST command enables you to list these volume serial numbers to identify the tape data sets that need to be mounted for restart.

For a checkpoint data set with DSORG=PS (sequential data set), you can select one or more specific checkpoint entries for which the tape information is to be listed by a single CHKLIST command. By not selecting any specific entry, all checkpoint entries will be processed.

You can use CHKLIST to process a checkpoint data set with DSORG=PO (partitioned data set) in the following manner:

- Specify DSNAME=*dsname(member)* on the JCL statement that defines the checkpoint data set.
- Do not select a specific checkpoint entry, so that the single entry specified by member after dsname will be processed.

The CHKLIST command causes the following information to be listed:

- The checkpoint identifier for the entry being processed.
- For each tape data set that was open at the time of the checkpoint, the following items are listed:

  dsname

  ddname

  type of unit on which the volume was mounted

  the sequence number of the mounted volume

  volume serial numbers with an * by the volume serial number of the mounted volume

To process multiple members of a partitioned checkpoint data set, use the CHKLIST command once for each member.

Note: The CHKLIST command cannot be invoked as a TSO command.

CHKLIST Command

The format of the CHKLIST command is:

CHKLIST	**INFILE(***dname***)** **[OUTFILE(***dname***)]** **[CHECKID(***checkid*...**)]**

where:

CHKLIST
specifies that identification of each tape data set that was open at the time of a checkpoint is to be listed. CHKLIST can be abbreviated CKLST.

INFILE(*dname***)**
specifies the ddname of the DD statement that identifies the checkpoint data set that contains the checkpoint entries to be processed. The parameter must be specified and can be abbreviated IFILE.

OUTFILE(*dname***)**
specifies the ddname of the DD statement that identifies a data set other than the SYSPRINT data set to be used as an output data set. An output data set must meet the requirements shown under "Output Data Sets" in the "Introduction."If OUTFILE is not specified, the tape data set information is listed in the SYSPRINT data set. OUTFILE can be abbreviated OFILE.

CHECKID(*checkid*...**)**
specifies one or more checkpoint identifiers of entries in the checkpoint data set for which to list tape data sets that were open at the time of the checkpoint. Each checkid must be one to sixteen characters in length. It must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, slashes, or asterisks. The checkid must be the same as was specified in the CHKPT macro. A maximum of 255 checkids can be specified. Multiple checkids may be specified in any sequence.

If CHECKID is omitted, identification of tape data sets that were open at the time of a checkpoint are listed for all entries in the checkpoint data set. CHECKID must be omitted when a member of a partitioned checkpoint data set has been specified after dsname in the DD statement identified by INFILE.

If the checkpoint data set contains duplicates of a checkid, you can cause all of the checkpoint entries with that checkid to be listed either by specifying the checkid at least twice or by specifying, along with the checkid, a checkid for which there is *no* entry in the checkpoint data set. If you do neither, only the first checkpoint entry found with the checkid is listed.

CHECKID can be abbreviated CHKID.

CHKLIST Examples

Selecting Specific Checkpoint Entries: Example 1

In this example, the tape data sets that were open at checkpoint time are identified and listed on SYSPRINT for checkpoint entries C0000001 and C0000002.

```
//CHKLIST1  JOB    ...
//STEP1     EXEC   PGM=IDCAMS
//SYSPRINT  DD     SYSOUT=A
//CHKPT     DD     DSN=CHKPT.DATASET,DISP=OLD
//SYSIN     DD     *
    CHKLIST -
        INFILE(CHKPT) -
        CHECKID(C0000001 C0000002)
/*
```

The job control statements are:

- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.
- CHKPT DD, which identifies the checkpoint data set that contains the entries for which the tape data set information is to be listed.

The CHKLIST command prints the tape data set information as specified by the command's parameters:

- INFILE, which points to the CHKPT DD statement. The CHKPT DD statement identifies the checkpoint data set.
- CHECKID, which specifies that the checkpoint entries with checkids 00000001 and 00000002 are the only ones on the checkpoint data set for which the tape data set information is to be listed.

The output shown in "Appendix C: Sample Output from CHKLIST" was obtained with this JCL.

Partitioned Checkpoint Data Set: Example 2

In this example, the checkpoint data set is a partitioned data set, and member C0000001 is selected for processing by CHKLIST.

```
//CHKLIST2  JOB    ...
//STEP1     EXEC   PGM=IDCAMS
//SYSPRINT  DD     SYSOUT=A
//CHKPTDD   DD     DSN=EXAMPLE.CHKPTDS2(C0000001),DISP=SHR
//SYSIN     DD     *
    CHKLIST -
        INFILE(CHKPTDD)
/*
```

The job control statements are:

- SYSPRINT DD, which is required in all Access Method Services job steps. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are sent.
- CHKPTDD DD, which identifies the partitioned checkpoint data set and the specific member for which the tape data set information is to be listed.

The CHKLIST command prints the tape data set information as specified by its parameter, INFILE, which points to the CHKPTDD DD statement. The

CHKPTDD DD statement identifies the member of the partitioned checkpoint data set.

Note: The CHECKID parameter is omitted so that the checkpoint entry C0000001 will automatically be processed by CHKLIST.

CONTROLLING COMMAND EXECUTION

This chapter describes the

- IF-THEN-ELSE command sequence, which is used to control command execution on the basis of condition codes.
- DO-END command sequence, which specifies more than one functional Access Method Services command and its parameters.
- SET command, which is used to reset condition codes.
- PARM command, which is used to specify diagnostic-aids and printed-output options.

These commands cannot be used when Access Method Services is being executed under TSO.

Condition Codes

The condition codes that are tested in the IF-THEN-ELSE command sequence are:

- **0**, which indicates that the function was executed as directed and expected. Some informational messages may have been issued.
- **4**, which indicates that some problem was met in executing the complete function, but it was possible to continue. The continuation might not provide the user with exactly what he wanted, but no permanent harm will have been done by such continuation. A warning message was issued. An example of the kind of problem encountered is: the system was unable to locate an entry in a LISTCAT command.
- **8**, which indicates that a requested function was completed, but major specifics were unavoidably bypassed. For example, an entry to be deleted or altered could not be found in the catalog, or a duplicate name was found while an entry was being defined and the define action was terminated.
- **12**, which indicate that the requested function could not be performed. This condition code might be set as a result, for example, of the following: unable to open a user data set in REPRO or PRINT; FROMKEY or TOKEY was specified, but the data set was neither indexed sequential nor key sequenced; or the catalog named in the CATALOG parameter is not available.
- **16**, which indicates that a severe error occurred that caused the remainder of the command stream to be flushed. This condition code might be set as a result, for example, of the following: a system output data set cannot be opened; an unrecoverable error occurred in a system data set; or Access Method Services encountered improper IF-THEN-ELSE command sequences.

IF-THEN-ELSE Command Sequence

The IF-THEN-ELSE command sequence is used to control command execution.

The format of the IF-THEN-ELSE command sequence is:

IF	{**LASTCC** | **MAXCC**} *comparand number* **THEN**[*command* | **DO** *command set* **END**] [**ELSE**[*command* | **DO** *command set* **END**]]

where:

IF
specifies that one or more functional commands is to be executed based on a test of a condition code. The condition code is set by a SET command or is set to reflect the completion status of previous functional commands.

LASTCC
specifies that the condition code value that resulted from the immediately previous function command is to be compared as indicated by the *comparand* to determine whether the THEN action is to be performed. See "Condition Codes" earlier in this chapter for the meaning of condition codes.

MAXCC
specifies that the maximum condition code value that has been established by any previous function command or by a SET command is to be compared as indicated by the *comparand* to determine whether the THEN action is to be performed. See "Condition Codes" earlier in this chapter for the meaning of condition codes.

comparand
specifies the comparison to be made between the variable and the following number. This can be any of six possible comparisons:

Equal, specified as "=" or "EQ"

Not equal, specified as "¬=" or "NE"

Greater than, specified as ">" or "GT"

Less than, specified as "<" or "LT"

Greater than or equal, specified as "<=" or "GE"

Less than or equal, specified as "<=" or "LE"

number
specifies the decimal integer that is to be compared with MAXCC or LASTCC. The number can be up to ten digits long. Values greater than 16 are reduced to 16; both LASTCC and MAXCC are initialized to zero upon entry to Access Method Services. See "Condition Codes" earlier in this chapter for the meaning of condition codes.

THEN
specifies that a single command or a group of commands (introduced by DO) are to be executed if the comparison was true. THEN can be followed by another IF command.

ELSE
specifies that a single command or a group of commands (introduced by DO) is to be executed if the previous comparison is false. ELSE can be followed by another IF command.

When an IF command appears in a THEN or ELSE clause, it is called a nested IF command. The maximum level of nesting allowed is 10, starting with the first time you specify IF.

Within a nest of IF commands, the innermost ELSE clause is associated with the innermost THEN clause, the next innermost ELSE clause with the next innermost THEN clause, and so on. (To say it another way, each ELSE is matched with the nearest preceding unmatched THEN.) Should there be an IF command that does not require an ELSE clause, follow the THEN clause with a null ELSE clause (ELSE), unless the nesting structure does not require one.

DO-END Command Sequence

DO
specifies that the group of commands that follow is to be treated as a single unit, that is, to be executed as a result of a single IF command. The set of commands is terminated by END. Commands following a DO must begin on a new line.

END
specifies the end of a set of commands initiated by the nearest unended DO. END must be on a line by itself.

Null Commands

If THEN or ELSE is not followed by a continuation character or by a command in the same record, the THEN or ELSE results in no action.

The null command supports an ELSE command that balances an IF-THEN-ELSE command sequence, and allows null THEN commands. The null command is, in essence, a THEN or ELSE command that is *not* followed by a command-continuation character.

If you want to indicate a null ELSE command, specify:

```
ELSE
```

If you want to indicate a null THEN command, specify:

```
IF ... THEN
ELSE ...
```

The null command is used to indicate that no action is to be taken if the IF clause is satisfied (a null THEN command) or if the IF clause is not satisfied (a null ELSE command).

SET Command

The SET command is used to change or reset a previously defined condition code. See "Condition Codes" earlier in this chapter for the meaning of condition codes.

The format of the SET command is:

SET	{**MAXCC** | **LASTCC**}= *number*

where:

SET
specifies that a condition-code value is to be set. A SET command that follows a THEN or ELSE that is not executed does not cause the value of LASTCC or MAXCC to be altered.

MAXCC
specifies that the value to be reset is the maximum condition code set by a previous functional command. Setting MAXCC does not affect the value of LASTCC.

LASTCC
specifies that the value to be reset is the condition code set by the immediately previous functional command.

number
specifies the value to be assigned to MAXCC or LASTCC. The maximum value that can be assigned is 16; a greater value will be reduced to 16. If the value assigned to LASTCC is greater than the value of MAXCC, MAXCC is set equal to the higher value.

Note: The symbol EQ may be used instead of the "=" sign.

It is possible to terminate all processing simply by setting MAXCC or LASTCC to 16.

PARM Command

The PARM command specifies processing options to be used during execution. These options remain in effect until changed by another PARM command.

You can also specify these options in the PARM field of an EXEC JCL statement.

The format of the PARM command is:

PARM	**[TEST(** {**[TRACE]** **[AREAS(***areaid* [*areaid*...])] **[FULL((***dumpid* [*count1* [*count2*]]) [(*dumpid*...)])] | **[OFF]**})] **[GRAPHICS(CHAIN(***chain*) | **TABLE(***mname*))] **[MARGINS(***leftmargin rightmargin*)]

where:

TEST({[TRACE]
[AREAS(*areaid* [*areaid*...])]
[FULL((*dumpid* [*count1*[*count2*]])
[(*dumpid*...)])]} |
[OFF])
specifies the diagnostic aids to be used. Once the TEST option has been established, it remains in effect until it is reset by another PARM command. The TRACE, AREAS, and FULL parameters may be used concurrently.

TRACE
specifies that trace tables are to be listed whenever the built-in dump points of the processor are encountered.

AREAS(*areaid* [...])
identifies modules that are to have selected variables dumped at their dump points. Each item in the *areaid* is a two-character area-identifier defined within the implementation. See *OS/VS2 Access Method Services Logic* for more information.

FULL((*dumpid* [*count1*)[*count2*]])[...])
specifies that a region dump, as well as the trace tables and selected variables, are to be provided at the specified points. *dumpid* specifies the four-character identifier of the dump point. See *OS/VS2 Access Method Services Logic* for more information. *Count1* is a decimal integer that specifies the number of times (default is 1) the program is to go through the dump point before beginning the dump listing.*Count2* specifies a decimal integer which is the number of times (default is 1) through the dump-point that dumps are to be listed.

OFF
specifies that testing is to stop.

GRAPHICS(CHAIN(*chain*) | **TABLE(** *mname*))
specifies the print chain/train graphic character set or a special graphics table to be used in producing the output. Any character to be printed is translated to the bit pattern found in such a table at the position

corresponding to its numeric value (0-255). If the print chain does not have a graphic for a byte's bit pattern, the table should specify a period as the output graphic. See *OS/VS2 Access Method Services Logic* for more information.

CHAIN(AN | HN | PN | QN | RN | SN | TN)
specifies the graphic character set of the print chains or trains you wish employed. PN is used by the processor unless explicitly directed to use another set of graphics.

TABLE(*mname* **)**
specifies the name of a user-supplied table. This table defines the graphics for each of the possible 256 bit patterns. It must be stored as a module accessible through the LOAD macro (VS).

MARGINS(*leftmargin* b *rightmargin* **)**
specifies that the margins of input records on which command statements are written are to be changed. The normal left and right margins are 2 and 72, respectively. If MARGINS is coded, all subsequent input records are scanned in accord with the new margins. This feature may be used in conjunction with the comment feature: respecification of margins could be used to cause a /* and an */ characters to be omitted from the scan and so cause comments to be treated as commands.

leftmargin
specifies the location of the left margin.

rightmargin
specifies the location of the right margin. The right margin value must be greater than the left margin value.

Control Command Execution Examples

The examples in the topics that follow show the use of the IF-THEN-ELSE command sequence, the SET command, and the PARM command.

Control Command Execution Example: 1

In this example, nested IF commands are used to determine whether a REPRO, DELETE, or PRINT command is to be executed.

```
IF LASTCC > 4  -
  THEN IF MAXCC < 12  -
    THEN REPRO...
    ELSE DELETE...
  ELSE IF LASTCC = 4  -
    THEN
    ELSE PRINT...
```

This example specifies that if the value of LASTCC is greater than 4, then the value of MAXCC is to be tested. If the value of MAXCC is less than 12, the REPRO command is executed, while if the value of MAXCC is 12 or greater, the DELETE command is executed instead. Again, if the value of LASTCC is 4 or less, LASTCC is tested for being exactly 4: no action is to be taken in this case. If, however, LASTCC is less than 4, the PRINT command is to be executed.

