GC26-3841-1
File No. S370-30

Systems

OS/VS2 Access Method Services

Release 3.7

IBM

Second Edition (March 1976)

This edition replaces the previous edition (numbered GC26-3841-0) and makes it obsolete. This edition also replaces and makes obsolete *OS/VS2 Independent Component: Access Method Services,* GC26-3843-0.

This edition applies to Release 3.7 of OS/VS2 and to all subsequent releases of that system unless otherwise indicated in new editions or technical newsletters.

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. Because the technical changes in this editon are extensive and difficult to localize, they are not marked by vertical lines in the left margin; the entire edition should be carefully reviewed.

Information in this publication is subject to significant change. Any such changes will be published in new editons or technical newsletters. Before using the publication, consult the latest *IBM 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, General Products Division, Programming Publishing—Department J57, 1501 California Avenue, Palo Alto, California 94304. All comments and suggestions become the property of IBM.

© Copyright International Business Machines Corporation 1975, 1976

ABOUT THIS BOOK

This book describes the use of Access Method Services commands, a group of utility functions vital to Virtual Storage Access Method (VSAM). This publication provides VSAM information required to use Access Method Services to establish and maintain data sets. For information on VSAM data set and catalog format and structure see *Planning for Enhanced VSAM under OS/VS,* GC26-3842. For information on the use of VSAM macro instructions, VSAM optimization options and various VSAM algorithms affecting performance, see *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide,* GC26-3838.

In this manual, any references made to an IBM program product are not intended to state or imply that only IBM's program product may be used; any functionally equivalent program may be used instead. This manual has references to the following IBM program products:

RACF - Resource Access Control Facility
Program Number 5740-XXH

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 and the use of Time Sharing Option (TSO) and the 3850 Mass Storage System with VSAM and Access Method Services. Everyone who intends to use Access Method Services should first read this chapter.
- "Data Security and Protection," which describes the recovery features of VSAM that are specified as options in Access Method Services commands.
- "Creating and Cataloging Objects," which describes the use of the DEFINE command to define catalogs, data spaces, VSAM data sets, alternate indexes, paths, and page spaces, and to catalog generation data groups, aliases and nonVSAM data sets.
- "Building an Alternate Index," which describes the use of the BLDINDEX command to build an alternate index. An alternate index allows you to access a data set with a new set of key values.
- "Modifying Catalog Information," which describes the use of the ALTER command to modify attributes in catalog entries.
- "Displaying Catalog Information," 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 an OS Catalog's Entries to VSAM Catalog Entries," which describes the use of the CNVTCAT command to convert OS catalog entries to VSAM catalog entries and merge them into a VSAM catalog.
- "Restoring a Cluster's End-of-File Values," which describes the use of the VERIFY command to ensure that the true end-of-file is reflected in the catalog.
- "Restoring Catalog Entries After System Failure," which describes the use of the LISTCRA, EXPORTRA, and IMPORTRA commands to retrieve the contents of VSAM data sets after a VSAM catalog is damaged.

- "Copying and Printing," which describes the use of the REPRO command to copy and to reorganize data, and to convert data from indexed-sequential or sequential organization to VSAM organization and from VSAM organization to sequential organization, and to copy catalogs; and the use of the PRINT command to print VSAM and nonVSAM data sets.
- "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, SET, and PARM commands to control command execution and to specify diagnostic aids and printed-output options.
- "Appendix A: Examples of Jobs Using Access Method Services Commands," which describes how two or more Access Method Services commands can be used together to accomplish a task.
- "Appendix B: Interpreting LISTCAT Output Listings," which provides information on the structure of LISTCAT output and shows sample output.
- "Appendix C: JCL 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.
- "Appendix D: Interpreting LISTCRA Output Listings," which describes the format and uses of LISTCRA output.
- "Appendix E: Sample Output from CHKLIST," which shows a CHKLIST listing and explains its contents.
- "Appendix F: Command Parameters Summary," which sumarizes each Access Method Services command with a table that shows each parameter, its abbreviation, its default value (if any), and an example of its use. The command summaries are grouped together in an appendix to allow you to remove those pages and use them as a quick reference guide for Access Method Services commands.
- "Appendix G: Invoking Access Method Services from a Problem Program," which describes how Access Method Services may be dynamically invoked from a user's problem program.
- "Appendix H: Making the Master Catalog Recoverable," which describes how to apply the recoverable attribute to the VSAM master catalog, changing it from nonrecoverable to recoverable.
- "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:

- *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide,* GC26-3838, which describes VSAM optimization options and various VSAM algorithms affecting performance, macro instructions used to

process VSAM data sets, and 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.

- *Planning for Enhanced VSAM under OS/VS,* GC26-3842, which includes information on the structure and contents of a VSAM Catalog and data sets.
- *OS/VS Utilities,* GC35-0005, which describes the utility programs available for use with nonVSAM data sets on OS/VS2 systems.
- *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/VS2 JCL,* GC28-0692, which describes the JCL parameters referred to in this publication.
- *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.
- *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.
- *OS/VS2 System Programming Library: System Generation Reference,* GC26-3792, which describes the use of the DATASET macro to create a master catalog.

Related Publications

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

- *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; and processing an index as data.
- *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/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 user.
- *OS/VS2 Using OS Catalog Management with the Master Catalog: CVOL Processor,* GC35-0010, which describes the use of control volumes.
- *OS/VS2 Independent Component: Virtual Storage Access Method (VSAM) Logic,* SY26-3846, which describes the functional operation of VSAM (Record Management, Open, Close, End-of-Volume, and Control Block Manipulation commands).
- *OS/VS2 Independent Component: Catalog Management Logic,* SY26-3847, which describes the functional operation of VSAM Catalog Management.
- *OS/VS2 Independent Component: Access Method Services Logic,* SY26-3845, which describes the functional operation of Access Method Services programs.
- *OS/VS2 MVS Resource Access Control Facility (RACF) General Information Manual.*

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 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 ␣.
- **Boldface** type indicates the exact characters to be entered. Such items must be entered exactly as illustrated (in upper case, except in TSO).
- *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

OS/VS2 MVS Data Management (VS2.03.808)

Extended CVOL Support

In OS/VS2 MVS Release 2, the VSAM catalog replaced the system master catalog and support of OS CVOLs was limited to a subset of what had been provided in OS/VS2 SVS Release 1. Extended CVOL support provides a CVOL function that is equivalent to SVS while retaining the VSAM master catalog as the only system master catalog. This support required that changes be made to the section "VSAM's Use of Catalogs" and the chapter "Converting an OS Catalog's Entries to VSAM Catalog Entries."

Resetting a Catalog

A new Access Method Services command, RESETCAT, has been added to support recoverable catalogs. The new command will allow you to reset your VSAM catalog to the level of its owned volumes. That is, if the catalog or any of its volumes become damaged, RESETCAT can be used to synchronize a catalog to a restored (not current) level version of a volume. Or, if the catalog has to be restored or reloaded, its entries can be changed to reflect the current level of its volumes with RESETCAT. The new sections "Resetting a Catalog" and "RESETCAT" have been added to describe and parameterize this command.

OS/VS2 MVS Supervisor Performance #2 (VS2.03.807)

Auxiliary Storage Manager (ASM) Redesign

SWAP space data sets can now be defined and preformatted. Also, a new integrity attribute has been added to both SWAP and PAGE spaces.

Resource Access Control Facility (RACF)

Changes are included to support protection of VSAM data sets through RACF.

Release 3.7

LISTCAT Format

The format of LISTCAT output has been modified to improve readability.

Miscellaneous Topics

Two new appendixes have been added. They are: Appendix G: Invoking AMS from a Problem Program, and Appendix H: Making a Master Catalog Recoverable.

All AMS command formats, parameter descriptions, and examples have been moved to a new chapter: Command Format. In the new chapter, commands are arranged in alphabetical order.

Information relating to such topics as VSAM Data Structure, How Data is Physically Stored, Types of VSAM Data Sets, VSAM Catalog Structure, Performance Optimization, and Data Integrity has been removed from this book. The information removed can be found in other VSAM publications.

Independent Component Release

Following is a summary of major changes to Access Method Services for the release of enhanced VSAM as an independent component under OS/VS2.

Alternate Index

The alternate index allows you to access data records using a key field other than the key field established when the data set was defined (for key-sequenced VSAM data sets), and to access the data records of an entry-sequenced data set with a key field.