Control Command Execution Example: 2

In this example, nested IF commands are used to determine whether a REPRO or PRINT command is executed.

```
IF LASTCC > 4  -
  THEN IF MAXCC < 12  -
    THEN REPRO ...
    ELSE
  ELSE IF LASTCC = 4  -
    THEN PRINT ...
```

Should the first IF command determine that LASTCC is greater than 4, and the second IF command determine that MAXCC is 12 or greater, no functional command in the example is executed. The null ELSE command is employed here to specify that the next ELSE is to correspond to the first THEN.

Control Command Execution Example: 3

In this example, if the maximum condition code is zero, a catalog is listed and a data set is printed.

```
IF MAXCC=0 THEN DO
  LISTCAT  CATALOG (AMASTCAT/MST27) ENT (MN01.0005)
  PRINT INFILE (AJK006)
  END
  ELSE . . . .
```

Control Command Execution Example: 4

If you want to list a catalog and print a data set if the last condition code is zero, but list its catalog entry before and after a VERIFY command if the last condition code is greater than zero, specify:

```
IF LASTCC = 0 THEN DO
  LISTCAT
  PRINT INFILE (AJK006)
  END
  ELSE DO
    LISTCAT ENTRY (AJK006) ALL
    VERIFY FILE (AJKJCL6)
    LISTCAT ENTRY (AJK006) ALL
    END
```

Control Command Execution Example: 5

If you want to set the last condition code established to 12, specify:

```
SET LASTCC=12
```

Control Command Execution Example: 6

If you want to replace the highest condition code established in processing so far with 8, specify:

```
SET MAXCC=8
```

RECORDS WRITTEN TO THE SMF DATA SET

Record types 63 and 67 are written to the SMF data set to support VSAM. These records conform to the SMF header format, but they are not to be treated as SMF records. These records are subject to revision.

Record Type 63 (VSAM Cluster or Component Cataloged)

Record type 63 is written after a VSAM cluster (including a catalog) or component is defined and when the definition is altered. One record is written for each catalog entry (cluster or component) affected. For example, if a cluster is defined, three records are written: one for the relationship between the components of the cluster and one for each of the components (the data and index components). This record is also written after a nonVSAM data set is defined in a VSAM catalog and when a definition is altered. The length is 128 bytes plus the length of the catalog records required to describe the cluster or component.

Record type 63 identifies the catalog in which the cluster or component is defined, gives the new definition and, for an alteration, gives the parts of the old record that have been altered. The job is identified by job log number and user identification.

Note: Record type 63 conforms to the SMF header format and is written to the SMF data set; however, record type 63 is not to be treated as an SMF record. This record is subject to revision.

The format is:

Dec. Displacement	Hex. Displacement	Field Size	Data Format	Content
0	0	1	binary	System indicator **Bit** **Meaning When Set** 6 VS2 7 VS1
1	1	1	binary	Record type 63
2	2	4	binary	Time, in hundredths of a second, record was moved to SMF buffer
6	6	4	packed	Date record was moved to SMF buffer in the form 00YYDDDF, where F is the sign
10	A	2	EBCDIC	System identification
12	C	2	EBCDIC	System model identifier
14	E	8	EBCDIC	Job name[1]
22	16	4	binary	Time, in hundredths of a second, that reader recognized JOB card for this job[1]
26	1A	4	packed	Date reader recognized JOB card for this job, in form 00YYDDDF, where F is the sign[1]
30	1E	8	EBCDIC	User identification field from common exit parameter area
38	26	1	binary	New or alteration indicator **Bit** **Meaning When Set** 0 New definition 1 Altered definition
39	27	1	binary	Type of entry 0 Cluster 1 VSAM data component 2 Index component 3 Catalog 4 NonVSAM data set 5 Generation data group 6 Alias
40	28	2	binary	Size of new catalog record[2]
42	2A	2	binary	Size of old catalog record (altered records only)[3]
44	2C	44	EBCDIC	Name of the catalog in which the entry is defined
88	58	44	EBCDIC	Entry name
132	84		binary	New catalog record[4]
			binary	Old catalog record (contains only those records that were altered)

[1] The job name and the time and date that the reader recognized the JOB statement for this job constitute the job log number.

[2] The size of the new catalog record is also the size of the first variable field.

[3] The size of the old catalog record (altered records only) is also the size of the second variable field.

[4] The format of a record in a VSAM catalog is described in *OS/VS2 Catalog Management Logic.*

Record Type 67 (VSAM Entry Deleted)

Record type 67 is written when a VSAM catalog entry (a cluster, catalog, component, or nonVSAM data set) is deleted. The length is 126 bytes plus the length of the catalog records that describe the entry.

Record type 67 identifies the entry deleted and the VSAM catalog in which the object was defined. Record type 67 also gives the record that was deleted from the catalog. A record is written for each entry deleted. For example, three records are written for an indexed cluster—one for the relationship between the components of the cluster, one for the data component, and one for the index component. The job is identified by job log number and user identification.

Note: Record type 67 conforms to the SMF header format and is written to the SMF data set; however, record type 67 is not to be treated as an SMF record. This record is subject to revision.

The format is:

Dec. Displacement	Hex. Displacement	Field Size	Data Format	Content
0	0	1	binary	System indicator **Bit** **Meaning When Set** 6 VS2 7 VS1
1	1	1	binary	Record type 67
2	2	4	binary	Time, in hundredths of a second, record was moved to SMF buffer
6	6	4	packed	Date record was moved to SMF buffer, in the form 00YYDDDF, where F is the sign
10	A	2	EBCDIC	System identification
12	C	2	EBCDIC	System model identifier
14	E	8	EBCDIC	Job name[1]
22	16	4	binary	Time, in hundredths of a second, that reader recognized the JOB card for this job[1]
26	1A	4	packed	Date reader recognized the JOB card for this job, in the form 00YYDDDF, where F is the sign[1]
30	1E	8	EBCDIC	User identification field from common exit parameter area
38	26	1	binary	Type of deletion **Bit** **Meaning When Set** 0 Uncataloged[2] 1 Scratched[2]

[1]The job name and the time and date that the reader recognized the JOB statement for this job constitute the job log number.

[2]Both indicators are set for a VSAM cluster or component; for other entries, only the uncataloged bit is set, except for a nonVSAM entry, in which case the uncataloged bit is always set, but the scratched bit is set only if the physical nonVSAM space was deleted.

Dec. Displacement	Hex. Displacement	Field Size	Data Format	Content
39	27	11	binary	Indicator of object deleted[3] **Bit** **Meaning When Set** 0 Cluster 1 Data component 2 Index component 3 Catalog 4 NonVSAM data set 5 Generation data group 6 Alias
40	28	44	EBCDIC	Name of the catalog in which the entry was defined
84	54	44	EBCDIC	Name of the entry that was deleted
128	80	2	binary	Size of the catalog record that defined the entry[4]
130	82		binary	Catalog record[5]

[3] An index alone cannot be deleted but, when a cluster is deleted, one of the three catalog records deleted is a record for the index if the cluster was indexed.

[4] The size of the entry is also the size of the field containing the catalog record.

[5] The format of a record in a VSAM catalog is described in *OS/VS2 Catalog Management Logic.*

APPENDIX A: SAMPLE OUTPUT FROM PRINT

Figure 16 shows the three different formats of output available from the PRINT command.

Dump Format

```
IDCAMS  SYSTEM SERVICES                          TIME:  19:25:14     03/08/73  PAGE  2

LISTING OF DATA SET -EXAMPLE.LISTC

KEY OF RECORD - F0F0F0F0F0F0F0F0F0F0F0F0F0F1
0000    C1C2C3C4 F0F0F0F0 F0F0F0F0 F0F0F0F1   C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7   *ABCD000000000001ABCDEFGHIJKLMNOP*
0020    D8D9E2E3 E4E5E6E7 E8E9F0F1 F2F3F4F5   F6F7F8F9 C1C2C3C4 C5C6C7C8 C9D1D2D3   *QRSTUVWXYZ0123456789ABCDEFGHIJKL*
0040    D4D5D6D7 D8D9E2E3 E4E5E6E7 F8F9F0F1   F2F3F4F5 F6F7F8F9 C1C2C3C4 C5C6C7C8   *MNOPQRSTUVWXYZ0123456789ABCDEFGH*
0060    C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7   E8E9F0F1 F2F3F4F5 F6F7F8F9 C1C2C3C4   *IJKLMNOPQRSTUVWXYZ0123456789ABCD*
0080    C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3   E4E5E6E7 E8E9F0F1 F2F3F4F5 F6F7F8F9   *EFGHIJKLMNOPQRSTUVWXYZ0123456789*
00A0    C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7   D8D9E2E3 E4E5E6E7 E8E9F0F1 F2F3F4F5   *ABCDEFGHIJKLMNOPQRSTUVWXYZ012345*
00C0    F6F7F8F9 C1C2C3C4 C5C6C7C8 C9D1D2D3   D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E9F0F1   *6789ABCDEFGHIJKLMNOPQRSTUVWXYZ01*
00E0    F2F3F4F5 F6F7F8F9 C1C2C3C4 C5C6C7C8   C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7   *23456789ABCDEFGHIJKLMNOPQRSTUVWX*
0100    E8E9F0F1 F2F3F4F5 F6F7F8F9 C1C2C3C4   C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3   *YZ0123456789ABCDEFGHIJKLMNOPQRST*
0120    E4E5E6E7 E8E9F0F1 F2F3F4F5 F6F7F8F9   C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7   *UVWXYZ0123456789ABCDEFGHIJKLMNOP*
0140    D8D9E2E3 E4E5E6E7 E8E9F0F1 F2F3F4F5   F6F7F8F9 C1C2C3C4 C5C6C7C8 C9D1D2D3   *QRSTUVWXYZ0123456789ABCDEFGHIJKL*
0160    D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E9F0F1   F2F3F4F5 F6F7F8F9 C1C2C3C4 C5C6C7C8   *MNOPQRSTUVWXYZ0123456789ABCDEFGH*
0180    C9D1D2D3 D4D5D6D7 D8D9E2E3 00000000                                       *IJKLMNOPQRST....                *
```

Hex Format

```
IDCAMS   SYSTEM SERVICES                                                  TIME:  19:26:13     03/08/73  PAGE  4

LISTING OF DATA SET -EXAMPLE.LISTC.DATA

RBA OF RECORD - 0
C1C2C3C4F0F0F0F0F0F0F0F0F0F0F0F1C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8
C9D1D2D3D4D5D6D7D8D9E2D3D4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5
F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3
E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8
C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5
F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F-F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3
E4E5E6E7E8E9F0F1F2F3F4F5f6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E300000000
```

Character Format

```
IDCAMS   SYSTEM SERVICES                          TIME:  19:26:13     03/08/73  PAGE  6

LISTING OF DATA SET -EXAMPLE.LISTC.DATA

RBA OF RECORD - 0
ABCD000000000001ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ012345
6789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGH
IJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRST
UVWXYZ0123456789ABCDEFGHIJKLMNOPQRST....
```

Figure 16. PRINT Output

APPENDIX B: INTERPRETING LISTCAT OUTPUT LISTINGS

The various LISTCAT command options allow you to select the LISTCAT output that gives you the information you want. This appendix provides information on the structure of LISTCAT output when you specify certain options. It also lists and describes fields that can be printed for each type of catalog entry.

Each listed entry is identified by its type (that is, cluster, nonVSAM, data, etc.) and by its entryname. Entries are listed in alphabetic order of the entrynames, except when the ENTRIES parameter is used. The entries are then listed in the order in which they are specified in the ENTRIES parameter.

An entry which has associated entries is immediately followed by the listing of each associated entry, unless type options (CLUSTER, DATA, SPACE, etc.) have been specified which exclude the associated entry. That is, a cluster's data component (and, if the cluster is key-sequenced, its index component) is listed immediately following the cluster.

This appendix is organized in three parts:

- LISTCAT Output Keywords, which lists all field names that can be listed for each type of entry.
- Description of Keyword Fields, which describes each field name within a group of related field names.
- Examples of LISTCAT Output Listings, which describes and illustrates the LISTCAT output that results when various LISTCAT options are specified.

LISTCAT Output Keywords

This section of the appendix lists the field names associated with each type of catalog entry. Each field name is followed by an abbreviation that points to a group of related-field descriptions in the next section. Keywords are listed in alphabetic order, not in the order of appearance in the LISTCAT output.

The group names and abbreviations are:

Abbreviations	Group Names
ALC	Allocation Group
ASN	Associations Group
ATT	Attributes Group
DSP	Data Space Group
GDG	Generation Data Group Base Entry, Special Fields for
NVS	NonVSAM Entry, Special Field for
OWN	Owner and Date Group
PRT	Protection Group
STA	Statistics Group
VLS	Volumes Group
VOL	Volume Entry, Special Fields for

Alias Entry Keywords

ASSOCIATIONS (ASN)
entryname (OWN)

Cluster Entry Keywords

ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
CODE (PRT)
CTLPW (PRT)
DATA (ASN)
entryname (OWN)
HISTORY (OWN)
  CREATION (OWN)
  EXPIRATION (OWN)
  OWNER-IDENT (OWN)
INDEX (ASN)
MASTERPW (PRT)
PROTECTION (PRT)
READPW (PRT)
UPDATEPW (PRT)
USER-SECURITY-AUTHORIZATION
  RECORD (PRT)
USVR (PRT)

Data Entry Keywords

ALLOCATION (ALC)
ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
ATTRIBUTES (ATT)
AVGLRECL (ATT)
BIND (ATT)
BUFSPC (ATT)
BYTES-PER-TRACK (VLS)
CI/CA (ATT)
CISZ (ATT)
CLUSTER (ASN)
CODE (PRT)
CTLPW (PRT)
CYLF (ATT)
DEVTYPE (VLS)
DSTGW (ATT)
entryname (OWN)
ERASE (ATT)
EXTENT (VLS)
  NUMBER (VLS)
  TYPE (VLS)
EXTENTS (VLS)
  HIGH-CCHH (VLS)
  HIGH-RBA (VLS)
  LOW-CCHH (VLS)
  LOW-RBA (VLS)
  TRACKS (VLS)
FREESPACE (STA)
  %BYTES IN-CI (STA)
  %CI'S IN-CA (STA)
  TOTAL BYTES IN DATA SET (STA)
HIGH-KEY (VLS)
HIGH-KEY-RBA (VLS)
HISTORY (OWN)
  CREATION (OWN)
  EXPIRATION (OWN)
  OWNER-IDENT (OWN)
IMBED (ATT)
INH-UPD (ATT)
IXD (ATT)
KEYLEN (ATT)
LOW-KEY (VLS)
MASTERPW (PRT)
MAXLRECL (ATT)
NERAS (ATT)
NIMBD (ATT)
NIXD (ATT)
NREPL (ATT)
NUMBER (STA)
  EXCPS (STA)
  EXTENTS (STA)
NWCK (ATT)
ORD (ATT)
PGSPC (ASN)
PHYRECS PER-TRK (VLS)
PHYSICAL REC-SIZE (VLS)
PROTECTION (PRT)
RBA (ALC)
  HIGH-ALLOC (ALC)
  HIGH-USED (ALC)
RBA (VLS)
  HIGH-ALLOC (VLS)
  HIGH-USED (VLS)
RCVY (ATT)
READPW (PRT)
RECORDS (STA)
  DELETED (STA)
  INSERTED (STA)
  RETRIEVED (STA)
  TOTAL (STA)
  UPDATED (STA)
REPL (ATT)
RKP (ATT)
SHR (ATT)
SPACE (ALC)
  PRIMARY (ALC)
  SECONDARY (ALC)
  TYPE (ALC)
SPEED (ATT)
SPLITS (STA)
  CA (STA)
  CI (STA)
STATISTICS (STA)
SUBAL (ATT)
SYSTEM-TIMESTAMP (STA)
TEMP-EXP (ATT)
TRACKS PER-CA (VLS)
UNORD (ATT)
UNQ (ATT)
UPDATEPW (PRT)
USER-SECURITY-AUTHORIZATION-RECORD (PRT)
USVR (PRT)
VOLFLAG (VLS)
VOLSER (VLS)
VOLUMES (VLS)
WCK (ATT)

Index Entry Keywords

ALLOCATION (ALC)
ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
ATTRIBUTES (ATT)
AVGLRECL (ATT)
BIND (ATT)
BUFSPC (ATT)
BYTES-PER-TRACK (VLS)
CI/CA (ATT)
CISZ (ATT)
CLUSTER (ASN)
CODE (PRT)
CTLPW (PRT)
CYLF (ATT)
DEVTYPE (VLS)
DSTGW (ATT)
entryname (OWN)
ERASE (ATT)
EXTENT (VLS)
  NUMBER (VLS)
  TYPE (VLS)
EXTENTS (VLS)
  HIGH-CCHH (VLS)
  HIGH-RBA (VLS)
  LOW-CCHH (VLS)
  LOW-RBA (VLS)
  TRACKS (VLS)
FREESPACE (STA)
  %BYTES IN-CI (STA)
  %CI'S IN-CA (STA)
  TOTAL BYTES IN DATA SET (STA)
HIGH-KEY (VLS)
HISTORY (OWN)
  CREATION (OWN)
  EXPIRATION (OWN)
  OWNER-IDENT (OWN)
IMBED (ATT)
INDEX (STA)
  ENTRIES PER SECT (STA)
  HIGH-LEVEL RBA (STA)
  LEVELS (STA)
  SEQUENCE-SET RBA (STA)
INH-UPD (ATT)
KEYLEN (ATT)
LOW-KEY (VLS)
MASTERPW (PRT)
MAXLRECL (ATT)
NERAS (ATT)
NIMBD (ATT)
NREPL (ATT)
NUMBER (STA)
  EXCPS (STA)
  EXTENTS (STA)
NWCK (ATT)
ORD (ATT)
PGSPC (ASN)
PHYRECS PER-TRK (VLS)
PHYSICAL REC-SIZE (VLS)
PROTECTION (PRT)
RBA (ALC)
  HIGH-ALLOC (ALC)
  HIGH-USED (ALC)
RBA (VLS)
  HIGH-ALLOC (VLS)
  HIGH-USED (VLS)
RCVY (ATT)
READPW (PRT)
RECORDS (STA)
  DELETED (STA)
  INSERTED (STA)
  RETRIEVED (STA)
  TOTAL (STA)
  UPDATED (STA)
REPL (ATT)
RKP (ATT)
SHR (ATT)
SPACE (ALC)
  PRIMARY (ALC)
  SECONDARY (ALC)
  TYPE (ALC)
SPEED (ATT)
SPLITS (STA)
  CA (STA)
  CI (STA)
STATISTICS (STA)
SUBAL (ATT)
SYSTEM-TIMESTAMP (STA)
TEMP-EXP (ATT)
TRACKS PER-CA (VLS)
UNORD (ATT)
UNQ (ATT)
UPDATEPW (PRT)
USER-SECURITY-AUTHORIZATION-RECORD (PRT)
USVR (PRT)
VOLFLAG (VLS)
VOLSER (VLS)
VOLUMES (VLS)
WCK (ATT)

Generation Data Group Base Entry Keywords

ASSOCIATIONS (ASN)
ATTRIBUTES (GDG)
  EMPTY (GDG)
  LIMIT (GDG)
  NOEMPTY (GDG)
  NSCR (GDG)
  SCR (GDG)
entryname (OWN)
HISTORY (OWN)
  CREATION (OWN)
  EXPIRATION (OWN)
  OWNER-IDENT (OWN)
NONVSAM (ASN)

NonVSAM Entry Keywords

ALIAS (ASN)
ASSOCIATIONS (ASN)
DEVTYPE (VLS)
entryname (OWN)
FSEQN (NVS)
HISTORY (OWN)
  CREATION (OWN)
  EXPIRATION (OWN)
  OWNER-IDENT (OWN)
VOLSER (VLS)

Page Space Entry Keywords

ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
CODE (PRT)
CREATION (OWN)
CTLPW (PRT)
DATA (ASN)
entryname (OWN)
EXPIRATION (OWN)
INDEX (ASN)
MASTERPW (PRT)
OWNER-IDENT (OWN)
PROTECTION (PRT)
READPW (PRT)
UPDATEPW (PRT)
USER-SECURITY-AUTHORIZATION-RECORD (PRT)
USVR (PRT)

User Catalog Entry Keywords

ALIAS (ASN)
ASSOCIATIONS (ASN)
DEVTYPE (VLS)
entryname (OWN)
VOLFLAG (VLS)
VOLSER (VLS)

Volume Entry Keywords

ATTRIBUTES (DSP)
  AUTO (DSP)
  EXPL (DSP)
  MCAT (DSP)
  SUBAL (DSP)
  UCAT (DSP)
  UNQ (DSP)
BYTE/TRK (VOL)
CYL/VOL (VOL)
DATASET DIRECTORY (DSP)
  ATTRIBUTES (DSP)
  EXTENTS (DSP)
DATASETS (DSP)
DATASETS ON-VOLUME (VOL)
DATASPACE (DSP)
DATASPACES ON-VOLUME (VOL)
DEVTYPE (VLS)
EXTENT-DESCRIPTOR (DSP)
  BEG-CCHH (DSP)
  SPACE-MAP (DSP)
    TRACKS (DSP)
    TOTAL (DSP)
    USED (DSP)
  EXTENTS (DSP)
FORMAT 1 DSCB (DSP)
  C C H H R (DSP)
  TIMESTAMP (DSP)
MAX-DEVICE PHYREC-SIZE (VOL)
MAX-EXTENTS PER-ALLOC (VOL)
SECONDARY ALLOCATION (DSP)
TRK/CYL (VOL)
TYPE (DSP)
volume serial number (OWN)
VOLUME-TIMESTAMP (VOL)

Description of Keyword Fields

This section of the appendix contains a description of each field name. The field names are in the following groups of related information:

Abbreviations	Group Names
ALC	Allocation Group
ASN	Associations Group
ATT	Attributes Group
DSP	Data Space Group
GDG	Generation Data Group Base Entry, Special Fields for
NVS	NonVSAM Entry, Special Field for
OWN	Owner and Date Group
PRT	Protection Group
STA	Statistics Group
VLS	Volumes Group
VOL	Volume Entry, Special Fields for

Groups are in alphabetic order. Field names within each group are in alphabetic order, not the order of appearance in the listed entry.

ALC: Allocation Group

The fields in this group describe the space allocated to the data or index component defined by the entry.

RBA
Describes, in terms of RBA (relative byte address), the high limits of the space allocated to the data or index component.

HIGH-ALLOC
The highest RBA (plus 1) available within allocated space to store data.

HIGH-USED
The highest RBA (plus 1) within allocated space that actually contains data.

SPACE
Describes the total quantity of auxiliary storage allocated to the data or index component.

PRIMARY
Gives the number of units (indicated under TYPE) of space allocated to the data or index component when the cluster (that is, its data or index component) was defined. This amount of space is to be allocated whenever a data component (or key range within it, and its associated sequence set, if IMBED is an attribute of the cluster) is extended onto a new volume.

SECONDARY
Gives the number of units (indicated under TYPE) of space to be allocated whenever a data set (or key range within it) is extended on the same volume.

TYPE
Indicates the unit of space allocation:

CYL	Cylinders
REC	Records
TRK	Tracks

ASN: Associations Group

The fields in this group identify entries associated with the entry in which they appear:

- Each cluster entry has an associated data entry and, in addition, each key-sequenced cluster has an associated index entry.
- Each data or index entry has an associated cluster entry.
- A nonVSAM data set entry or a user catalog entry might have associated alias entries.
- Each page space entry has an associated data entry (the page space is cataloged as an entry-sequenced cluster, with a cluster entry and an associated data entry.)
- A generation data group base entry usually has many associated nonVSAM data set entries.
- An alias entry has an associated nonVSAM data set entry or user-catalog entry and associated alias entries (all alias entries for a nonVSAM data set or a user catalog are chained together).

ALIAS
Identifies an alias entry.

CLUSTER
Identifies a cluster entry.

DATA
Identifies a data entry.

GDG
Identifies a generation data group (GDG) base entry.

INDEX
Identifies an index entry.

NONVSAM
Identifies a nonVSAM data set entry.

PGSPC
Identifies a page space entry.

UCAT
Identifies a user catalog entry.

ATT: Attributes Group

The fields in this group describe the attributes of the data or index component defined by the entry. See the DEFINE command for further discussion of most of these attributes.

AVGLRECL
The average length of data records. AVGLRECL equals MAXLRECL when the records are fixed-length.

BIND
The component is staged from mass storage to a direct-access storage staging drive when it is opened, and it is retained (bound) in direct-access storage until it is closed.

BUFSPC
The minimum buffer space in virtual storage to be provided by a processing program.

CI/CA
The number of control intervals per control area.

CISZ
The size of a control interval.

CYLF
The component isn't staged from mass storage to a direct-access storage staging drive when it is opened, but data from it is staged as the data is needed.

DSTGW
The component is destaged from direct-access storage to mass storage before VSAM returns control to the program that is closing it.

ERASE
Records are to be erased (set to binary 0s) when deleted.

IMBED
The sequence-set index record is stored along with its associated data control area.

INH-UPD
The data component cannot be updated. Either the data component was exported with INHIBITSOURCE specified, or its entry was modified by way of ALTER, with INHIBIT specified.

IXD
The data component has an index—it is key-sequenced.

KEYLEN
The length of the key field in a data record.

MAXLRECL
The maximum length of data or index records. AVGLRECL equals MAXLRECL when the records are fixed-length.

NERAS
Records are not to be erased (set to binary 0s) when deleted.

NIMBD
The sequence-set index record is not stored along with its associated data control area.

NIXD
The data component has no index—it is entry-sequenced.

NREPL
Index records are not replicated.

NWCK
Write operations are not checked for correctness.

ORD
Volumes are used for space allocation in the order they were specified when the cluster was defined.

REPL
Index records are replicated (that is, each is duplicated around a track of the index's direct-access device.)

RKP
The relative key position—the displacement from the beginning of a data record to its key field.

RCVY
A temporary CLOSE is issued as each control area of the data set is loaded, so the whole data set won't have to be reloaded if a serious error occurs during loading.

SHR
(n,m) The numbers n and m identify the types of sharing permitted. See SHAREOPTIONS in the DEFINE CLUSTER section for more details.

SPEED
CLOSE is not issued until the data set has been loaded.

SUBAL
More than one VSAM cluster can share the data space. A VSAM catalog might also occupy the data space.

TEMP-EXP
The data component was temporarily exported.

UNORD
Volumes specified when the cluster was defined can be used for space allocation in any order.

UNQ
Only one VSAM cluster or catalog can occupy the data space—the cluster or catalog is unique.

WCK
Write operations are checked for correctness.

DSP: Data Space Group

The fields in this group are included by LISTCAT, as part of a volume entry, for each data space on the volume. If a volume contains no data spaces, it is a candidate volume.

ATTRIBUTES
Describes the attributes of the data space.

AUTO
The data space was created automatically, or implicitly, as part of a secondary allocation operation. A VSAM cluster needed additional space on a volume, but couldn't get it from an existing data space.

EXPL
The data space was created explicitly by a DEFINE SPACE, DEFINE PAGESPACE, DEFINE CATALOG, or DEFINE CLUSTER command.

MCAT
The data space contains the master catalog.

SUBAL
The data space might contain several VSAM clusters.

UCAT
The data space contains a user catalog.

UNQ
The data space contains a single (unique) VSAM cluster or catalog.

DATASET DIRECTORY
Lists the VSAM data sets that can be stored (see CAN below) or actually are stored (in whole or in part) in the data space.

ATTRIBUTES
Describes the relation between the named data set and the data space.

CAN The data space is a candidate for storing the data set
(NULL) The data set is stored (in whole or in part) in the data space

EXTENTS
The number of data set extents for the data set within the data space.

DATASETS
The number of VSAM data sets stored (in whole or in part) in the data space. (The number includes data sets for which the data space is a candidate.)

EXTENT-DESCRIPTOR
Describes the data space extent.

BEG-CCHH
The device address (that is, CC = cylinder and HH = track) of the extent.

SPACE-MAP
A hexadecimal number that tells what tracks are used and what tracks are free in the extent. The number consists of one or more RLCs (run length codes). The first RLC gives the number of contiguous *used* tracks, starting at the beginning of the extent; if all of the tracks in the extent are used, there is only one RLC. The second RLC gives the number of contiguous *free* tracks, beyond the used tracks. A third RLC gives used tracks again, a fourth gives free tracks, and so on.

A 1-byte RLC gives the number of tracks less than 250; a 3-byte RLC gives the number of tracks equal to or greater than 250. That is, if the first byte of an RLC is X'F9' (249) or less, it is the only byte of the RLC and gives the number of tracks. If the first byte of an RLC is X'FA' (250) or more, the byte is followed by two more bytes that give the number of tracks (the first byte means nothing more than to look at the next two bytes).

TRACKS
The number of tracks:
TOTAL
Allocated to the data space altogether.
USED
Used to store data.

EXTENTS
The number of data space extents in the data space.

FORMAT 1 DSCB
Identifies the Format-1 DSCB that describes the data space.

C C H H R
The device address (that is, CC = cylinder, HH = track, and R = record number) of the DSCB in the VTOC.

TIMESTAMP
The time the data space was allocated (System/370 time-of-day clock value). The DSCB contains the timestamp.

SECONDARY ALLOCATION
Gives the number of units (indicated under TYPE) of space to be allocated whenever the data space is extended.

TYPE
Indicates the unit of space allocation:
CYL
Cylinders
TRK
Tracks

GDG: Generation Data Group Base Entry, Special Fields for

The special fields for a generation data group base entry describe attributes of the generation data group.

ATTRIBUTES
This field includes the following fields:

EMPTY
All generation data sets in the generation data group will be uncataloged when the maximum number (given under LIMIT) is reached and one more data set is to be added to the group.

LIMIT
The maximum number of generation data sets allowed in the generation data group.

NOEMPTY
Only the oldest generation data set in the generation data group will be uncataloged when the maximum number (given under LIMIT) is reached and one more data set is to be added to the group.

NSCR
Generation data sets are not to be scratched (see SCR below) when uncataloged.