Reusable Data Set

The reusable data set is an attribute that can be applied to VSAM entry sequenced, key sequenced, relative-record and alternate index data sets. This attribute allows the data set to be used for temporary storage of data. Whenever a reusable VSAM data set is opened, its high-used RBA can be reset to zero.

Spanned Records

Data records in VSAM data sets can be larger than one control interval, and are contained in two or more control intervals.

Relative-Record Data Set

The relative-record data set is a VSAM data set whose records are fixed-length and are sequenced according to record number.

VSAM Volume Cleanup

The VSAM volume cleanup process allows you to erase VSAM data from a volume's VTOC and reset the VTOC's format-4 DSCB to show system ownership of the volume. The process is to be used when a system or device failure occurs that damages the volume's information in its VSAM catalog (that is, the VSAM catalog that owns the volume).

VSAM Catalog Cleanup

The VSAM catalog cleanup process allows you to delete a VSAM catalog without first deleting each of its cataloged entries. You can use this function to remove an unwanted catalog from your system. (Certain safeguards prevent indiscriminate use of the function.)

Record Replacement (During REPRO)

You can merge two data sets so that, when duplicate records are encountered, the older record (that is, the one in the target data set) is replaced with the newer record (the one in the source data set).

Catalog Recovery

Three new Access Method Services commands support recoverable catalogs: LISTCRA, EXPORTRA, and IMPORTRA. The catalog recovery function allows you to repair damaged entries in a VSAM catalog when a system failure occurs.

Catalog Recovery Area

Copies of each catalog record are maintained (on the volume containing the catalog entry's data records) in an area called the catalog recovery area. The catalog recovery area is built and maintained only for recoverable catalogs, and is used to recover from damaged catalog entries.

Exception Exit Routine

You can write an I/O error handling routine, called the exception exit routine, that is tailored for a cluster's or alternate index's data or index component, and which is called before the user program's SYNAD routine gets control.

Miscellaneous Topics

The KEYS and RECORDSIZE parameters of the DEFINE CLUSTER command have been modified so that a default value exists for them. The ALTER command's parameter set now includes the KEYS and RECORDSIZE parameters.

You can import an exported object into a predefined object.

You can specify blocksize for exported and imported objects.

The *dname* subparameter of the CATALOG parameter has been omitted from the syntax of those commands that no longer require its specification. If you specify *dname,* it will be ignored.

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 (MSS)" 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. The sections "Updating a Backup Catalog," and "VSAM Volume Cleanup" in the chapter "Data Security and Protection" 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 Accessibility of Secondary Extents" in the chapter "Data Security and Protection" 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.

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.

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. 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 an OS Catalog's Entries to VSAM 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 "Creating and Cataloging Objects" 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 "Creating and Cataloging Objects" 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 the chapters on these commands for information on the changes.

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 "Creating and Cataloging Objects" for information about defining a page space. The ALTER, DELETE, and LISTCAT commands have been expanded to support page spaces; see the chapters on these commands for information on the changes.

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 "Modifying Catalog Information" 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 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 "Displaying Catalog Information," "Modifying Catalog Information," 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 "Displaying Catalog Information" 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
Alter	the information in a catalog entry	ALTER
Attach	a user catalog to the master catalog	DEFINE USERCATALOG or IMPORT CONNECT
Backup	a VSAM data set	EXPORT
	a nonVSAM data set	REPRO
	a VSAM catalog	REPRO
Build	an alternate index	DEFINE ALTERNATEINDEX and BLDINDEX
	a generation data group	DEFINE GENERATIONDATAGROUP
Catalog	a VSAM data set	DEFINE CLUSTER
	a nonVSAM data set	DEFINE NONVSAM
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 CONNECT
	an OS/VS CVOL to a master catalog	DEFINE NONVSAM
Convert	a data set to VSAM format	REPRO
	a VSAM data set to sequential format	REPRO
	OS/VS CVOL entries to VSAM catalog entries	CNVTCAT
Copy	a data set or catalog	REPRO
Create	a backup copy of a data set	REPRO or EXPORT
	a catalog	DEFINE USERCATALOG
	an alias for a nonVSAM data set	DEFINE ALIAS
	an alias for a VSAM data set	DEFINE PATH
	a VSAM data set	DEFINE CLUSTER
	an alternate index	DEFINE ALTERNATEINDEX and BLDINDEX
	the relationship between an alternate index and its base cluster	DEFINE PATH
	a generation data group	DEFINE GENERATIONDATAGROUP
	a generation data set	DEFINE NONVSAM
	a page space	DEFINE PAGESPACE
	a VSAM data space	DEFINE SPACE
	a catalog entry for a nonVSAM data set	DEFINE NONVSAM
Define	(see above entries for Create)	
Delete	a user catalog	DELETE USERCATALOG
	an alias	DELETE ALIAS
	an alternate index and its paths	DELETE ALTERNATEINDEX
	a path	DELETE PATH
	any type of catalog entry	DELETE
	a VSAM data set	DELETE CLUSTER

Figure (Part 1 of 3). Tasks and Commands

Operation		Command
Delete (continued)	a VSAM data space	DELETE SPACE
	a generation data group	DELETE GENERATIONDATAGROUP
	a nonVSAM data set	DELETE NONVSAM
	a page space	DELETE PAGESPACE
Disconnect	a user catalog	EXPORT DISCONNECT
Display	a catalog's contents	LISTCAT
	a catalog recovery area's contents	LISTCRA
Enter	a nonVSAM data set in a catalog	DEFINE NONVSAM
Export	a user catalog	EXPORT DISCONNECT
	a VSAM data set	EXPORT
Import	a user catalog	IMPORT CONNECT
	a VSAM data set	IMPORT
List	a password	LISTCAT
	a data set's contents	PRINT
	contents of a catalog or of a catalog entry	LISTCAT
	contents of a catalog recovery area	LISTCRA
	tapes mounted at a checkpoint	CHKLIST
Load	records into a data set	REPRO
	an alternate index	BLDINDEX
Modify	a data set's description in the catalog	ALTER
Move	a catalog to another system	EXPORT DISCONNECT and IMPORT CONNECT
	a VSAM data set to another system	EXPORT and IMPORT
	a nonVSAM data set to another system	REPRO
Password Protect	establish for a VSAM data set	DEFINE CLUSTER
	alternate index	DEFINE ALTERNATEINDEX
	path	DEFINE PATH
	catalog	DEFINE USERCATALOG or DEFINE MASTERCATALOG
	page space	DEFINE PAGESPACE
	modify an existing password, or add a password to an existing catalog entry	ALTER
	delete a password	ALTER
	list passwords	LISTCAT
Print	a data set	PRINT
Recover	from a processing-program failure	VERIFY
	from a catalog failure	LISTCRA EXPORTRA and IMPORTRA RESETCAT
Release	a user catalog from the master catalog	EXPORT DISCONNECT

Figure 1 (Part 2 of 3). Tasks and Commands

Operation		Command
Rename	a data set	ALTER
Reproduce	a VSAM or nonVSAM data set	REPRO
Restore	a data set's end-of-file information a catalog entry and/or the contents of its object	VERIFY LISTCRA EXPORTRA and IMPORTRA RESETCAT
Uncatalog	a data set	DELETE
Unload	a data set or catalog	REPRO
Verify	a VSAM data set's end-of-file indicators	VERIFY

Figure 1 (Part 3 of 3). Tasks and Commands

INTRODUCTION

Access Method Services is a service program that is used with VSAM (Virtual Storage Access Method) to establish and maintain catalogs and data sets. If you plan on using 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 or catalog, convert an OS CVOL's entries to VSAM catalog entries, and make a data set portable from one operating system to another.

The user invokes Access Method Services functions by issuing a command and specifying its parameters. The user can execute the IDCAMS program and include the command and its parameters as input to the program. The user 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 as though they were TSO commands. TSO users cannot issue modal commands.

There are Access Method Services commands for:

- Defining, altering, and deleting data sets
- Listing catalog entries
- Copying and printing data sets
- Constructing alternate indexes for data sets
- Moving catalogs and data sets from one operating system to another
- Aiding in recovery from damage to data sets or catalogs
- Converting an OS catalog's entries to VSAM catalog entries
- 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.