SCR
Generation data sets are to be scratched (that is, the DSCB that describes each one is removed from the VTOC of the volume on which it resides) when uncataloged.

NVS: NonVSAM Entry, Special Field for

The special field for a nonVSAM data set describes a nonVSAM data set stored on magnetic tape.

FSEQN
The sequence number (for the tape volume indicated under the "VOLUMES group" keyword VOLSER) of the file in which the nonVSAM data set is stored.

OWN: Owner and Date Group

The fields in this group identify the owner of the cluster, data component, index component, nonVSAM data set, generation data group, or page space defined by the entry and give its creation and expiration dates.

entryname
The name of the cataloged object. The entryname can be specified with the ENTRIES parameter of LISTCAT to identify a catalog entry.

HISTORY
This field includes the following fields:

CREATION
The julian date (YY.DDD) the entry was created.

EXPIRATION
The julian date (YY.DDD) after which the entry can be deleted without specifying the PURGE parameter in the DELETE command. Julian date 99.999 indicates that PURGE is always required to delete the object.

OWNER-IDENT
The identity of the owner of the object described by the entry.

volume serial number
The name of the cataloged volume entry. The volume serial number can be specified with the ENTRIES parameter of LISTCAT to identify the volume entry.

PRT: Protection Group

The fields in this group describe how the cluster, data component, index component, or page space defined by the entry is protected. NULL or SUPPRESSED might be listed under PROTECTION:

NULL indicates that the object defined by the entry has no passwords.

SUPPRESSED indicates that the master password of neither the catalog nor the entry was specified, so authority to see protection information is not granted.

ATTEMPTS
Gives the number of times a console or terminal operator is allowed to attempt to enter a correct password. When a TSO terminal user is asked to supply a logon password, this counts as one attempt.

CODE
Gives the code used to tell a console or terminal operator what catalog, cluster, data component, or index component requires him to enter a password. NULL is listed under CODE if a code is not used—the object requiring the password is identified with its full name.

CTLPW
The control-interval password (that is, the password for control-interval access). NULL indicates no control-interval password.

MASTERPW
The master password.

READPW
The read-only password. NULL indicates no read-only password.

UPDATEPW
The update password. NULL indicates no update password.

USER-SECURITY-AUTHORIZATIONRECORD
The contents (1 to 255 bytes, in character format) of the USAR (user-security-authorization record).

USVR
The name of the USVR (user-security-verification routine).

STA: Statistics Group

The fields in this group give numbers and percentages that tell how much activity has taken place in the processing of a data or index component.

FREESPACE
Gives the percentages of free space in a data component and the actual number of bytes of free space in a data or index component.

%BYTES IN-CI
Percentage of space to be left free in a control interval for subsequent processing.

%CI'S IN-CA
Percentage of control intervals to be left free in a control area for subsequent processing.

TOTAL BYTES IN DATA SET
Actual number of bytes of free space in the total amount of space allocated to the data or index component.

INDEX
This field appears only in an index entry. The fields under it describe activity in the index component.

ENTRIES PER SECT
The number of entries in each section of entries in an index record.

HIGH-LEVEL RBA
The RBA (relative byte address) of the highest-level index record.

LEVELS
The number of levels of records in the index. The number is 0 if no records have been loaded into the key-sequenced data set to which the index belongs.

SEQUENCE-SET RBA
The RBA (relative byte address) of the first sequence-set record.

NUMBER
The number of:

EXCPS
EXCP (execute channel program—SVC 0) macro instructions issued by VSAM against the data or index component.

EXTENTS
Extents in the data or index component.

RECORDS
Statistics about the records in a data or index component.

DELETED
The number of records that have been deleted from the data or index component.

INSERTED
The number of records that have been added to the data or index component. It includes the number of records originally loaded.

RETRIEVED
The number of records that have been retrieved from the data or index component, whether for update or not for update.

TOTAL
The total number of records actually in the data or index component.

UPDATED
The number of records that have been retrieved for update and rewritten.

SPLITS
The number of times a control-area or control-interval split occurred—that is, half the data records in a control area or control interval were written into a new control area or control interval and then were deleted from the old control area or control interval.

CA
Control-area splits.

CI
Control-interval splits.

SYSTEM-TIMESTAMP
The time (System/370 time-of-day clock value) the data or index component was last closed (after being opened for operations that might have changed its contents.)

VLS: Volumes Group

The fields in this group identify the volume(s) on which a data component, index component, user catalog, or nonVSAM data set is stored. It also identifies candidate volume(s) for a data or index component. The fields describe the type of volume and give, for a data or index component, information about the space the object uses on the volume.

- If an entry-sequenced cluster's data component has more than one VOLUMES group, each group describes the extents that contain data records for the cluster on a specific volume.
- If a key-sequenced cluster's data component has more than one VOLUMES group, each group describes the extents that contain data records for the cluster, or one of its key ranges, on a specific volume.
- If a key-sequenced cluster's entry component has more than one VOLUMES group, each group describes the extents that contain index records for the cluster, or one of its key ranges, on a specific volume. The first VOLUMES group describes the extent that contains the high-level index records (that is, index records in levels above the sequence set level). Each of the next groups describe the extents that contain sequence-set index records for the cluster, or one of its key ranges, on a specific volume.

BYTES-PER-TRACK
The number of bytes that VSAM can write on a track.

DEVTYPE
The type of device to which the volume belongs.

EXTENT
Gives information about the extents allocated to a data or index component.

NUMBER
The number of extents allocated for the data or index component on the volume.

TYPE
The type of extents:

00 The extents are contiguous
40 The extents are not preformatted
80 A sequence set occupies a track adjacent to a control area.

EXTENTS
Gives the physical and relative-byte addresses of each extent.

HIGH-CCHH
The device address (that is, CC = cylinder and HH = track) of the end of the extent.

HIGH-RBA
The RBA (relative byte address) of the end of the extent.

LOW-CCHH
The device address (that is, CC = cylinder and HH = track) of the beginning of the extent.

LOW-RBA
The RBA (relative byte address) of the beginning of the extent.

TRACKS
The number of tracks in the extent, from low to high device addresses.

HIGH-KEY
For a key-sequenced data set, the highest hexadecimal value allowed on the volume in the key field of a record in the data set or key range. A maximum of 64 bytes can appear in HIGH-KEY.

HIGH-KEY-RBA
For a key-sequenced data set, the RBA (relative byte address) of the control interval on the volume that contains the highest keyed record in the data set or key range.

LOW-KEY
For a key-sequenced data set, the lowest hexadecimal value allowed on the volume in the key field of a record in the data set or key range. A maximum of 64 bytes can appear in LOW-KEY.

PHYRECS PER-TRACK
The number of physical records (of the size indicated under PHYSICAL REC-SIZE) that VSAM can write on a track on the volume.

PHYSICAL REC-SIZE
The number of bytes that VSAM uses for a physical record in the data or index component.

RBA
Describes, in terms of RBA (relative byte address), the high limits of the space allocated on the volume to the data component, its key range, the index component, or the sequence set records of a key range.

HIGH-ALLOC
The higest RBA (plus 1) available within allocated space to store data or index records.

HIGH-USED
The highest RBA (plus 1) within allocated space that actually contains data or index records.

TRACKS PER-CA
The number of tracks in a control area in the data component.

VOLFLAG
Indicates whether the volume is a candidate volume or the first or a subsequent volume on which data in a given key range is stored.

CANDIDATE
The volume is a candidate for storing the data or index component.

OVFLO
The volume is an overflow volume on which data records in a key range are stored.

PRIME
The volume is the first volume on which data records in a key range are stored.

VOLSER
The serial number of the volume.

VOL: Volume Entry, Special Fields for

The special fields for a volume entry describe the space that VSAM uses on the volume.

BYTE/TRK
The number of bytes that VSAM can use on each track on the volume.

CYL/VOL
The number of cylinders that VSAM can use on the volume.

DATASETS ON-VOLUME
The number of VSAM clusters that reside, in whole or in part, on the volume.

DATASPACES ON-VOLUME
The number of VSAM data spaces on the volume.

MAX-DEVICE PHYREC-SIZE
The size of the largest physical record that VSAM can write on the volume.

MAX-EXTENTS PER-ALLOC
The maximum number of extents that can be allocated on the volume for a single data set.

TRK/CYL
The number of tracks in each cylinder on the volume.

VOLUME-TIMESTAMP
The time (System/370 time-of-day clock value) VSAM last changed the contents of the volume. The Format-4 DSCB contains the timestamp at offset 76 (X'4C').

Examples of LISTCAT Output Listings

This section of the appendix illustrates the kind of output you can get when you specify LISTCAT parameters. It also describes the job control language you can specify and the output messages you get when the LISTCAT procedure executes successfully.

Job Control Language (JCL) for LISTCAT

The following example shows the job control language (JCL) statements that can be used to list a catalog's entries.

```
//LISTCAT  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//STEPCAT  DD     DSN=YOURCAT,DISP=SHR
//OUTDD    DD     DSN=LISTCAT.OUTPUT,UNIT=2400,
//          VOL=SER=TAPE10,LABEL=(1,NL),DISP=(NEW,KEEP),
//          DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     LISTCAT -
        CATALOG(YOURCAT/PASSWORD) -
        OUTFILE(OUTDD) -
        ...
/*
```

The job control statements are:

- LISTCAT, the JOB statement which contains user and accounting information required for your installation.
- STEP1, the EXEC statement which identifies the program to be executed, IDCAMS (that is, the Access Method Services program.)
- STEPCAT DD, which allocates your catalog. The catalog can be allocated dynamically if the STEPCAT and JOBCAT are omitted. If the catalog is dynamically allocated, its volume must be mounted as permanently RESIDENT or RESERVED.
- OUTDD, which specifies an alternate output file, so that the LISTCAT output can be written onto an auxilliary storage device. The LISTCAT command's OUTFILE parameter points to the OUTDD DD statement. Only the LISTCAT output is written to the alternate output device. JCL statements, system messages, and job statistics are written to the SYSPRINT output device.

  DSN=LISTCAT.OUTPUT specifies the name for the magnetic tape file.

  UNIT=2400 and VOL=SER=TAPE10 specifies that the file is to be contained on magnetic tape volume TAPE10.

  LABEL=(1,NL) specifies that this is the first file on a nonlabelled tape. You can also use a standard-labelled tape by specifying LABEL=(1,SL). If subsequent job steps produce additional files of LISTCAT output on the same tape volume, you should increment the file number in each job step's LABEL subparameter (that is,

LABEL=(2,NL) for the second job step, LABEL=(3,NL) for the third job step, etc.)

DISP=(NEW,KEEP) specifies that this is a new tape file and is to be rewound when the job finishes. If a subsequent job step prints the tape, DISP=(NEW,PASS) should be specified. If your job step contains more than one LISTCAT command, DISP=(MOD,KEEP) or DISP=(MOD,PASS) can be used to concatenate all of the LISTCAT output in one sequential file.

DCB=(RECFM=VBA,LRECL=125,BLKSIZE=129) specifies that the LISTCAT output records are variable-length, blocked 5-to-1, and are preceded by an ASCI print-control character.

- SYSPRINT DD, which is required for each Access Method Services job step. It identifies the output queue, SYSOUT=A, on which all LISTCAT output and system output messages are printed (unless the OUTFILE parameter and its associated DD statement is specified—see OUTDD above).

Note: If you want *all* output to be written to an auxiliary storage device, replace 'OUTDD'with 'SYSPRINT' in the OUTDD DD statement and omit the SYSPRINT DD SYSOUT=A statement.

- SYSIN DD, which specifies, with an asterisk (*), that the statements that follow are the input data statements. A '/*' terminates the input data statements.

The LISTCAT command parameters shown above are common to the LISTCAT examples that follow. Other LISTCAT parameters are coded with each example and the output that results is illustrated. These two parameters are optional:

CATALOG, which identifies the catalog, YOURCAT, whose entries are to be listed. If the catalog is password protected, its read (or higher level) password, PASSWORD, is required. If the passwords and protection attributes of each entry are to be listed, the catalog's master password is required.

OUTFILE, which points to the OUTDD DD statement. The OUTDD DD statement allocates an alternate output file for the LISTCAT output.

If you want to print the LISTCAT output that is contained on an alternate output file, you can use the IEBGENER program. The following shows the JCL required to print the alternate output file, LISTCAT.OUTPUT, that was allocated previously:

```
//PRINTOUT  JOB    ...
//STEP1     EXEC   PGM=IEBGENER
//SYSUT1    DD     DSN=LISTCAT.OUTPUT,UNIT=2400,
//          VOL=SER=TAPE10,LABEL=(1,NL),DISP=(OLD,KEEP),
//          DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629)
//SYSUT2    DD     SYSOUT=A
//SYSPRINT  DD     SYSOUT=A
//SYSIN     DD     DUMMY
/*
//
```

LISTCAT and AMS Output Messages

When the LISTCAT job completes, Access Method Services provides messages and diagnostic information. If an error occurred, an analysis of the error message can be found in *OS/VS Message Library: OS/VS2 System Messages.* When your LISTCAT job completes successfully, Access Method Services provides messages that follow the entry listing (see Figure 17).

```
                    LISTING FROM CATALOG -- AMASTCAT

     THE NUMBER OF ENTRIES PROCESSED WAS:
               ALIAS -------          0
               CLUSTER -----          2
               DATA --------          3
               GDG ---------          1
               INDEX -------          2
               NONVSAM -----          3
               PAGESPACE ---          1
               SPACE -------          0
               USERCATALOG -          1
               TOTAL -------         13
     THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC0002I IDCAMS PROCESSING COMPLETE, MAXIMUM CONDITION CODE WAS 0
```

Figure 17. Messages that Follow the Entry Listing

The first line identifies the catalog contained the listed entries.

The next group of lines specify the number of each entry-type, and the total number of entries, that were listed. This statistical information can help you determine the approximate size, in records, of your catalog.

The next line specifies the number of entries that couldn't be listed because the appropriate password was not specified.

The last two messages indicate that the LISTCAT command (FUNCTION) and the job step (IDCAMS) completed successfully. When LISTCAT is invoked from a TSO terminal, IDC0001I is not printed.

LISTCAT Output Listing

When you specify LISTCAT without any parameters (besides CATALOG), the entryname and type of each entry is listed (see Figure 18). The same listing would result if the NAMES parameter were specified. In this example, the ENTRIES parameter identifies each entry to be listed.

You can use this type of listing to list the name of each cataloged object and to determine the number of entries in the catalog. The total number of entries is an approximate size, in records, of your catalog.

Note: When LISTCAT is invoked from a TSO terminal and no parameters are specified or only NAME is specified (besides CATALOG), only the name of each entry associated with the TSO user's logon ID is listed.

```
  /* A:   LIST ENTRYNAMES OF SELECTED ENTRIES  */
     LISTCAT -
          ENTRIES ( -
            AMASTCAT -
            SYS1.STGINDEX -
            GDG01 -
            GDG01.G0001V00 -
            D74.PROCLIB -
            SYS1.PAGE01 -
            CAT1 -
            D27UCAT1 ) -
          CATALOG (AMASTCAT)

                          LISTING FROM CATALOG -- AMASTCAT
CLUSTER      -- AMASTCAT
   DATA      -- VSAM.CATALOG.BASE.DATA.RECORD
   INDEX     -- VSAM.CATALOG.BASE.INDEX.RECORD
CLUSTER      -- SYS1.STGINDEX
   DATA      -- VSAMDSET.DFD74077.T851EE1F.T3DC4710
   INDEX     -- VSAMDSET.DFD74077.T851EE1F.T3DEEF30
GDG BASE     -- GDG01
   NONVSAM   --                 GDG01.G0001V00
NONVSAM      -- GDG01.G0001V00
NONVSAM      -- D74.PROCLIB
PAGESPACE    -- SYS1.PAGE01
   DATA      -- VSAMDSET.DFD74053.T8500629.T7E77B30
ALIAS        -- CAT1
USERCATALOG -- D27UCAT1

IDCAMS  SYSTEM SERVICES                TIME: 22:40:00      03/18/74
                          LISTING FROM CATALOG -- AMASTCAT
          THE NUMBER OF ENTRIES PROCESSED WAS:
                    ALIAS -------          1
                    CLUSTER -----          2
                    DATA --------          3
                    GDG ---------          1
                    INDEX -------          2
                    NONVSAM -----          3
                    PAGESPACE ---          1
                    SPACE -------          0
                    USERCATALOG -          1
                    TOTAL -------         14
          THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 18. An Example of LISTCAT Output When No Parameters are Specified

LISTCAT VOLUME Output Listing

When the LISTCAT command is specified with the VOLUME parameter, the volume serial number of and device type of each volume that contains part or all of the cataloged object is listed (see Figure 19). In this example, the ENTRIES parameter identifies each entry to be listed.

```
  /* B:   LIST VOLUMES FOR SELECTED ENTRIES  */
     LISTCAT -
          VOLUME -
          ENTRIES ( -
            AMASTCAT -
            SYS1.STGINDEX -
            GDG01 -
            GDG01.G0001V00 -
            D74.PROCLIB -
            SYS1.PAGE01 -
            D27UCAT1 ) -
          CATALOG (AMASTCAT)

                          LISTING FROM CATALOG -- AMASTCAT
CLUSTER     -- AMASTCAT
   DATA     -- VSAM.CATALOG.BASE.DATA.RECORD
   VOLSER     DEVTYPE
   VS3330   X'30502009'
   INDEX    -- VSAM.CATALOG.BASE.INDEX.RECORD
   VOLSER     DEVTYPE
   VS3330   X'30502009'
CLUSTER     -- SYS1.STGINDEX
   DATA     -- VSAMDSET.DFD74077.T851EE1F.T3DC4710
   VOLSER     DEVTYPE
   VS3330   X'30502009'
   INDEX    -- VSAMDSET.DFD74077.T851EE1F.T3DEEF30
   VOLSER     DEVTYPE
   VS3330   X'30502009'
GDG BASE    -- GDG01
   NONVSAM  --                GDG01.G0001V00
   VOLSER     DEVTYPE
   VSER03   X'30C02008'
NONVSAM     -- GDG01.G0001V00
   VOLSER     DEVTYPE
   VSER03   X'30C02008'
NONVSAM     -- D74.PROCLIB
   VOLSER     DEVTYPE
   SPOOL1   X'30502009'
PAGESPACE   -- SYS1.PAGE01
   DATA     -- VSAMDSET.DFD74053.T8500629.T7E77B30
   VOLSER     DEVTYPE
   VS3330   X'30502009'
USERCATALOG -- D27UCAT1
   VOLSER     DEVTYPE
   VSER02   X'30C02008'

IDCAMS  SYSTEM SERVICES               TIME: 22:40:00      03/18/74
                          LISTING FROM CATALOG -- AMASTCAT
         THE NUMBER OF ENTRIES PROCESSED WAS:
                   ALIAS -------          0
                   CLUSTER -----          2
                   DATA --------          3
                   GDG ---------          1
                   INDEX -------          2
                   NONVSAM -----          3
                   PAGESPACE ---          1
                   SPACE -------          0
                   USERCATALOG -          1
                   TOTAL -------         13
         THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 19. An Example of LISTCAT VOLUME Output

LISTCAT SPACE ALL Output Listing

When the LISTCAT command is specified with the SPACE and ALL parameters, all of the information for each volume entry in the catalog is listed (see Figure 20). You can use this type of listing to determine how space on each cataloged volume is allocated to VSAM data spaces. You might have to list the volume's table of contents (VTOC) to determine how all of the volume's space is allocated.

```
 /* C:  LIST VOLUME INFORMATION FOR THE VOLUMES -
        CONTROLLED BY THE CATALOG  */
      LISTCAT -
         SPACE -
         ALL -
         CATALOG(AMASTCAT)

                      LISTING FROM CATALOG -- AMASTCAT

VOLUME     -- PG3330
              MAX-DEVICE                                                    MAX-EXTENTS    DATASETS   DATASPACES
   DEVTYPE------PHYREC-SIZE---BYTE/TRK---TRK/CYL---CYL/VOL---VOLUME-TIMESTAMP--------PER-ALLOC---ON-VOLUME----ON-VOLUME
   X'30502009'         13030       13165         19        411   X'84FF8CC0CA22E000'             5            5            5
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R        TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000204'    X'84FF8D18F60B6000'    UNQ      EXPL                     1            0   CYL            1
                          ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL     USED     BEG-CCHH       SPACE-MAP
                            570      570     X'001F0000'    FD023A
      DATASET DIRECTORY                                     ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74053.T84FF8D1.T81644A0                (NULL)             1
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R        TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000205'    X'84FF8D6F871A5000'    UNQ      EXPL                     1            0   CYL            1
                          ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL     USED     BEG-CCHH       SPACE-MAP
                            570      570     X'003D0000'    FD023A
      DATASET DIRECTORY                                     ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74053.T84FF8D6.TEE0A300                (NULL)             1
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R        TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000206'    X'84FF8DC5288E3000'    UNQ      EXPL                     1            0   CYL            1
                          ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL     USED     BEG-CCHH       SPACE-MAP
                            570      570     X'005B0000'    FD023A
      DATASET DIRECTORY                                     ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74053.T84FF8DC.T497F140                (NULL)             1
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R        TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000207'    X'84FF8E1B7418B000'    UNQ      EXPL                     1            0   CYL            1
                          ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL     USED     BEG-CCHH       SPACE-MAP
                            570      570     X'00790000'    FD023A
      DATASET DIRECTORY                                     ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74053.T84FF8E1.TAEC0460                (NULL)             1
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R        TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000208'    X'84FF8E7162B36000'    UNQ      EXPL                     1            0   CYL            1
                          ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL     USED     BEG-CCHH       SPACE-MAP
                            570      570     X'00970000'    FD023A
      DATASET DIRECTORY                                     ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74053.T84FF8E7.T0DD5780                (NULL)             1
```

Figure 20 (Part 1 of 2). An Example of LISTCAT SPACE ALL Output

```
                    LISTING FROM CATALOG -- AMASTCAT

VOLUME     -- VSER05
             MAX-DEVICE                                                    MAX-EXTENTS    DATASETS   DATASPACES
   DEVTYPE------PHYREC-SIZE---BYTE/TRK---TRK/CYL---CYL/VOL---VOLUME-TIMESTAMP--------PER-ALLOC---ON-VOLUME----ON-VOLUME
   X'30C02008'          7294        7294        20        203   X'851EE641C04EF000'             5           2            3
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R       TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000103'   X'851EE641C04EF000'     SUBAL   EXPL                    1           10   CYL           1
                         ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL    USED    BEG-CCHH      SPACE-MAP
                           2000       0    X'00010000'   00FD07D0
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R       TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000104'   X'851EE6CCFFD30000'     UNQ     EXPL                    1            0   CYL           1
                         ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL    USED    BEG-CCHH      SPACE-MAP
                            200      200    X'00650000'   C8
      DATASET DIRECTORY                                    ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74077.T851EE6C.TC96C030               (NULL)             1
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R       TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000105'   X'851EE6E7DbD2D000'     SUBAL   EXPL                    1           10   CYL           1
                         ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL    USED    BEG-CCHH      SPACE-MAP
                            400      200    X'006F0000'   C8C8
      DATASET DIRECTORY                                    ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74077.T851EE6E.T9748F40               (NULL)             1

VOLUME     -- VS3330
             MAX-DEVICE                                                    MAX-EXTENTS    DATASETS   DATASPACES
   DEVTYPE------PHYREC-SIZE---BYTE/TRK---TRK/CYL---CYL/VOL---VOLUME-TIMESTAMP--------PER-ALLOC---ON-VOLUME----ON-VOLUME
   X'30502009'         13030       13165        19        411   X'84FF796D05964000'             5           4            4
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R       TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000206'   X'84FF796D05864000'     SUBAL   EXPL    MCAT            1            0   CYL           1
                         ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL    USED    BEG-CCHH      SPACE-MAP
                            950      948    X'00100000'   FD03B402
      DATASET DIRECTORY                                    ATTRIBUTES   EXTENTS
         AMASTCAT                                          (NULL)             3
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R       TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'000000021A'   X'851EE1F4D6700000'     UNQ     EXPL                    1            0   CYL           1
                         ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL    USED    BEG-CCHH      SPACE-MAP
                            114      114    X'01110000'   72
      DATASET DIRECTORY                                    ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74077.T851EE1F.T3DC4710               (NULL)             1
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R       TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'000000021B'   X'851EE1F5E470C000'     UNQ     EXPL                    1            0   TRK           1
                         ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL    USED    BEG-CCHH      SPACE-MAP
                              1        1    X'00800011'   01
      DATASET DIRECTORY                                    ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74077.T851EE1F.T3DEEF30               (NULL)             1
   DATASPACE
      -----------FORMAT 1 DSCB -----------                                                SECONDARY
      C C H H R       TIMESTAMP              -------ATTRIBUTES-------   EXTENTS---ALLOCATION---TYPE---DATASETS
      X'0000000220'   X'85006298A7CE7000'     UNQ     EXPL                    1            0   CYL           1
                         ---TRACKS----
      EXTENT-DESCRIPTOR:  TOTAL    USED    BEG-CCHH      SPACE-MAP
                            570      570    X'01580000'   FD023A
      DATASET DIRECTORY                                    ATTRIBUTES   EXTENTS
         VSAMDSET.DFD74053.T8500629.T7E77B30               (NULL)             1

IDCAMS  SYSTEM SERVICES                TIME: 22:40:00      03/18/74

                    LISTING FROM CATALOG -- AMASTCAT

         THE NUMBER OF ENTRIES PROCESSED WAS:
                   ALIAS -------          0
                   CLUSTER -----          0
                   DATA --------          0
                   GDG ---------          0
                   INDEX -------          0
                   NONVSAM -----          0
                   PAGESPACE ---          0
                   SPACE -------          3
                   USERCATALOG -          0
                   TOTAL -------          3

         THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 20 (Part 2 of 2). An Example of LISTCAT ALL Output

LISTCAT ALL Output Listing

When you specify the LISTCAT command and include the ALL parameter, all of the cataloged information for each entry is listed (see Figure 21). In this example, the ENTRIES parameter identifies each entry to be listed. This example illustrates the LISTCAT output for each type of catalog entry (an entry for an entry-sequenced cluster is the same as the page space entry). You can use this type of listing to obtain all cataloged information about each entry that is listed.

Note: Because ENTRIES is specified, all entrynames identify catalog entries that are not volume entries. If a volume serial number is specified with the ENTRIES parameter, than entrynames of other entry types cannot also be specified. However, if the ENTRIES or LEVEL parameters are not specified and if entry types are not specified (that is, CLUSTER, SPACE, DATA, etc.), all entries in the catalog, including volume entries, are listed.

```
  /* D:   LIST ALL CATALOGED INFORMATION FOR SELECTED ENTRIES  */
     LISTCAT -
          ALL -
          ENTRIES ( -
            AMASTCAT -
            SYS1.STGINDEX -
            GDG01 -
            GDG01.G0001V00 -
            D74.PROCLIB -
            SYS1.PAGE01 -
            CAT1 -
            D27UCAT1 ) -
          CATALOG (AMASTCAT)

                        LISTING FROM CATALOG -- AMASTCAT

CLUSTER     -- AMASTCAT
      OWNER-IDENT    CREATION    EXPIRATION
         (NULL)       74.053       00.000
      PROTECTION -
         MASTERPW---CTLPW------UPDATEPW---READPW-----CODE-------ATTEMPTS---USVR
         EDITH       MICHEAL     GLORIA      ARCHIE      FAMILY             3   MAUDE
         USER-SECURITY-AUTHORIZATION-RECORD
         (NONE)
      ASSOCIATIONS
         DATA       VSAM.CATALOG.BASE.DATA.RECORD
         INDEX      VSAM.CATALOG.BASE.INDEX.RECORD
   DATA     -- VSAM.CATALOG.BASE.DATA.RECORD
      OWNER-IDENT    CREATION    EXPIRATION
         (NULL)       00.000       00.000
      PROTECTION (NULL)
      ASSOCIATIONS
         CLUSTER    AMASTCAT
      ATTRIBUTES
           RKP---KEYLEN-----AVGLRECL-----MAXLRECL-------BUFSPC---------CISZ---CI/CA
             0       44           505           505          3072           512      40
         SHR(3,3)  RCVY       SUBAL      NERAS      IXD        NWCK       IMBED      NREPL      UNORD
      STATISTICS                 FREESPACE:  %BYTES    %CI'S    TOTAL BYTES
         SYSTEM-TIMESTAMP                    IN-CI    IN-CA    IN DATA SET
         X'84FF79708CB1B000'                    0        0        6444032
         ----------------------------RECORDS----------------------------   ----------SPLITS--------   ----NUMBER----
              TOTAL      DELETED     INSERTED      UPDATED    RETRIEVED            CI           CA   EXCPS  EXTENTS
                15            0            0            0            1             0            0      29         2
      ALLOCATION   -----------SPACE-----------   ----------RBA----------
                   TYPE     PRIMARY   SECONDARY   HIGH-ALLOC    HIGH-USED
                   TRK                        0      6451200      5836800
         VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'      5816320        20480          512        20         3
               LOW-KEY:   00
               HIGH-KEY:  3F
               HIGH-KEY-RBA:  6144
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'00100000'   X'003C000F'       852            0      5816319
         VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'      6451200      5836800          512        20         3
               LOW-KEY:   40
               HIGH-KEY:  FF
               HIGH-KEY-RBA:  5816320
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'003D0000'   X'00410010'        93      5816320      6451199
```

Figure 21 (Part 1 of 4). An Example of LISTCAT ALL Output