Functional Commands

The functional commands are:

- ALTER, which is used to alter previously defined catalog entries.
- BLDINDEX, which constructs alternate indexes for existing data sets.
- CHKLIST, which is used to identify tape volumes mounted when a checkpoint was taken.
- CNVTCAT, which is used to convert entries in an OS CVOL (control volume) into entries in a VSAM catalog.
- DEFINE, which is used to create catalog entries for data sets, catalogs, and space that VSAM is to allocate from.
- DELETE, which is used to delete 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.
- EXPORTRA, which retrieves VSAM data sets and catalog entries that are no longer accessible from a VSAM catalog.
- 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.
- IMPORTRA, which returns VSAM data sets and catalog entries to an accessible condition.
- LISTCAT, which is used to list catalog entries.
- LISTCRA, which is used to diagnose suspected problems in VSAM catalogs.
- PRINT, which is used to print VSAM, ISAM, or SAM data sets.
- 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.
- RESETCAT, which synchronizes a catalog to the level of its owned volumes.
- 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.

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

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

All functional commands except CHKLIST can be used when you execute Access Method Services at a TSO terminal.

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 denote the beginning and ending of a command 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 Appendix F.

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. In Access Method Services, positional parameters are always required. 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. When the subparameters are positional, the first subparameter is required and the second is optional (that is, you cannot specify the second subparameter without also specifying the first).

Many of the keywords may be abbreviated. The abbreviations permitted are listed in Appendix F. 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. Positional parameter 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*)) ...)

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 only by blanks or a comment. 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 DD 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 want *WORD*/ to be the password, you 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 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 blanks 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 TEST (- /*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 second 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. It 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 must exist and be cataloged. The catalog that contains the entry must be either a user catalog identified with a JOBCAT or STEPCAT DD statement, the master catalog, or a user catalog whose name or alias is the first name of the qualified data-set name.

In order to dynamically allocate a volume, it must already be mounted as permanently resident or reserved. The PRIVATE and PUBLIC use attributes should be carefully considered when you mount a volume.

When a JCL DD statement is used to identify a VSAM 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. When specifying an alternate output data set, you should not 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) one command at a time, then calls the appropriate functional routines to perform all services required by that command.

There are three ways you can invoke the Access Method Services processor:

- As a job or jobstep
- From a TSO terminal
- From within your own program

If you want more details on the structure and operation of Access Method Services in an OS/VS2 system, see *OS/VS2 Independent Component: Access Method Services Logic.*