```
INDEX    -- VSAM.CATALOG.BASE.INDEX.RECORD
   OWNER-IDENT   CREATION   EXPIRATION
      (NULL)      00.000      00.000
   PROTECTION (NULL)
   ASSOCIATIONS
      CLUSTER    AMASTCAT
   ATTRIBUTES
        RKP---KEYLEN-----AVGLRECL-----MAXLRECL-------BUFSPC---------CISZ---CI/CA
          0       44           0          505           0          512      20
      SHR(3,3)  RCVY      SUBAL     NERAS     NWCK      IMBED     NREPL     UNORD
   STATISTICS               FREESPACE:  %BYTES   %CI'S   TOTAL BYTES
      SYSTEM-TIMESTAMP                   IN-CI    IN-CA   IN DATA SET
      X'84FF79708CB1B000'                   0        0        190464
      ---------------------------RECORDS---------------------------   ----------SPLITS--------   ----NUMBER----
           TOTAL      DELETED      INSERTED      UPDATED    RETRIEVED            CI           CA   EXCPS  EXTENTS
             3            0             0            0            0             0            0      14         3
      -----------------INDEX-----------------------
              ENTRIES   SEQUENCE-SET   HIGH-LEVEL
      LEVELS   PER SECT            RBA          RBA
           2          7          30720            0
   ALLOCATION   -----------SPACE------------   ----------RBA----------
                TYPE      PRIMARY   SECONDARY   HIGH-ALLOC    HIGH-USED
                TRK             3           3       192000       176640
      VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
         VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
         VS3330   X'30502009'   PRIME           1   X'00'        30720          512          512        20         1
         EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                    X'003C0010'   X'003C0012'         3            0        30719
      VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
         VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
         VS3330   X'30502009'   PRIME           1   X'80'        176128        31232          512        20         3
            LOW-KEY:   00
            HIGH-KEY:  3F
         EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                    X'00100000'   X'003C000F'       852         30720       176127
      VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
         VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
         VS3330   X'30502009'   PRIME           1   X'80'        192000       176640          512        20         3
            LOW-KEY:   40
            HIGH-KEY:  FF
         EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                    X'003D0000'   X'00410010'        93        176128       191999
```

Figure 21 (Part 2 of 4). An Example of LISTCAT ALL Output

```
CLUSTER     -- SYS1.STGINDEX
      OWNER-IDENT   CREATION   EXPIRATION
         (NULL)      74.077      00.000
      PROTECTION (NULL)
      ASSOCIATIONS
         DATA      VSAMDSET.DFD74077.T851EE1F.T3DC4710
         INDEX     VSAMDSET.DFD74077.T851EE1F.T3DEEF30

   DATA     -- VSAMDSET.DFD74077.T851EE1F.T3DEEF30
      OWNER-IDENT   CREATION   EXPIRATION
         (NULL)      74.077      00.000
      PROTECTION (NULL)
      ASSOCIATIONS
         CLUSTER   SYS1.STGINDEX
      ATTRIBUTES
          RKP---KEYLEN-----AVGLRECL-----MAXLRECL-------BUFSPC---------CISZ---CI/CA
            8       12         2041         2041         5120         2048      114
         SHR(1,3)  RCVY      UNQ       NERAS     IXD       NWCK      NIMBD     NREPL     UNORD
      STATISTICS                FREESPACE:  %BYTES   %CI'S   TOTAL BYTES
         SYSTEM-TIMESTAMP                    IN-CI   IN-CA   IN DATA SET
         X'851EE1FA5B1FE000'                    0       0       1380352
         ---------------------------RECORDS--------------------------   ----------SPLITS--------   ----NUMBER----
              TOTAL      DELETED      INSERTED      UPDATED    RETRIEVED            CI           CA   EXCPS  EXTENTS
                27            0             1            0            0             0            0     159         1
      ALLOCATION   -----------SPACE-----------   ----------RBA----------
                   TYPE      PRIMARY   SECONDARY   HIGH-ALLOC    HIGH-USED
                   CYL             6           0      1400832       233472
         VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'       1400832       233472         2048         6        19
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'01110000'   X'01160012'      114            0      1400831

   INDEX    -- VSAMDSET.DFD74077.T851EE1F.T3DEEF30
      OWNER-IDENT   CREATION   EXPIRATION
         (NULL)      74.077      00.000
      PROTECTION (NULL)
      ASSOCIATIONS
         CLUSTER   SYS1.STGINDEX
      ATTRIBUTES
          RKP---KEYLEN-----AVGLRECL-----MAXLRECL-------BUFSPC---------CISZ---CI/CA
            8       12            0         1017            0         1024       11
         SHR(1,3)  RCVY      UNQ       NERAS     NWCK      NIMBD     NREPL     UNORD
      STATISTICS                FREESPACE:  %BYTES   %CI'S   TOTAL BYTES
         SYSTEM-TIMESTAMP                    IN-CI   IN-CA   IN DATA SET
         X'851EE1FA5B1FE000'                    0       0         10240
         ---------------------------RECORDS--------------------------   ----------SPLITS--------   ----NUMBER----
              TOTAL      DELETED      INSERTED      UPDATED    RETRIEVED            CI           CA   EXCPS  EXTENTS
                 3            0             0            0            0             0            0      27         1
         -----------------INDEX-----------------------
                   ENTRIES   SEQUENCE-SET   HIGH-LEVEL
         LEVELS   PER SECT        RBA          RBA
              1         10          0            0
      ALLOCATION   -----------SPACE-----------   ----------RBA----------
                   TYPE      PRIMARY   SECONDARY   HIGH-ALLOC    HIGH-USED
                   TRK             1           0        11264         1024
         VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'         11264         1024         1024        11         1
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'00800011'   X'00800011'        1            0        11263

GDG BASE    -- GDG01
      ATTRIBUTES
         LIMIT:255      SCR        NOEMPTY
      ASSOCIATIONS
         NONVSAM    GDG01.G0001V00

   NONVSAM  --                GDG01.G0001V00
      VOLSER   DEVTYPE       FSEQN
      VSER03   X'30C02008'       0
      ASSOCIATIONS
         GDG        GDG01

NONVSAM     -- GDG01.G0001V00
      VOLSER   DEVTYPE       FSEQN
      VSER03   X'30C02008'       0
      ASSOCIATIONS
         GDG        GDG01

NONVSAM     -- D74.PROCLIB
      VOLSER   DEVTYPE       FSEQN
      SPOOL1   X'30502009'       0
      ASSOCIATIONS (NULL)
```

Figure 21 (Part 3 of 4). An Example of LISTCAT ALL Output

```
PAGESPACE    -- SYS1.PAGE01
      OWNER-IDENT    CREATION    EXPIRATION
         (NULL)       74.053       00.000
      PROTECTION -
         MASTERPW---CTLPW------UPDATEPW---READPW-----CODE-------ATTEMPTS---USVR
         USER-SECURITY-AUTHORIZATION-RECORD
         (NONE)
         MPW         (NULL)      (NULL)      (NULL)      (NULL)             0    (NULL)
      ASSOCIATIONS
         DATA       VSAMDSET.DFD74053.T8500629.T7E77B30

   DATA      -- VSAMDSET.DFD74053.T8500629.T7E77B30
      OWNER-IDENT    CREATION    EXPIRATION
         (NULL)       74.053       00.000
      PROTECTION (NULL)
      ASSOCIATIONS
         PGSPC       SYS1.PAGE01
      ATTRIBUTES
           RKP---KEYLEN-----AVGLRECL-----MAXLRECL-------BUFSPC---------CISZ---CI/CA
             0        0     67895296     67895296         8192         4096      58
         SHR(1,3)  RCVY       UNQ       NERAS      NIXD       NWCK       NIMBD      NREPL      UNORD      TRKOVFL
      STATISTICS                 FREESPACE:  %BYTES    %CI'S    TOTAL BYTES
         SYSTEM-TIMESTAMP                     IN-CI    IN-CA    IN DATA SET
         X'850062EA7371F000'                      0        0        7127040
         ----------------------------RECORDS----------------------------   ----------SPLITS--------    ----NUMBER----
              TOTAL      DELETED     INSERTED      UPDATED     RETRIEVED            CI            CA    EXCPS  EXTENTS
                 0            0            0            0             0             0             0     1740         1
      ALLOCATION    -----------SPACE------------    ----------RBA----------
                    TYPE      PRIMARY    SECONDARY    HIGH-ALLOC     HIGH-USED
                    CYL            30            0       7127040             0
         VOLUMES                                    ----EXTENT----    ----------RBA----------      PHYSICAL    PHYRECS    TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330    X'30502009'    PRIME            1    X'00'       7127040             0          4096          3        19
            BYTES-PER-TRACK:      12504
            EXTENTS:    LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                        X'01580000'    X'01750012'       570             0       7127039

ALIAS        -- CAT1
      ASSOCIATIONS
         UCAT        D27UCAT1

USERCATALOG -- D27UCAT1
      VOLSER    DEVTYPE        VOLFLAG
      VSER02    X'30C02008'    PRIME
      ASSOCIATIONS
         ALIAS       CAT1

IDCAMS  SYSTEM SERVICES                TIME: 22:40:00      03/18/74

                        LISTING FROM CATALOG -- AMASTCAT

          THE NUMBER OF ENTRIES PROCESSED WAS:
                    ALIAS -------          1
                    CLUSTER -----          2
                    DATA --------          3
                    GDG ---------          1
                    INDEX -------          2
                    NONVSAM -----          3
                    PAGESPACE ---          1
                    SPACE -------          0
                    USERCATALOG -          1
                    TOTAL -------         14
          THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 21 (Part 4 of 4). An Example of LISTCAT ALL Output

LISTCAT LEVEL (SYS1) Output Listing

When you specify the LISTCAT command and include the LEVEL parameter, you can identify a group of entries to be listed (see Figure 22). In this example, LEVEL(D74) is specified. Each entry whose two-level or higher qualified entryname begins with the simple name 'D74' is listed. Four entries with two-level entrynames and one entry with a three-level entryname are listed.

You can use the LEVEL parameter to list only those entries with three-level (and higher) entrynames by specifying the first two simple names of the entryname—that is, LEVEL(D74.TSO). You cannot use the LEVEL parameter to list *only* entries with two-level entrynames (that is, to prevent listing entries with three-level and higher entrynames.)

```
  /* E:  LIST ENTRIES IDENTIFIED WITH 'LEVEL'  */
     LISTCAT -
          LEVEL(D74) -
          CATALOG (AMASTCAT)

                          LISTING FROM CATALOG -- AMASTCAT

NONVSAM      -- D74.LOAD

NONVSAM      -- D74.PROCLIB

NONVSAM      -- D74.SOURCE

NONVSAM      -- D74.TESTCASE

NONVSAM      -- D74.TSO.LOAD

IDCAMS  SYSTEM SERVICES                TIME: 22:40:00      03/18/74

                          LISTING FROM CATALOG -- AMASTCAT

          THE NUMBER OF ENTRIES PROCESSED WAS:
                    ALIAS -------          0
                    CLUSTER -----          0
                    DATA --------          0
                    GDG ---------          0
                    INDEX -------          0
                    NONVSAM -----          5
                    PAGESPACE ---          0
                    SPACE -------          0
                    USERCATALOG -          0
                    TOTAL -------          5

          THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 22. First Example of LISTCAT LEVEL Output

You can list *only* those entries whose qualified entrynames are two-level and *begin with a specified simple name.* You specify a generic name with the ENTRIES parameter—that is, ENTRIES(D74.*). Unlike the previous example, this (Figure 23) list doesn't include the entry for D74.TSO.LOAD because it has a three-level entryname. If you want to list only entries with three-level entrynames, specify either ENTRIES(D74.*.LOAD) or ENTRIES(D74.TSO.*); when you specify LEVEL(D74.TSO), you can list entries with three-level *and higher* entrynames.

```
  /* F:   LIST ENTRIES IDENTIFIED BY GENERIC NAME  */
     LISTCAT -
          ENTRIES ( -
            D74.* ) -
          CATALOG (AMASTCAT)

                        LISTING FROM CATALOG -- AMASTCAT

NONVSAM      -- D74.LOAD

NONVSAM      -- D74.PROCLIB

NONVSAM      -- D74.SOURCE

NONVSAM      -- D74.TESTCASE

IDCAMS  SYSTEM SERVICES                TIME: 22:40:00      03/18/74

                        LISTING FROM CATALOG -- AMASTCAT

          THE NUMBER OF ENTRIES PROCESSED WAS:
                    ALIAS -------          0
                    CLUSTER -----          0
                    DATA --------          0
                    GDG ---------          0
                    INDEX -------          0
                    NONVSAM -----          4
                    PAGESPACE ---          0
                    SPACE -------          0
                    USERCATALOG -          0
                    TOTAL -------          4

          THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 23. Second Example of LISTCAT LEVEL Output

LISTCAT ALLOCATION Output Listing

When you specify the LISTCAT command and include the ALLOCATION parameter, each cataloged object with space allocated to it from a VSAM data space is listed (see Figure 24). All information about the object's space is listed, but none of the object's other cataloged information is listed. The entry types that can be specified when the ALLOCATION parameter is specified are limited to: CLUSTER (including the catalog-being-listed's space information), PAGESPACE, DATA, and INDEX.

```
  /* G:   LIST SPACE INFORMATION FOR SELECTED ENTRIES  */
     LISTCAT -
        ALLOCATION -
          ENTRIES ( -
            AMASTCAT -
            SYS1.STGINDEX -
            SYS1.PAGE01 ) -
          CATALOG (AMASTCAT)

                        LISTING FROM CATALOG -- AMASTCAT

CLUSTER      -- AMASTCAT
   DATA      -- VSAM.CATALOG.BASE.DATA.RECORD
      ALLOCATION   ----------SPACE----------   ----------RBA----------
                   TYPE     PRIMARY   SECONDARY   HIGH-ALLOC    HIGH-USED
                   TRK                        0      6451200      5836800
         VOLUMES                                ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'      5816320        20480          512        20         3
               LOW-KEY:   00
               HIGH-KEY:  3F
               HIGH-KEY-RBA:  6144
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'00100000'   X'003C000F'      852            0      5816319
         VOLUMES                                ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'      6451200      5836800          512        20         3
               LOW-KEY:   40
               HIGH-KEY:  FF
               HIGH-KEY-RBA:  5816320
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'003D0000'   X'00410010'       93      5816320      6451199

   INDEX     -- VSAM.CATALOG.BASE.INDEX.RECORD
      ALLOCATION   ----------SPACE----------   ----------RBA----------
                   TYPE     PRIMARY   SECONDARY   HIGH-ALLOC    HIGH-USED
                   TRK            3           3       192000       176640
         VOLUMES                                ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'        30720          512          512        20         1
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'003C0010'   X'003C0012'         3            0        30719
         VOLUMES                                ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'80'       176128        31232          512        20         3
               LOW-KEY:   00
               HIGH-KEY:  3F
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'00100000'   X'003C000F'      852        30720       176127
         VOLUMES                                ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'80'       192000       176640          512        20         3
               LOW-KEY:   40
               HIGH-KEY:  FF
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'003D0000'   X'00410010'       93       176128       191999
```

Figure 24 (Part 1 of 2). An Example of LISTCAT ALLOCATION Output

```
CLUSTER     -- SYS1.STGINDEX
   DATA     -- VSAMDSET.DFD74077.T851EE1F.T3DC4710
      ALLOCATION   -----------SPACE-----------   ----------RBA----------
                   TYPE      PRIMARY   SECONDARY   HIGH-ALLOC    HIGH-USED
                   CYL             6           0      1400832       233472
         VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'      1400832       233472         2048         6        19
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'01110000'   X'01160012'      114            0      1400831

   INDEX    -- VSAMDSET.DFD74077.T851EE1F.T3DC4710
      ALLOCATION   -----------SPACE-----------   ----------RBA----------
                   TYPE      PRIMARY   SECONDARY   HIGH-ALLOC    HIGH-USED
                   TRK             1           0        11264         1024
         VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'        11264         1024         1024        11        1
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'00800011'   X'00800011'        1            0        11263

PAGESPACE   -- SYS1.PAGE01
   DATA     -- VSAMDSET.DFD74053.T8500629.T7E77B30
      ALLOCATION   -----------SPACE-----------   ----------RBA----------
                   TYPE      PRIMARY   SECONDARY   HIGH-ALLOC    HIGH-USED
                   CYL            30           0      7127040            0
         VOLUMES                                 ----EXTENT----   ----------RBA----------     PHYSICAL   PHYRECS   TRACKS
            VOLSER---DEVTYPE-------VOLFLAG----NUMBER---TYPE----HIGH-ALLOC----HIGH-USED-----REC-SIZE---PER-TRK---PER-CA
            VS3330   X'30502009'   PRIME           1   X'00'      7127040            0         4096         3        19
            BYTES-PER-TRACK:     12504
            EXTENTS:   LOW-CCHH------HIGH-CCHH-----TRACKS------LOW-RBA-----HIGH-RBA
                       X'01580000'   X'01750012'      570            0      7127039

IDCAMS  SYSTEM SERVICES                TIME: 22:40:00      03/18/74

                        LISTING FROM CATALOG -- AMASTCAT

          THE NUMBER OF ENTRIES PROCESSED WAS:
                    ALIAS -------          0
                    CLUSTER -----          2
                    DATA --------          3
                    GDG ---------          0
                    INDEX -------          2
                    NONVSAM -----          0
                    PAGESPACE ---          1
                    SPACE -------          0
                    USERCATALOG -          0
                    TOTAL -------          8

          THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 24 (Part 2 of 2). An Example of LISTCAT ALLOCATION Output

LISTCAT HISTORY Output Listing

When you specify the LISTCAT command and include the HISTORY parameter, only the name, ownerid, creation date, and expiration date are listed for each entry that is selected (see Figure 25). Only these types of entries have HISTORY information: CLUSTER, DATA, INDEX, GENERATIONDATAGROUP, NONVSAM, and PAGESPACE.

```
/* H: LIST HISTORY INFORMATION FOR SELECTED ENTRIES */
LISTC -
  HIST
                                LISTING FROM CATALOG -- AMASTCAT
CLUSTER     -- AMASTCAT
      HISTORY
         OWNER-IDENT---CREATION---EXPIRATION
         (NULL)          74.043      00.000
  DATA      -- VSAM.CATALOG.BASE.DATA.RECORD
     HISTORY
        OWNER-IDENT---CREATION---EXPIRATION
        (NULL)          00.000      00.000
  INDEX     -- VSAM.CATALOG.BASE.INDEX.RECORD
     HISTORY
        OWNER-IDENT---CREATION---EXPIRATION
        (NULL)          00.000      00.000
CLUSTER     -- CLUSTER.TEST1
      HISTORY
         OWNER-IDENT---CREATION---EXPIRATION
         OWNCLUS         74.072      75.072
  DATA      -- DATA.TEST1
     HISTORY
        OWNER-IDENT---CREATION---EXPIRATION
        OWNDATA         74.072      00.000
  INDEX     -- INDEX.TEST1
     HISTORY
        OWNER-IDENT---CREATION---EXPIRATION
        OWNINDX         74.072      00.000
NONVSAM     -- NONVSAM.TEST1
      HISTORY
         OWNER-IDENT---CREATION---EXPIRATION
         (NULL)          74.060      74.300
VOLUME      -- SYSRSM
USERCATALOG -- USERCAT1
IDCAMS  SYSTEM SERVICES        TIME: 08:54:25       04/26/74
                               LISTING FROM CATALOG -- AMSTCAT
         THE NUMBER OF ENTRIES PROCESSED WAS:
                   ALIAS -------          0
                   CLUSTER -----          2
                   DATA --------          2
                   GDG ---------          0
                   INDEX -------          2
                   NONVSAM -----          1
                   PAGESPACE ---          0
                   SPACE -------          1
                   USERCATALOG -          1
                   TOTAL -------          9
         THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 25. An Example of LISTCAT HISTORY Output

LISTCAT CREATION/EXPIRATION Output Listing

When you specify the LISTCAT command and include the CREATION or EXPIRATION parameter (or both), entries that have a creation or expiration date are selected according to the number of days you specify in the subparameter. In Figure 26, for example, LISTCAT was run April 16, 1974 (74.106). The creation day of 74.099 is 7 days earlier, and so entries created on or before 74.099 are listed (if they also meet the EXPIRATION criterion). If 8 days were specified for CREATION, entries created on 74.099 wouldn't be listed.

These types of entries can have a creation or expiration date: CLUSTER, DATA, INDEX, GENERATIONDATAGROUP, and NONVSAM. PAGESPACE can have a creation date.

```
/* I: LIST INFORMATION FOR ENTRIES ACCORDING TO
       CREATION AND EXPIRATION DATES */

LISTC -
   ENTRY(USERCAT1.CLUSTER.TEST1/MASTCLUS)
   EXPIR(359) -
 CREATION(7) -
   VOL

CLUSTER     -- USERCAT1.CLUSTER.TEST1
      IN-CATALOG:  USERCAT1
      HISTORY
         OWNER-IDENT---CREATION---EXPIRATION
         OWNCLUS        74.099      75.099

  DATA     -- USERCAT1.DATA.TEST1
     IN-CATALOG:  USERCAT1
     HISTORY
        OWNER-IDENT---CREATION---EXPIRATION
        OWNDATA        74.099      00.000
     VOLSER     DEVTYPE
     AMSLB1   X'30C02008'

  INDEX    -- USERCAT1.INDEX.TEST1
     IN-CATALOG:  USERCAT1
     HISTORY
        OWNER-IDENT---CREATION---EXPIRATION
        OWNINDX        74.099      00.000
     VOLSER     DEVTYPE
     AMSLB1   X'30C02008'

IDCAMS  SYSTEM SERVICES      TIME: 04:12:38        04/16/74

         THE NUMBER OF ENTRIES PROCESSED WAS:
                   ALIAS -------          0
                   CLUSTER -----          1
                   DATA --------          1
                   GDG ---------          0
                   INDEX -------          1
                   NONVSAM -----          0
                   PAGESPACE ---          0
                   SPACE -------          0
                   USERCATALOG -          0
                   TOTAL -------          3

         THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 26. An Example of LISTCAT CREATION/EXPIRATION Output

APPENDIX C: SAMPLE OUTPUT FROM CHKLIST

Figure 27 shows the format of output from the CHKLIST command.

For each checkpoint entry listed under "CHECKID," the other columns give dsname, ddname, unit, and volume information for each tape data set that was open at the time of the checkpoint. Asterisks under "UNIT" indicate an unrecognizable unit type. "VOLUME X OF Y IS CURRENT" indicates the volume sequence number of the volume mounted at the time of the checkpoint and the total number of volumes in the data set. Each volume's volume serial number is listed, and an asterisk indicates the volume mounted at the time of the checkpoint.

```
CHKLIST INFILE(CHKPT) CHECKID(C0000001 C0000002)
IDCAMS  SYSTEM SERVICES          TIME: 21:34:43       12/16/39
OPEN TAPE DATA SET LIST FROM CHECKPOINT DATA SET - CHKPT DATASET
CHECKID          DSNAME                DDNAME   UNIT       VOLUMES - * INDICATES CURRENT VOLUME
C0000001
                 USER TAPE DATASET0    DDN2VS11 2400-7TRK  VOLUME    1 OF   1 IS CURRENT
                                                           120001*
                 USER.TAPE.DATASET1    DDN3VS11 2400-9TRK  VOLUME    3 OF   4 IS CURRENT
                                                           130001   130002  130003* 130004
                 USER.TAPE.DATASET2    DDN4VS11 3400-7TRK  VOLUME    1 OF   1 IS CURRENT
                                                           140001*
C0000002
                 USER.TAPE.DATASET3    DDNAME32 2400-9TRK  VOLUME    8 OF   8 IS CURRENT
                                                           230001   230002  230003  230004  230005
                                                           230006   230007  230008*
                 USER.TAPE.DATASET4    DDNAME42 2400-9TRK  VOLUME   20 OF  24 IS CURRENT
                                                           240001   240002  240003  240004  240005
                                                           240006   240007  240008  240009  240010
                                                           240011   240012  240013  240014  240015
                                                           240016   240017  240018  240019  240020*
                                                           240021   240022  240023  240024
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Figure 27. An Example of CHKLIST Output

APPENDIX D: JCL DD PARAMETERS TO TAKE CARE WITH

Because VS2 does not disallow any DD statement parameters and subparameters for VSAM, you should be aware of the specifications that either do nothing in a VSAM environment or might cause problems for you. Figure 28 shows the DD statement parameters and subparameters to be avoided with VSAM.

Parameter	Subparameter	Comment
AFF	*ddname*	You must use this parameter carefully. If the cluster components, the data and its index, reside on unlike devices, the results of an AFF specification are unpredictable.
CHKPT	**EOV**	Because checkpoints at end of volume are executed only for BSAM and QSAM data sets, this parameter doesn't apply to VSAM data sets and need not be coded.
DATA		Because there is no way to get VSAM data into the input stream, this parameter is not applicable to VSAM.
DCB	All	The access-method control block (ACB), not the DCB, describes VSAM data sets; therefore, the DCB parameter is not applicable to VSAM. An access-method control block is generated by an ACB or GENCB macro, and can be modified by a MODCB macro.
DISP	**CATLG**	VSAM data sets are cataloged and uncataloged as a result of an Access Method Services command; if CATLG is coded, a message is issued, but the data set is not cataloged.
	DELETE	VSAM data sets are deleted as a result of an Access Method Services command; if DELETE is coded, a message is issued, but the data set is not deleted.
	MOD	For VSAM data sets, MOD is treated as if OLD were specified, except for processing with an ISAM program, in which case MOD indicates resume load.
	KEEP	Because KEEP is implied for VSAM data sets, it need not be coded.
	NEW	VSAM data spaces are initially allocated as a result of the Access Method Services DEFINE command. If NEW is specified, VS2 also allocates space, and it is never used by VSAM. Moreover, an Access Method Services request for space may fail if the DISP=NEW acquisition of space causes too little space to remain available.
	UNCATLG	VSAM data sets are cataloged and uncataloged as a result of Access Method Services commands; if UNCATLG is coded, a message is issued, but the data set is not uncataloged.

Figure 28 (Part 1 of 3). JCL DD Parameters

Parameter	Subparameter	Comment
DSNAME	*dsname(areaname)*	The name is used; areaname is ignored.
	dsname(generation)	The name is used; generation is ignored.
	dsname(member)	The name is used; member is ignored.
	All temporary *dsnames*	Because VSAM data sets are built by Access Method Services, which uses the data-set name supplied in the DEFINE command, temporary names cannot be used with VSAM.
	All backward DD references of the form **. ddname*	If the object referred to is a cluster and the data set and index reside on unlike devices, the results of a backward DD reference are unpredictable.
LABEL	**BLP, NL, NSL**	Because these subparameters have no meaning for direct-access devices, they do not apply for VSAM data sets, which all reside on direct-access storage.
	IN	Because IN is used to override DCB subparameters and the DCB parameter does not apply to VSAM data sets, IN does not apply.
	OUT	Because OUT is used to override DCB subparameters and the DCB parameter does not apply to VSAM data sets, OUT does not apply.
	NPWREAD	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NPWREAD specification in the LABEL parameter.
	PASSWORD	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NPWREAD specification in the LABEL parameter.
	SL, SUL	Although these parameters apply to direct-access storage devices, SL is always used for VSAM, whether you specify SL, SUL, or neither.
SEP	*ddname*	You must use this parameter carefully. If the cluster components, the data and its index, reside on unlike devices, the results of a SEP specification are unpredictable.
SPACE		VSAM data spaces are initially allocated as a result of the Access Method Services DEFINE command. If SPACE is specified, therefore, an extent is allocated that is never used by VSAM. Moreover, an Access Method Services request for space may fail as a result of the SPACE acquisition of space.
SPLIT		VSAM data spaces are initially allocated as a result of the Access Method Services DEFINE command. If SPLIT is specified, VS2 also allocates space, and it is never used by VSAM. Moreover, an Access Method Services request for space may fail if the SPLIT acquisition of space causes too little space to remain available.

Figure 28 (Part 2 of 3). JCL DD Parameters

Parameter	Subparameter	Comment
SUBALLOC		VSAM data spaces are initially allocated as a result of the Access Method Services DEFINE command. If SUBALLOC is specified, VS2 also allocates space, and it is never used by VSAM. Moreover, an Access Method Services request for space may fail if the SUBALLOC acquisition of space causes too little space to remain available. If the data set for which SUBALLOC is specified is a VSAM data set, the request is denied.
SYSOUT		If SYSOUT is coded with a mutually exclusive parameter (for example, DISP), the job step is terminated with an error message.
UCS	All	Because this parameter applies only to unit-record devices, it does not apply to VSAM.
UNIT	**AFF**	You must use this subparameter carefully. If the cluster components, the data and its index, reside on unlike devices, the results of UNIT=AFF are unpredictable.
	SEP	You must use this subparameter carefully. If the cluster components, the data and its index, reside on unlike devices, the results of UNIT=SEP are unpredictable.
VOLUME	**REF**	You must use this subparameter carefully. If the referenced volumes are not a subset of those contained in the catalog record for the data set, the results are unpredictable.
	volseq#	Results are unpredictable.
	volcount	This subparameter is used to request some number of nonspecific volumes. Because all VSAM volumes must be specifically defined before processing, volcount is not applicable to VSAM data sets.

Figure 28 (Part 3 of 3). JCL DD Parameters

GLOSSARY

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

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

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

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

alias: An alternative name for an entry in a VSAM catalog.

alias entry: An entry in a VSAM catalog that relates an alias (alternate entryname) to the real entryname of a user catalog or nonVSAM data set.

application: As used in this publication, the use to which an access method is put or the end result that it serves; contrasted to the internal operation of the access method.

bind: (Verb.) To keep a data set that has been staged from a mass storage volume to a direct-access storage staging drive on the staging drive until the data set is closed.

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

catalog connector: A catalog entry, called either a user catalog entry or a catalog connector entry, in the master catalog that points to a user catalog's volume (that is, it contains the volume serial number of the direct-access volume that contains the user catalog).

cluster: A data component and an index component when data is key sequenced; a data component alone when data is entry sequenced.

cluster entry: A catalog entry that contains information about a key-sequenced or entry-sequenced VSAM cluster: ownership, cluster attributes, and the cluster's passwords and protection attributes. A key-sequenced cluster entry points to a data entry and an index entry. An entry-sequenced cluster entry points to a data entry.