As a Job or Jobstep

You can use job control language (JCL) statements to invoke the Access Method Services processor. The examples that illustrate each command's use invoke the processor with JCL statements. You identify the Access Method Services processor with PGM=IDCAMS.

```
//YOURJOB  JOB  YOUR INSTALLATION'S JOB-ACCOUNTING DATA
//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 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 can invoke the Access Method Services processor 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 that can be used when processing the jobstep. If user catalogs are identified with JOBCAT and STEPCAT DD statements, only the catalog(s) identified with the STEPCAT DD statement and the master catalog are used with the jobstep. 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," above.

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 builds the appropriate interface information and invokes Access Method Services.

You can issue one command at a time. Access Method Services processes the command completely before TSO allows you to continued processing. The following Access Method Services commands are supported in a TSO environment:

ALTER
BLDINDEX
CNVTCAT
DEFINE -
  ALIAS
  ALTERNATEINDEX
  CLUSTER
  GENERATIONDATAGROUP
  NONVSAM
  PAGESPACE
  PATH
  SPACE
  USERCATALOG
DELETE
EXPORT
EXPORTRA
IMPORT
IMPORTRA
LISTCAT
LISTCRA
PRINT
REPRO
RESETCAT
VERIFY

The command you issue is processed immediately. You cannot issue modal commands (that is, the IF, THEN, ELSE, DO, END, PARM, and SET commands) that 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 each appropriate parameter.

From a User's Program

A processing program can invoke Access Method Services with the ATTACH, LINK or LOAD and CALL macros. Before the program issues the invoking macro, 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 Appendix G.

Authorized Program Facility

The authorized program facility (APF) limits the use of sensitive system services and resources to authorized system and user programs. See "Authorized Program Facility (APF)" in *OS/VS2 System Programming Library: Supervisor,* for information about program authorization.

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

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

The following situations could cause the APF authorization of Access Method Services to be violated:

- An Access Method Services module is loaded from an unauthorized library.
- A user security verification routine (USVR) is loaded from an unauthorized library during Access Method Services processing.
- An Exception Exit routine is loaded from an unauthorized library during Access Method Services processing.

Since APF authorization is established at the job step task level, Access Method Services will not be authorized if invoked by an unauthorized problem program or an unauthorized Terminal Monitor Program (TMP). The restricted functions performed by Access Method Services that cannot be requested in an unauthorized state are:

- REPRO—the copy catalog or catalog unload/reload facility
- PRINT—when the object to be printed is a VSAM catalog.
- LISTCRA
- EXPORTRA
- RESETCAT

If the above functions are required and Access Method Services is invoked from a problem program or a TSO Terminal Monitor Program, the invoking program must be authorized.

VSAM Data Sets

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, entry sequence, or record-number 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.

Records in a *relative-record data set* are stored without respect to their contents and to their sequence of entry. Their sequence is determined by the user-specified record number that identifies the record's position in the data set. Each record is fixed-length and is stored in a *slot*. Records can be inserted between other records (that is, in an empty slot between two filled slots), can be added to the end of the data set, or can replace records within 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 set and its index make up a cluster. An entry-sequenced or relative record data set is also defined as a cluster, even though it does not have an index.

Apart from the primary index, one or more *alternate indexes* can be built over a single key-sequenced or entry-sequenced data set. Each alternate index accesses the data records of a given data set via a different key field (the alternate key) within these records. A VSAM cluster that contains data records pointed to by an alternate index is called a *base cluster*.

In order to gain access to a base cluster via an alternate index, you must define a *path* between the alternate index and the base cluster. When a path is defined, you must specify the name of the alternate index which is to be considered as the entry for the path. The termination of the path is the base cluster to which the alternate index is related. The path, which is an entry in the catalog, has a name of its own which always refers to the alternate index and the related base cluster as a pair.

In addition to data sets, VSAM accesses catalogs. All data—ordinary user data, indexes, and catalogs—is physically stored and manipulated in the same way.

Refer to *Planning for Enhanced VSAM Under OS/VS* for more information on the structure of VSAM data sets and how data is physically stored.

VSAM Volume Ownership

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 have a maximum of 16 extents.

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, alternate indexes, 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 (due to damage that resulted from a system or hardware failure), you can reset the ownership bit by using the ALTER REMOVEVOLUMES command (see "Modifying Catalog Information").

The VTOC contains the name of each VSAM data space on the volume, and might contain VSAM-generated names for the data and index components of a cluster, alternate index, or page space. The format-1 DSCB is identified with the object's entryname. When you name the data or index component of a VSAM data set, alternate index, or page space, and when the object or its component is in its own data space (that is, it was defined with the UNIQUE attribute), the name of the component identifies the data space. Otherwise, VSAM generates a name for the data space. The name generated by VSAM has the following format:

- For a data space containing suballocated VSAM objects, the VSAM generated name is:

  Z999999n.VSAMDSPC.Taaaaaaa.Tbbbbbbb

where:

- n=2 if no catalog resides in the data space
- n=4 if a user catalog resides in the data space, or if the master catalog resides in the data space and the master catalog was created under OS/VS2 Release 2 or a later release
- n=6 if the master catalog resides in the data space
- aaaaaaabbbbbbb is the timestamp value

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

  Tbbbbbbb.VSAMDSET.DFDyyddd.Taaaaaaa.Tbbbbbbb

  where:

  - yyddd is the date (year and Julian day)
  - aaaaaaabbbbbbb is the timestamp value

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

Each volume owned by a VSAM catalog contains two timestamps that are written in the VTOC when the volume is first cataloged. Both timestamps are updated but only one (the second) is checked. The first is maintained for compatibility with earlier VSAM releases. The volume timestamps are updated as follows:

- For a volume owned by a recoverable VSAM catalog, each time the catalog's Volume record is modified.
- For a volume owned by a nonrecoverable VSAM catalog, each time space is allocated, extended or scratched by VSAM. When the volume is mounted, the system compares the second timestamp on the volume to the timestamp in the catalog owning the volume. If the volume's timestamp is earlier than the catalog's timestamp, the volume is considered down-level. Access Method Services will not normally open a data set on a down-level volume.

Some of the implications of VSAM volume ownership are:

- All VSAM clusters, alternate indexes, page spaces, and data spaces on a volume must be cataloged in the catalog that owns the volume. Only one VSAM catalog can own the volume.
- VSAM volume ownership does not affect nonVSAM data sets that reside on the volume. NonVSAM data sets can exist on a volume owned by a VSAM catalog, and can (*but should not*) be cataloged in a catalog that doesn't own the volume. Since OS control volumes (CVOLs) are considered nonVSAM data sets, an OS CVOL can also exist on a volume owned by a VSAM catalog.
- In order to release a volume from ownership by a VSAM catalog, you must delete all VSAM objects that reside on the volume. The catalog also contains a volume entry, which describes the volume and its VSAM data spaces. After deleting the VSAM objects, you issue the DELETE SPACE command. The DELETE SPACE command deletes the VSAM data spaces

on the volume, removes the volume entry from the catalog, and revises the format-1 and format-4 DSCBs in the volume's VTOC.

- A user catalog cannot exist on a volume that contains active page spaces, paging data sets (that is, system data sets that are paged into and out of the CPU's virtual storage), or the SYS1.STGINDEX. These objects are cataloged in the master catalog and, therefore, the volume(s) on which they reside are owned by the master catalog.

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 VSAM catalogs: master catalogs and user catalogs. In addition, OS CVOLs (control volumes) can be attached to a VSAM catalog. In a VS2 system, the VSAM master catalog is the system's primary catalog and is required. Any number of VSAM user catalogs are optional. The system's nucleus contains a pointer to the VSAM master catalog, which in turn points with catalog connector entries to any VSAM user catalogs. The OS CVOLs are cataloged in the master catalog as nonVSAM data sets with the name "SYSCTLG.*catname*."

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. VSAM "ownership" of a volume by a catalog is established the first time VSAM space is allocated on that volume. The DEFINE with the UNIQUE attribute (of a cluster, alternate index, or page space), DEFINE USERCATALOG, or DEFINE SPACE commands establish such ownership. All subsequent VSAM data sets defined on the volume are required to be defined in the same catalog (that is, the catalog that owns the volume).

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), whether the name of the data set is a qualified entryname, and on whether the catalog is to be used to define a new entry or searched for existing entries. See "Order of Catalog Use" for an explanation of how a catalog is selected during the processing of an Access Method Services command. ("Order of Catalog Use" sections appear in the chapters that describe each of the following commands: DEFINE, BLDINDEX, ALTER, LISTCAT, and DELETE.)

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 2 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).

If you need information on the structure of a VSAM catalog or more details on the information contained in the catalog, refer to *Planning for Enchanced VSAM under OS/VS.*

The Master Catalog

In an OS/VS2 system, the VSAM master catalog is the system's primary catalog. When you generate the system (that is, when you use the SYSGEN procedure), you define the system's master catalog during the first step in stage 2 of SYSGEN. The name you specify for the catalog must be different from the name of the master catalog of the driving system. See *OS/VS2 System Programming Library: System Generation Reference* for details on the system generation process.

The master catalog does not have to reside on the IPL volume. The SYSCTLG member of SYS1.NUCLEUS, a nonVSAM data set on the IPL

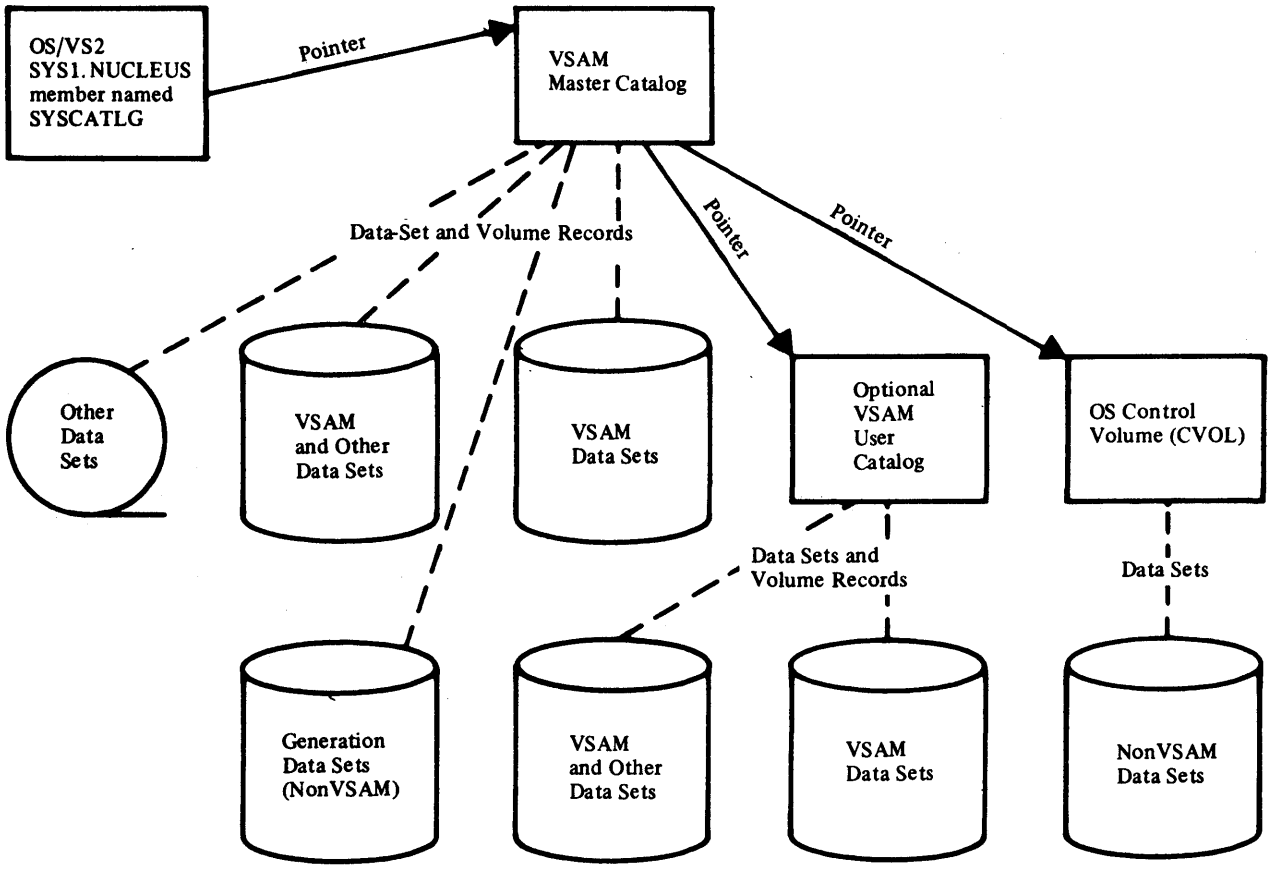


Figure 2. Catalog Relationships in an OS/VS2 System

volume, points to the volume containing the master catalog. The SYSCTLG member is referenced during IPL and NIP processing. It is recommended that the master catalog and VSAM objects cataloged in it not reside on the IPL volume—if the master catalog has to be recovered, then restoring the master catalog will not affect the IPL volume.

The master catalog is the only catalog referenced during NIP processing. Therefore, all system data sets, data sets entered in LNKLSTxx, paging data sets, and libraries specified in the JES 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 copy to another system. If your master catalog is recoverable, you can use the LISTCRA, EXPORTRA, IMPORTRA, and RESETCAT commands to repair damaged entries that result from system failure.

A master catalog cannot be used simultaneously as the VSAM master catalog for two OS/VS2 systems (that is, the master catalog cannot be shared between two systems as each system's master catalog). However, one system's master catalog can be used as a user catalog on another OS/VS2 system that includes VSAM when the catalog is on a shared direct-access device. When a catalog is used simultaneously by two or more systems, all of the catalog's volumes must be on shared direct-access devices. If you do this, take care to assign passwords to each of the catalog's page spaces and system data sets to prevent their accidental or unauthorized use.

Transporting User Catalogs

You can move a user catalog to another OS/VS system (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 its OS/VS system. The EXPORT command deletes the user catalog's connector entry in the master catalog. The catalog, its cataloged objects, and its volume(s) become unavailable to the old system.

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.

JOBCAT and STEPCAT DD Statements

User catalogs can be dynamically allocated. User catalogs can also be allocated directly, by using 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 that is to be available to all steps of a job. The STEPCAT DD statement identifies a catalog that is to be available to only one of the job steps. The STEPCAT catalog, when specified for a jobstep, is used instead of the JOBCAT catalog during that job step.

You can supply a JOBCAT or STEPCAT DD statement to allocate the catalog when:

- Your job processes a user catalog (that is, when your sequence of Access Method Services commands adds, modifies, lists, or deletes a VSAM catalog entry)
- Your job includes a space allocation request directed toward a volume that contains VSAM data spaces (that is, a DEFINE SPACE or DELETE SPACE command)
- Your job uses IEHDASDR to dump a volume owned by a user catalog

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

OS Control Volumes (CVOLs)

In addition to the VSAM master catalog, your system can include user (or private VSAM) catalogs and OS control volumes (CVOLs or private OS catalogs). The system's catalog structure can be organized with a master catalog only, or with a master catalog and any combination of one or more user catalogs and/or OS CVOLs.

The master catalog (the primary level) points to each user catalog and OS CVOL in the system. A VSAM user catalog cannot access via another user catalog or OS CVOL.

The VSAM master catalog provides tthe following benefits in your OS/VS2 system:

- Portability of catalogs between systems
- The capability of mounting, as needed, the volumes containing catalogs
- Minimization of the effect of errors or damaged entries on a catalog
- Performance improvements by reducing contention for a catalog

The following differences between VSAM user catalogs and OS CVOLs might affect your choice of having user catalogs or OS CVOLs on your system:

- You cannot catalog VSAM objects in an OS CVOL.
- You cannot use Access Method Services commands to catalog objects in, or modify existing entries of, an OS CVOL.
- You can convert an OS CVOL's entries to VSAM catalog entries when you use the CNVTCAT command, but there is no corresponding facility to convert VSAM catalog entries to OS CVOL entries.
- A user catalog requires at least twice as much space as a corresponding OS CVOL.
- User catalogs can be moved only to other OS/VS systems that include VSAM. OS CVOLs can be moved to MVT and OS/VS2 systems that might not include VSAM.

You can use the DEFINE NONVSAM command to define a CVOL (as though it were a nonVSAM data set) in the master catalog when you name it "SYSCTLG.qualifier". You can't use IEHPROGM to create or extend a CVOL.

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.

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.

  Some abbreviations which are acceptable to Access Method Services such as CYL, CYLINDER, REC, RECORD cannot be used under TSO because the abbreviations do not contain enough initial characters to make the keyword unique. TSO can't tell whether you mean CYLINDERS or CYLINDERFAULT when CYL or CYLINDER is used. Similarly, abbreviations REC and RECORD are ambiguous; TSO doesn't know if you mean RECORDS or RECORDSIZE.

- 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.

  Note: At a TSO terminal, the logon password is checked first before the user is prompted to supply a password for a cluster. Checking the logon password counts as one attempt to obtain a password. If the user has not specified ATTEMPTS in his DEFINE command, he has one attempt to supply the catalog's password because the default is two.
- The modal commands, used to control execution (IF-THEN-ELSE command sequence, DO-END command sequence, SET, and PARM), are 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.*

Mass Storage System (MSS)

The IBM 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 cannot be stored on a mass storage volume.

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.

Space for an object larger than one cylinder that is stored on a mass storage volume should be allocated in cylinders to optimize data transfer (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.

DATA SECURITY AND PROTECTION

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 accidental 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 catalog 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 data set represented by the catalog entry is used when it is connected to a user's program (OPEN), or disconnected (CLOSE), or when it reaches its upper RBA-boundary (End-of-Volume). 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, alternate indexes, paths, 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 the 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, and to alter or delete nonVSAM entries in it.
- *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. If

you do not define passwords for the catalog, no password checking will take place during operations on the data set's catalog entries.

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 be null.

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.

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 an 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 that the operation affects. A catalog's master level password will allow you to refer to all catalog entries. However, a protected cluster cannot be processed with the catalog's master password.
- Specification of a password where none is required is always ignored.

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.

Note: Using the TSO log-on password counts as one attempt.

Passwords for NonVSAM Data Sets

When you define a nonVSAM data set in a VSAM catalog, the data set is not protected with passwords in its catalog entry. 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.)* You use the PROTECT macro instruction to assign a password to the nonVSAM data set (for more details, see *OS/VS Data Management Services Guide* and *OS/VS2 System Programming Library: Data Management).*

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

User-Security-Verification Routine

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.

Resource Access Control Facility (RACF) Protection

Resource Access Control Facility (RACF) provides an optional software access control measure you can use in addition to or instead of passwords. Password protection and RACF protection can coexist for the same data set. When RACF protection is applied to a data set which is already password protected, password protection is bypassed and access is controlled solely through the RACF authorization mechanism. If a User-Security-Verification-Routine (USVR) exists, it will not be invoked for RACF-defined data sets.

Although passwords are ignored for a RACF indicated data set, they can still provide protection if the data set is moved to another system which does not have RACF protection.

RACF authorization checking is generally compatible with the password authorization checking scheme. This compatilbility includes the time at which the authorization check is made, the sources of authorization, and the level of authorization required. The RACF authorization levels of *alter*, *control*, *update*, and *read*, correspond to the password levels of *master*, *control*, *update*, and *read*. As an example, consider the authorization required to delete a VSAM data set. With passwords this requires *master* level authorization to the data set or the containing catalog. (The catalog is a second source of authorization in this case.) If both the data set and catalog were RACF protected, it would take *alter* level authorization to either the data set or catalog to delete the data set.

Refer to *OS/VS2 Resource Access Control Facility (RACF) General Information Manual* for more information.

Protecting User's Data

Sometimes, due to system or hardware failure, your data may be destroyed or made inaccessible to you. You should recognize the possibility of this occurring and take actions that will allow you to recover your data if such a catastrophe happens.

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

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 (without its catalog entry), a VSAM catalog, or a sequential data set. 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 catalog 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 "Protecting VSAM Catalogs," below, 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 or a relative-record 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 *OS/VS Virtual Storage Access Method (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.

Corrective Measures

When you discover that a catastrophe destroyed part or all of your on-line data, you can replace the destroyed data set with its back-up copy (if one exists):

- You can use the REPRO command with the REPLACE option to merge the back-up copy with the destroyed data. When only part of your data set is damaged, you can specify REPRO parameters that allow you to replace only the records in the damaged part of the data set.
- You can use the IMPORT command to totally replace a VSAM cluster whose back-up copy was built using the EXPORT command. The IMPORT command uses the back-up copy to replace the cluster's contents and catalog information.

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. If some of the space on a volume doesn't belong to a VSAM data space, OS/VS may allocate that space to a nonVSAM data set. If you follow a dump/restore recovery procedure (outlined later) you will have to fall back to a previous copy not only of the VSAM data and catalog, but also of the nonVSAM data.

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 would 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 the 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 may have occurred:

- Entries may 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 may 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 may 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 Extent" 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. DELETE USERCATALOG FORCE can be used 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.

Note:

- 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 and is not referenced by any VSAM objects.
- 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.

Changing a Volume's Serial Number

When you change the volume serial number of a volume that contains VSAM objects, the VSAM objects can no longer be located when the objects' catalog volume information is referenced (you can use the LABEL statement of the IEHDASDR program to change a volume's serial number, or "clip" the volume). You should use the following procedure to change the volume's serial number, to allow the VSAM objects on the volume to be located:

1. Issue the EXPORT PERMANENT command for each VSAM object (that is, for each alternate index and cluster) on the volume. When the volume also contains a VSAM catalog, issue an EXPORT DISCONNECT command to disconnect the user catalog from the master catalog. The EXPORT command copies each VSAM object and its catalog entry onto a movable volume. The object's entry in the user catalog is deleted.
2. Issue the DELETE command to delete all empty data spaces on the volume, and to delete the volume entry from the catalog. NonVSAM data sets are described in the volume's VTOC and aren't affected by the DELETE command.
3. When the volume contains an empty catalog (that is, the catalog only describes space on the volume), delete the catalog. If the empty catalog describes space on more than its own volume, you can delete the space on each volume first, then delete the catalog.
4. Execute the IEHDASDR program with the LABEL statement to change the volume's serial number.
5. Issue the DEFINE SPACE command to build a data space on the volume, to establish the VSAM catalog's ownership of the volume, and to build a volume entry that points to the volume with its new serial number. The data space you define should be large enough to contain all of the suballocated VSAM objects you removed from the volume during step 1.

(When you execute the IEHDASDR program, you might reorganize the nonVSAM data sets remaining on the volume, so that the space available for the VSAM data space is contiguous.) If the catalog was deleted in step 3, it must be redefined when beginning this step.

6. Issue the IMPORT command for each VSAM object that was removed from the volume during step 1, specifying the new volume information for the object.

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 entires 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 after the time it was unloaded to make the backup. 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 DELETE CLUSTER or DELETE PAGESPACE is specified, by deleting the cluster or pagespace entry.
- When DELETE 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).

Moving a Catalog to Another System