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

compression: (*See* key compression.)

control area: A group of control intervals used as a unit for formatting a data set before adding records to it. Also, in a key-sequenced data set, the set of control intervals pointed to by a sequence-set index record; used by VSAM for distributing free space and for placing a sequence-set index record adjacent to its data.

control-area split: The movement of the contents of some of the control intervals in a control area to a newly created control area, to make possible the insertion or lengthening of a data record when a free control interval was needed and there was none in the original control area.

control interval: A fixed-length area of auxiliary-storage space in which VSAM stores records and distributes free space. It is the unit of information transmitted to or from auxiliary storage by VSAM.

control-interval split: The movement of some of the stored records in a control interval to a free control interval, to make possible the insertion or lengthening of a record that won't fit in the original control interval.

data entry: A catalog entry that describes a cluster's, catalog's, or page space's data component. A data entry contains the data component's attributes, allocation and extent information, and statistics. A data entry for a cluster's or catalog's data component can also contain the data component's passwords and protection attributes.

data integrity: Preservation of data or programs for their intended purpose. As used in this publication, the safety of data from inadvertent destruction or alteration.

data record: A collection of items of information from the standpoint of its use in an application, as a user supplies it to VSAM for storage.

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

data set: The major unit of data storage and retrieval in the operating system, consisting of data in a prescribed arrangement and described by control information to which the system has access. As used in this publication, a collection of fixed- or variable-length records in auxiliary storage, arranged by VSAM in key sequence or in entry sequence. *(See also* key-sequenced data set and entry-sequenced data set.)

data space: A storage area defined in the volume table of contents of a direct-access volume for the exclusive use of VSAM to store data sets, indexes, and catalogs.

destage: (Verb.) To transmit data from a direct-access storage staging drive to a mass storage volume.

direct access: The retrieval or storage of data by a reference to its location in a data set rather than relative to the previously retrieved or stored data. *(See also* addressed direct access and keyed direct access.)

distributed free space: Space reserved within the control intervals of a key-sequenced data set for inserting new records into the data set in key sequence; also, whole control intervals reserved in a control area for the same purpose.

dynamic allocation: The allocation of a data set or volume by the use of the data-set name or volume serial number rather than by the use of information contained in a JCL statement.

entry: A collection of information about a cataloged object in a VSAM master or user catalog. Each entry resides in one or more 512-byte record.

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

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

extent: A continuous space allocated on a direct-access storage volume, reserved for a particular data space or data set. An extent of a data set contains a whole number of control areas.

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

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

generation data group entry: An entry in a VSAM catalog that permits nonVSAM data sets to be associated with other nonVSAM data sets as generation data sets.

generation data set: One of a collection of historically related nonVSAM data sets; the collection of these data sets is known as a generation data group.

generic key: A high-order portion of a key, containing characters that identify those records that are significant for a certain application. For example, it might be desirable to retrieve all records whose keys begin with the generic key AB, regardless of the full key values.

generic name: A qualified name in which one qualifier is replaced by an asterisk; the generic name applies to all entries that match the qualifiers supplied in the generic name.

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

index: As used in this publication, an ordered collection of pairs, each consisting of a key and a pointer, used by VSAM to sequence and locate the records of a key-sequenced data set; organized in levels of index records. (*See also* index level, index set, and sequence set.)

index entry: A catalog entry that describes a catalog's or key-sequenced cluster's index component. An index entry contains the index component's attributes, passwords and protection attributes, allocation and extent information, and statistics.

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

index record: A collection of index entries that are retrieved and stored as a group. (Contrast to data record.)

index replication: The use of an entire track of direct-access storage to contain as many copies of a single index record as possible; reduces rotational delay.

index set: The set of index levels above the sequence set. The index set and the sequence set together comprise the index.

integrity: (*See* data integrity.)

ISAM interface: A set of routines that allow a processing program coded to use ISAM (indexed sequential access method) to gain access to a key-sequenced data set with an index.

key: One or more characters within an item of data that are used to identify it or control its use. As used in this publication, one or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records. (*See also* key field and generic key.)

key compression: The elimination of characters from the front and the back of a key that VSAM does not need to distinguish the key from the preceding or following key in an index record; reduces storage space for an index.

key field: A field located in the same position in each record of a data set, whose contents are used for the key of a record.

key sequence: The collating sequence of data records, determined by the value of the key field in each of the data records. May be the same as, or different from, the entry sequence of the records.

key-sequenced data set: A data set whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the data set in key sequence by means of distributed free space. RBAs of records can change.

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

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

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

mass storage volume: The unit of mass storage in the 3850 Mass Storage System.

master catalog: A key-sequenced data set with an index containing extensive data-set and volume information that VSAM requires to locate data sets, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and to accumulate usage statistics for data sets.

nonVSAM entry: A catalog entry that describes a nonVSAM data set. A nonVSAM entry contains the data set's volume serial number and device type. If the data set resides on a magnetic tape volume, the entry can also identify the data set's file number. When the data set resides on a direct-access device, the operating system obtains further information by examining the data set's DSCB (Data Set Control Block) in the volume's VTOC (volume table of contents).

page space: A VS2 system data set. A page space is cataloged as an entry-sequenced cluster (that is, the page space entry is similar to a cluster entry, and it points to a data entry).

password: A unique string of characters stored in a catalog that a program, a computer operator, or a terminal user must supply to meet security requirements before a program gains access to a data set.

physical record: On a track of a direct-access storage device, the space between interrecord gaps.

pointer: An address or other indication of location. For example, an RBA is a pointer that gives the relative location of a data record or a control interval in the data set to which it belongs. (*See also* horizontal pointer and vertical pointer.)

portability: The ability to use VSAM data sets with different operating systems. Volumes whose data sets are cataloged in a user catalog can be demounted from storage devices of one system, moved to another system, and mounted on storage devices of that system. Individual data sets can be transported between operating systems using Access Method Services.

qualified name: A name that is segmented by periods; each name segment is referred to as a qualifier.

random access: (*See* direct access.)

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

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

relative byte address: (*See* RBA.)

replication: (*See* index replication.)

security: (*See* data security.)

sequence set: The lowest level of the index of a key-sequenced data set; it gives the locations of the control intervals in the data set and orders them by the key sequence of the data records they contain. The sequence set and the index set together comprise the index.

sequential access: The retrieval or storage of a data record in either its entry sequence or its key sequence, relative to the previously retrieved or stored record. (*See also* addressed sequential access and keyed sequential access.)

simple name: A qualifier of a qualified entryname or dsnname. Simple names may be one to eight characters and, in a series, are separated from each other by a period.

skip sequential access: Keyed sequential retrieval or storage of records here and there throughout a data set, skipping automatically to the desired record or collating position for insertion: VSAM scans the sequence set to find a record or a collating position.

stage: (Verb.) To transmit data from a mass storage volume to a direct-access storage staging drive.

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

user catalog: A catalog used in the same way as the master catalog, but optional and pointed to by the master catalog, and also used to lessen the contention for the master catalog and to facilitate volume portability.

User catalog connector: (*See* catalog connector.

vertical pointer: A pointer in an index record of a given level that gives the location of an index record in the next lower level or the location of a control interval in the data set controlled by the index.

Volume entry: A catalog entry that describes a volume owned by the catalog. All VSAM data spaces on the volume are described in the volume entry.

INDEX

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

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

123

OS/VS2 Access Method Services (File No. S370-30) Printed in U.S.A. GC26-3841-0

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

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

GC26-3841-0

Fold and Staple

First Class Permit
Number 439
Palo Alto, California

Business Reply Mail

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

Postage will be paid by:

IBM Corporation
System Development Division
LDF Publishing—Department J04
1501 California Avenue
Palo Alto, California 94304

Fold and Staple

OS/VS2 Access Method Services (File No. S370-30) Printed in U.S.A. GC26-3841-0

IBM®

nternational Business Machines Corporation
)ata Processing Division
133 Westchester Avenue, White Plains, New York 10604
U.S.A. only)

3M World Trade Corporation
21 United Nations Plaza, New York, New York 10017
International)

OS/VS2 Access Method Services
GC26-3841-0

Reader's Comment Form

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

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

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

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

PUT—Write Next Logical Record (QSAM)

The PUT macro instruction causes the system to write a record in a sequential data set. Various modes are available and are specified in the DCB macro instruction. In the locate mode, the address of an area within an output buffer is returned in register 1 after the macro instruction is executed. The user should subsequently construct, at this address, the next sequential record or record segment. The move mode of the PUT macro instruction causes a logical record to be moved into an output buffer. In the data mode, which is available only for variable-length spanned records, the PUT macro instruction moves only the data portion of the record into one or more output buffers. When the substitute mode is specified, the macro transfers ownership of a work area containing a record to the control program. In return, the ownership of a buffer segment is transferred to the user, for use as a work area. There is no movement of data in storage.

The records are blocked by the control program (as specified in the data control block) before being placed in the data set. For undefined-length records, the DCBLRECL field determines the length of the record that is subsequently written. For variable-length records, the DCBLRECL field is used to locate a buffer segment of sufficient size (locate mode), but the length of the record actually constructed is verified before the record is written (the output block can be filled to the maximum if, before issuing the PUT macro, DCBLRECL is set equal to the record length). For variable-length spanned records, the system segments the record according to the record length, buffer length, and amount of unused space remaining in the output buffer. The smallest segment created will be 5 bytes, 4 for the segment descriptor word plus one byte of data.

If the ASCII translation routines are included when the operating system is generated, translation can be requested by coding LABEL=(,AL) or (,AUL) in the DD statement, or it can be requested by coding OPTCD=Q in the DCB macro instruction or DCB subparameter of the DD statement. When translation is requested, all QSAM records whose record format (RECFM operand) is F, FB, D, DB, or U are automatically translated from EBCDIC code to ASCII code. For translation to occur correctly, all output data must be in EBCDIC code; any EBCDIC character that cannot be translated into an ASCII character is replaced by a substitute character.

The PUT macro instruction is written as follows:

[*symbol*]	**PUT**	*dcb address* [,*area address*]

dcb address—RX-Type Address, (2-12), or (1)
The *dcb address* operand specifies the address of the data control block for the data set opened for output.

area address—RX-Type Address, (2-12), or (0)
The *area address* operand specifies the address of an area that contains the record to be written (move or data mode), or it specifies the address of an area to be exchanged for a buffer (substitute mode). The move, locate, data, or substitute mode can be used with QSAM, but they must not be mixed within the specified data control block. If the *area address* operand is omitted in the move, data, or substitute mode, the system assumes that register zero contains the area address. The following describes the operation of the four modes:

Locate Mode: If the locate mode is specified, the *area address* operand must be omitted. The system returns the address of the next available buffer in register 1; this is the buffer into which the next record is placed.

When variable-length spanned records are used and a record area has been provided for a logical record interface (BFTEK=A has been specified in the data control block or a BUILDRCD macro instruction has been issued), the address returned in register 1 points to an area large enough to contain the maximum record size (up to 32,756 bytes). The system segments the record and writes all segments, providing proper control codes for each segment. If, for variable-length spanned records, an area has not been provided, the actual length remaining in the buffer will be returned in register 0. In this case, it is the user's responsibility to segment the records and process them in terms of record segments. The record or segment is not written until another PUT macro instruction is issued for the same data control block. The last record is written when the CLOSE macro instruction is issued.

When a PUT macro instruction is used in the locate mode, the address of the buffer for the first record or segment is obtained by issuing a PUT macro instruction; QSAM returns the address of the buffer, but the record is not written until the next PUT macro instruction is issued.

Move Mode: If the move mode has been specified in the data control block, the *area address* operand specifies the address of the area that contains the record to be written. The system moves the record to an output buffer before control is returned. The address of the output buffer is returned in register 1 (this action provides compatibility with substitute mode operations, and makes it possible for the problem program to be used in instances where substitute mode is requested but cannot be supported by the system).

Data Mode: If the data mode is specified in the data control block (data mode can be specified for variable-length spanned records only), the *area address* operand specifies the address of an area in the problem program that contains the data portion of the record to be written. The system moves the data portion of the record to an output buffer before control is returned. The user must place the total data length in the DCBPRECL (not DCBLRECL) field of the data control block before the PUT macro instruction is issued.

Substitute Mode: If the substitute mode is specified in the data control block, the *area address* operand specifies the address of an area in the problem program that contains the next record to be written. The area is exchanged for an empty buffer. The address of the empty buffer is returned in register 1.

PUT Routine Exit

The error analysis (SYNAD) routine is given control if an output operation could not be completed satisfactorily. If the output operation could not be completed satisfactorily, the error analysis (SYNAD) routine is given control after the next PUT instruction is issued. The contents of the registers when the error analysis routine is given control are described in Appendix A.

PUTX—Write a Record from an Existing Data Set (QISAM and QSAM)

The PUTX macro instruction causes the control program to return an updated record to a data set (QISAM and QSAM) or to write a record from an input data set into an output data set (QSAM only). There are two modes of the PUTX macro instruction. The output mode (QSAM only) allows writing a record from an input data set on a different output data set. The output data set may specify the spanning of variable-length records, but the input data set must not contain spanned records.

The update mode returns an updated record to the data set from which it was read. The logical records are blocked by the control program, as specified in the data control block, before they are placed in the output data set. The control program uses the length specified in the DCBLRECL field as the length of the record currently being stored. Control is not returned to the user's program until the control program has processed the record.

The PUTX macro instruction can be used only in the output mode for SYSIN or SYSOUT data sets.

The PUTX macro instruction is written as follows:

[*symbol*]	**PUTX**	*dcb address* [,*input dcb address*]

dcb address—RX-Type Address, (2-12), or (1)
The *dcb address* operand specifies the address of the data control block for a data set opened for output.

input dcb address—RX-Type Address, (2-12), or (0)
The *input dcb address* operand specifies the address of a data control block opened for input. The PUTX macro instruction can be used for the following modes:

Output Mode: This mode is used with QSAM only. The *input dcb address* operand specifies the address of the data control block opened for input. If this operand is omitted, the system assumes that register 0 contains the *input dcb address.*

Update Mode: The *input dcb address* operand is omitted for update mode.

PUTX Routine Exit

The error analysis (SYNAD) routine is given control if the operation is not completed satisfactorily. The contents of the registers when the error analysis routine is given control are described in Appendix A.

OS/VS2 Access Method Services
GC26-3841-0

Reader's Comment Form

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

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

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

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

Fold and Staple

First Class Permit
Number 439
Palo Alto, California

Business Reply Mail

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

Postage will be paid by:

IBM Corporation
System Development Division
LDF Publishing—Department J04
1501 California Avenue
Palo Alto, California 94304

Fold and Staple

BM
®

ternational Business Machines Corporation
ta Processing Division
33 Westchester Avenue, White Plains, New York 10604
.S.A. only)

M World Trade Corporation
1 United Nations Plaza, New York, New York 10017
ternational)

OS/VS2 Access Method Services (File No. S370-30) Printed in U.S.A. GC26-3841-0

OS/VS2 Access Method Services (File No. S370-30) Printed in U.S.A. GC26-3841-0

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

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