You can move a user catalog from your VS2 system to another system (that is, to an OS/VS system that includes VSAM) to make the objects cataloged in the user catalog available to programs running on the other system. The user catalog might own only the volume on which it resides. In this case, the volume, its catalog, and its contents (that is, VSAM clusters, nonVSAM data sets, and free space) can be moved as a single unit to another system. When the catalog owns more than one volume, each volume owned by the catalog must either be moved to the other system with the catalog or be deleted before the catalog is moved.

Caution: When you move a recoverable catalog to a system that doesn't support recoverable catalogs, the recoverability of the catalog is destroyed when any of the catalog's records are modified, deleted, or added.

You use the EXPORT command with the DISCONNECT option to remove a user catalog from its OS/VS system. The EXPORT command deletes the user catalog's connector entry in the system's master catalog. The EXPORT command also deletes all alias entries that point to the user catalog's connector entry. The catalog, its cataloged objects, and its volume(s) are now unavailable to users of the old system.

When the user catalog has aliases, you should take care to list the aliases before you issue the EXPORT command. You can use the LISTCAT command to list the catalog's aliases, which are in the master catalog. You can use the list of aliases to help you reconstruct 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 OS/VS system. The IMPORT command builds in the new system's master catalog a user-catalog connector entry that points to the user catalog's volume. You should issue a DEFINE ALIAS command to build an alias entry in the new system's master catalog for each of the user catalog's aliases.

Exporting a Catalog's Data Sets

While the primary purpose of the EXPORT command is to allow you to move data sets and user catalogs between systems, it can also be used to create backup copies of particularly sensitive data sets. The EXPORT command copies a data set and extracts the data set's definition from the catalog in which it is defined. If you use Access Method Services to periodically export a data set, you can redefine the catalog should it be lost, define a VSAM data space if the data set is to be suballocated, and import the data set into the new catalog. With export/import you can control the level of each data set individually. If you exported a data set for backup after each run, you wouldn't have to rerun any jobs to bring the backup copy up to date.

A single volume per user catalog is desirable when your system contains a large number of single-volume data sets.

Catalog Recovery

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

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

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

RESETCAT is another catalog recovery tool. If, for any reason, inconsistencies develop between your catalog and the CRAs of its owned volumes, you should consider RESETCAT as your recovery vehicle. RESETCAT is basically a synchronization process between a catalog and its associated CRA(s). If your catalog or any of its owned volumes becomes unusable, you can restore a backup volume. However, this will not solve your problems because inconsistencies will undoubtedly exist between your catalog and the CRAs of the volumes it owns. Either the catalog or the CRAs will be downlevel (entries will not precisely describe the data sets). RESETCAT provides the necessary synchronization facility to ensure consistency between the catalog and its volumes. RESETCAT confines its processing to the

catalog and CRAs; VSAM data sets are unaffected by this operation; no data movement takes place.

You can use the LISTCRA command either to list the contents of the CRA before you do selective recovery or to list the entries in the recovery area that are different from those in its associated catalog. (See the "Restoring Catalog Entries After System Failure" chapter for more information and examples.)

If you are presently using an OS/VS2 system without recoverable catalogs, see Appendix G for information on making your master catalog recoverable.

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

- Inaccessible data set
- Unusable catalog
- Inaccessible volume

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

Catalog Volume Records

*MISCOMPARES Message	Type	Cause	Severity
DATA SPACE EXTENTS	Mismatched data space group	A difference in the number, size, and/or location of VSAM data spaces.	Requires recovery of the entire volume.
		A difference in the number and/or location of extents for one or more data sets.	Requires recovery of the entire volume.
		Space has been extended or deleted.	Requires recovery of the entire volume.
DATA SET DIRECTORY	Mismatched data set directory	A difference in the names and/or number of data sets associated with this volume.	Requires recovery of the entire volume.

VSAM Object Records

*MISCOMPARES Message	Type	Cause	Severity
CATALOG ENTRY HAS DIFFERENT NAME	Mismatched name	The catalog was restored, or the volume containing the data set was restored. As a result the record in the catalog pointed to by the CRA record is no longer for the same object.	Requires data set recovery.
VOLUME OR KEYRANGE	Mismatched volume or key range	The catalog was restored, or the volume contining the data set was restored. As a result the object's volume locations or keyranges in the CRA do not match those in the catalog.	Requires data set recovery.
EXTENTS	Mismatched extents	Data set was not properly closed, the catalog was restored, or the volume containing the data set was restored.	Requires data set recovery.

VSAM Object Records (Continued)

***MISCOMPARES**

Message	Type	Cause	Severity
HIGH USED RBA	Mismatched high used Relative Byte Address	Same as for mismatched extents.	Requires use of the VERIFY command to correct the high RBA.
STATISTICS	Mismatched statistics	Same as for mismatched extents.	No recovery action is required; mismatched statistics do not affect the accessibility of data.
OTHER	Mismatch of something other than the above fields, e.g. passwords.	Same as for mismatched extents.	Same as for mismatched statistics.

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

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

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

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

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

REPRO can be used to create a backup copy of the catalog. The catalog can be reloaded (using REPRO) from this backup copy to minimize the amount of work required to accomplish recovery. The usefulness of a

backup copy depends on what modifications have been made to the catalog since the backup was created. subsequent to taking a backup copy will affect the usefulness of the backup catalog:

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

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

Suballocating space to a VSAM data set. The volume space map is invalidated which, in itself, is not serious. However, the data set entry is also invalidated, a serious problem. (See "Ensuring Accessibility of Secondary Extents" for more information.)

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

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

CREATING AND CATALOGING OBJECTS

VSAM uses catalogs as a central information point for all VSAM data sets and the direct-access

volumes on which they are stored. You can define a VSAM object in a VSAM catalog only by using the Access Method Services DEFINE command. Additionally, you can use the DEFINE command to define nonVSAM objects in a VSAM catalog.

When you issue the DEFINE command to catalog an object, Access Method Services builds a catalog entry to describe the object. The VSAM objects you can define are:

- User catalog, which is a collection of information about nonVSAM objects, and about VSAM objects that reside on the volumes owned by the user catalog. You can create a user catalog at any time. A pointer to the user catalog is put in the system's master catalog. A master catalog is created during the SYSGEN process. The VSAM master catalog is the OS/VS2 system's primary catalog.
- Data space, which is direct-access device space used for a catalog, for VSAM clusters, for page spaces, and for alternate indexes. VSAM controls the allocation of space within each data space.
- Cluster, or VSAM data set, which is a collection of user-data records. There are three types of cluster data organization:
  - Entry-sequenced, or sequential, in which data records are read or written sequentially from one end to the other (first to last for writing, either direction for reading).
  - Key-sequenced, or indexed, in which a data record is read or written based on its key value. A key is a field, shorter than and within the record, that identifies the record.
  - Relative-record, or direct, in which a data record is read or written based on its relative record number—its displacement, in records, from the beginning of the cluster.
- Alternate index, which allows you to read and write data records in an entry-sequenced or key-sequenced cluster (called the *base cluster*) based on an alternate key.
- Path, which is a data set name for the combination of an alternate index and its base cluster, or an alias for a VSAM cluster. A path entry can be password protected.
- Page space, which is an amount of direct-access device space to be used exclusively by the OS/VS2 system.

The nonVSAM objects you can define are:

- NonVSAM data set, which is a collection of data records with sequential, partitioned, direct, or indexed-sequential data organization (that is, not VSAM data organization).
- Generation data group, which is a collection of nonVSAM data sets that are grouped together in a time-dependent manner.
- Alias, which is an alternate name for a user catalog or nonVSAM data set.

Before an object can be defined, you must have a VSAM catalog in which to define the object. Every VS2 system must have a VSAM master catalog before initial program load (IPL) is done for the system.

When you define an object, you specify attributes to be associated with it. The attributes include, for example, any passwords required to use data and how space is to be allocated. After the object is defined, it can be processed with other Access Method Services commands and with the user's VSAM program. After a cluster is defined, for example, you can load data records into it by using the REPRO command.

VSAM clusters, alternate indexes, page spaces, and catalogs are stored in data spaces. You can allow a data space to contain many VSAM objects by simply not specifying otherwise (the usual case and the default), or you can restrict a data space so that it contains only one VSAM object. In the usual case, you define a data space first, then define the data sets. To define a VSAM object as the only one in its data space, specify the parameter UNIQUE when you define the data set. When the object is defined, VSAM allocates a unique data space and assigns it to the object.

Refer to the chapter: "Command Format" for a description of the command format for DEFINE USERCATALOG. Examples of defining user catalogs are also included.

Preventing Duplicate Names in Catalogs

With multiple catalogs (the master catalog, user catalogs, and OS/VS CVOLs), you should take care so that a data set name in one catalog is not duplicated in another catalog. It is possible to have the same data set name in more than one catalog.

Access Method Services prevents you from cataloging two objects with the same name in the same catalog, and from altering the name of an object so that its new name duplicates the name of another object in the same catalog. A following section, "Order of Catalog Use: DEFINE," describes the order in which one of the catalogs available to the system is selected to contain the to-be-defined catalog entry. When you define an object, you should ensure that the catalog the system selects is the catalog you want the object's entry in.

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 object doesn't normally require that a volume be mounted, because the availability 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 a catalog is being defined. If the catalog in which the object is being defined is recoverable, the volume containing the catalog recovery area to be updated must be mounted. Before a system component, such as VSAM or Access Method Services, can process a data set or volume, the OS/VS system allocates the data set or volume to the system component. Usually, the data set or volume is allocated to the system component when the OS/VS Scheduler processes your job control language (JCL) DD statements.

However, when a data set name or volume serial number is specified with an Access Method Services command, but its data set or volume hasn't been described with a DD statement, the data set or volume is allocated dynamically. The information describing a data set that the DD statement would have provided is extracted from the data set's entry in the catalog. This type of allocation is called *dynamic allocation.* The data set's volume should be mounted as permanently resident or reserved to ensure successful dynamic allocation.

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=( serial [ , serial2 , ...] )
```

- DISP=OLD is required because VSAM allocates space to VSAM objects (that is, catalogs, clusters, alternate indexes, and page spaces). When you code DISP=NEW, the OS/VS system allocates space to the object. The space that the system allocates is not recorded in the object's catalog entry. Unless the object is a nonVSAM data set, the space allocated by the system is not available for the object's use.
- UNIT= can specify a device type and volume serial number for any direct-access storage device that is supported by your OS/VS system.
- VOLUME=SER= indicates the serial number(s) of the volume(s) on which data spaces are to be allocated. When more than one volume is indicated, all volumes must be of the same device type.

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 selected to contain the to-be-defined entry.
- 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) is selected to contain the to-be-defined entry. Otherwise,
- 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 a user catalog, or
  - the alias of a user catalog,

  the catalog so identified is selected to contain the to-be-defined entry.
- If no catalog has been identified, either explicitly or implicitly, VSAM defines an object in the master catalog.

You can specify a catalog that is to be available for use only with a job step, or that is to be available for the entire job. A STEPCAT DD statement identifies a catalog that is available for a single job step; a JOBCAT DD statement identifies a catalog that is available during the entire job. When both JOBCAT and STEPCAT catalogs are specified, the user catalog specified by STEPCAT is used for the job step. The JOBCAT and STEPCAT statements you code should be in the form:

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

and

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

Because the master catalog contains information about each user catalog, you don't need to specify the catalog's volume and device information.

You can specify more than one user catalog 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:

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

How to Use One Object as a Model for Another Object

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

You can override some of the model's attributes by explicitly specifying them with the DEFINE command. If you do not want to change or add any attributes, you need only supply the entryname of the object being defined and the MODEL parameter. (When you define a catalog, you must also specify the catalog's volume and space information, because the catalog, when defined, owns the volume.)

- If MODEL is specified as a parameter of USERCATALOG, PAGESPACE, or PATH:
  - the attributes of the model are defined for the object being defined, then
  - any attributes explicitly specified as parameters of the defined object are defined and override those of the model.
- If MODEL is specified as a parameter of CLUSTER or ALTERNATEINDEX (that is, at the cluster level) but is not specified as a subparameter of the DATA or INDEX parameter:
  - the attributes of the model are defined for the cluster-level object being defined, then
  - any attributes explicitly specified as parameters of the cluster-level object are defined and override those of the model.
  - the attributes of the model are defined for the object's data and index components, then
  - attributes explicitly specified at the cluster level are propagated to the data and index components, then

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

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

- If MODEL is specified as a subparameter of the DATA or INDEX parameter:
  - the attributes of the model are defined for the cluster-level object being defined if MODEL has been specified at the cluster level, then
  - any attributes explicitly specified as parameters of the cluster-level object are defined and override those of the model.
  - attributes explicitly specified at the cluster level are propagated to the object's data and index components, then
  - attributes of the model specified for the data or index component are defined (that is, the model specified with the MODEL subparameter of the DATA or INDEX parameter), then
  - attributes explicitly specified for the data and index components are defined (that is, the attributes specified with subparameters of the DATA or INDEX parameter).

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

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

- Specify a different type of device with the VOLUMES parameter.
- Change the length or position of the keys with the KEYS parameter.
- Change the size of records, bufferspace, or control intervals with the RECORDSIZE, BUFFERSPACE, or CONTROLINTERVALSIZE parameters.
- Change the type of cluster (that is, entry-sequenced, key-sequenced, or relative-record), the type of alternate index (that is, key-pointer or RBA-pointer), or the allocation-type of the object (that is, unique or nonunique).
- Change the *unit of allocation* with the TRACKS, CYLINDERS, or RECORDS parameters.

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

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. 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. Because the master catalog is created at system-generation time and because only one master catalog is allowed on a system, the user cannot issue DEFINE MASTERCATALOG to create a master catalog in a VS2 system. If you issue DEFINE MASTERCATALOG, VSAM creates a user catalog.

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. The master catalog contains a user-catalog entry that identifies the user catalog's volume. Each user catalog resides on a different volume. All VSAM objects 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.

When you define a user catalog, you can specify passwords for it even when the master catalog is not password-protected. To prevent an unauthorized user from deleting your user catalog, specify a password for it (the catalog's master password is required to delete it, but any password specified for the catalog results in a master password. See the MASTERPW, CONTROLPW, UPDATEPW, and READPW descriptions.)

How Space Is Assigned to a Catalog

When a catalog is defined, the catalog is the first VSAM object contained on the volume.

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.

When you want to define a catalog on a volume that is completely empty, and you want to assign specific tracks to the catalog, you cannot use an Access Method Services command or JCL parameters to do this. However, you can use the following procedure to ensure that your catalog occupies certain tracks on an otherwise unused volume:

1. Using JCL, allocate a dummy nonVSAM data set to occupy the space on the volume up to the point where you want the catalog to start.
2. Issue the DEFINE USERCATALOG command to allocate space for the catalog's data space and to build the user catalog.
3. Using JCL, delete the dummy data set and release its space.

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 both 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 space subparameters are coded but INDEX subparameters are not 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, the primary allocation for the catalog's data space is 100 cylinders) and for the catalog's data and index components 5 cylinders, (that is, the sum of the DATA and INDEX primary allocation amounts is 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 a 3330, 3330 model 11, 3350, or 2305 model 2 (2 tracks for the control area and 1 track for the sequence set record).
- 4 tracks for a 2305 model 1 (3 tracks for the control area and 1 track for the sequence set record).
- 5 tracks for 2314/2319 or 3340/3344 (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.

Identifying the Catalog's Volume

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). The DSCB is added to the VTOC and the volume-ownership information, also in the VTOC, is reset to indicate that the volume is owned by a VSAM catalog. VSAM determines whether the space is available on the volume and, if so, formats a DSCB for the data space.

NonVSAM Data Sets and the Catalog's Update Password

When you expect the user catalog to contain nonVSAM data sets, you should take care when you specify passwords for the catalog. When you specify an update password for the catalog, Access Method Services requires that you supply the password whenever you (or another user) want to delete one of the catalog's nonVSAM data sets. You can protect the catalog against unauthorized usage by specifying a master password. See the UPDATEPW description for more details about how a catalog's update password is used.

Estimating the Catalog's Space Requirements

Before you define a catalog, you should estimate the amount of space needed for the catalog's data component. To determine the approximate number of records required for the catalog, use the worksheet in Figure 3:

Variable Quantities	Formulas	Estimates
Basic requirement = 10 records		10
A = number of key-sequenced clusters	Ax3	
A^1 = number of key-sequenced clusters with alternate indexes to be upgraded	A^1x4	
B = number of entry-sequenced clusters	Bx2	
B^1 = number of entry-sequenced clusters with alternate indexes to be upgraded	B^1x3	
C = number of relative-record clusters	Cx2	
D = number of alternate indexes	Dx3	
E = number of path entries	E	
F = number of nonVSAM data set entries	F	
G = number of generation data group entries	G	
H = number of alias entries	H	
I = number of pagespaces	I*2	
J = number of volumes, depending on device type, owned by the catalog:		
J^1 = number of 2305 volumes	J^1x2	
J^2 = number of 2314/2319 volumes	J^2x3	
J^3 = number of 3330 and 3340/3344 volumes	J^3x4	
J^4 = number of 3330 Model 11 and 3350 volumes	J^4x6	
K = for each key-sequenced cluster and alternate index (KSIS) with space on more than two volumes, add "1" for each additional group of one to five volumes:		
K^1 = number of KSDSs with 3 to 7 volumes	K^1	
K^2 = number of KSDSs with 8 to 12 volumes	K^2x2	
K^3 = number of KSDSs with 13 to 17 volumes	K^2x3	
L = for each entry-sequenced cluster and relative-record cluster (ESDS) with space on more than five volumes, add "1" for each additional group of one to eight volumes:		
L^1 = number of ESDSs with 6 to 13 volumes	L^1	
L^2 = number of ESDSs with 14 to 21 volumes	L^2	
M = for each group of four data spaces on a volume, add "1"	M	
N = number of entry records required for the catalog's data component (total of A through M above)	N	

Figure 3. Worksheet for Estimating a Catalog's Space Requirements

When you define the catalog's space parameters, use the following format:

```
DEFINE USERCATALOG -
          (TRACKS(prim secn) ... ) -
          DATA -
          (RECORDS(prec srec) ... )
```

where:

- prec =number of entry records for the catalog's data component, or the value of N from the worksheet.
- srec = number of records for the data component's secondary extent:

  srec = .2N

- prim = the minimum amount of space for the primary extent (that is, allocation) of the catalog's data space. When you expect to have other VSAM objects in the catalog's data space, the value of prim should be larger than the calculated minimum. The calculated value of prim should be rounded upward to a whole number of tracks.

  When the catalog resides on a 3330, 3330 Model 11, 3350, or 2305-2 volume, the minimum value of prim is:

  prim = (.105N + 3.3) tracks

  When the catalog resides on a 2314/2319, 3340/3344, or 2305-1 volume, the minimum value of prim is:

  prim = (.16N + 5.4) tracks

- secn = the minimum amount of space for each secondary extent of the catalog's data space. The calculated value of secn should be rounded upward to a whole number of tracks:

  secn = .1prim tracks

The Catalog's Secondary Allocation Amount

You should specify a secondary allocation amount for both the catalog's data space and the catalog's data component. You cannot use the ALTER command to add or change a secondary allocation amount once the catalog is defined.

The secondary allocation quantity is allocated each time one of the catalog's keyranges needs more space—up to a total of thirteen times. The secondary allocation quantity you specify for the catalog's data component applies to the catalog's high-keyrange and low-keyrange (including their sequence set tracks). The catalog's index is not extended. When the high-keyrange or low-keyrange runs out of space, the specified secondary allocation quantity is allocated to the keyrange that requires the space (when both keyranges run out of space at the same time, two separate extents are allocated: one for each keyrange, and each equal to the secondary allocation quantity).

You use the secondary allocation to minimize the size of the low-keyrange and to provide for an extension of the high-keyrange: after the catalog's index set is allocated, ten percent of the remaining space (or one allocation unit, whichever is larger) is allocated to the high-keyrange; the remainder of the catalog's space is allocated to the low-keyrange. However, the high-keyrange might achieve only 25 percent utilization. Rather than specify a primary allocation quantity large enough to allow for low utilization of the high-keyrange and, therefore, create an unnecessarily large low-keyrange, you can specify a secondary allocation. The secondary allocation quantity is allocated to the high-keyrange when it requires space, without affecting the size of the low-keyrange. (The same applies to the low-keyrange, when it requires space.)

Defining a Data Space

The DEFINE command can be used to define VSAM data spaces or to reserve volumes for future VSAM use.

Refer to the chapter: "Command Format" for a description of the command format for DEFINE SPACE. An example of defining a VSAM data space is also included.

VSAM Data Spaces on a Volume

A VSAM data space is space on a direct access volume that is owned and managed by VSAM (as opposed to space managed by OS/VS using Direct-Access Device Space Management—DADSM.) All of the data spaces (and VSAM data sets) on a volume are 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).

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

When you define a data space, you must specify the volume that is to contain the data space. When you specify more than one volume, a data space of the size you specify is allocated on each volume; each volume comes under the control of the defining catalog. Access Method Services creates a volume entry in the catalog to describe each volume on which one or more data spaces have been defined.

Allocating the Volume

You can use a DD statement to allocate the volume on which the data space is to be defined. If you do not use a DD statement to allocate the volume, the volume identified with the VOLUMES parameter is dynamically allocated. The volume must be mounted as permanently resident or reserved.

If there is space available and there is no volume ownership conflict, VSAM ownership is indicated in the volume's VTOC and a VSAM data space is allocated on the volume. When the data space is the first data space on a volume owned by a recoverable catalog, Access Method Services also allocates a recovery space on the volume. The volume's VTOC is modified so that an ownership indicator is set in the format-4 DSCB. A format-1 (identifier) DSCB describing the data space is also written into the VTOC.

When you use a DD statement to allocate the volume on which the data space is to be defined, it should be in the form:

```
//dname DD DISP=OLD,
//    UNIT=( devtype [ , unitcount ][ ,DEFER ] ),
//    VOLUME=SER=( volser1 , volser2 , ... ] )
```

VSAM Objects in a Data Space

All VSAM objects (that is, catalogs, clusters, alternate indexes, and page spaces) 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. A *nonunique* object can share a data space with other VSAM objects. That is, a catalog, clusters, page spaces, and alternate indexes might also be in the data space. 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 many nonunique data sets, and a data set can be stored in more than one data space, on the same volume or on different volumes.

Space Assignment to VSAM Objects

When VSAM allocates space for a new data set or extends the space for an existing 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 existing 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 the data space can't be extended because a suballocated data set's secondary allocation amount is greater than the data space's secondary allocation amount, VSAM builds a new data space: its primary and secondary allocation amounts are the same as the data set's secondary allocation amount. Consequently, the total amount of space allocated to VSAM data spaces on a volume might exceed the predicted limit. 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.

Defining a Cluster

The DEFINE command can be used to define a catalog entry and allocate space for an indexed, nonindexed, or relative-record 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.*

Refer to the chapter: "Command Format" for a description of the command format for DEFINE CLUSTER. Examples of defining clusters are also included.

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:

- Type of data organization
- Whether the cluster is reusable (that is, for temporary storage of data)
- 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. All of the volumes specified for the cluster's data component or for the index component must be of the same device type. However, the data component's device type can be different than the index component's device type.

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 *OS/VS Virtual Storage Access Method*

(VSAM) Programmer's Guide. If you specify track allocation or specify record allocation for less than one cylinder, space is actually allocated by tracks.

Performance Options Information

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

- Whether records can span control intervals
- Whether to replicate index records
- Whether to place the sequence set of an index adjacent to data
- Whether to place the cluster's index on a separate volume from data
- The amount of free space to remain in the data component's control intervals and control areas when the data records are loaded (applies only to a key-sequenced cluster's data component)
- Control-interval size for VSAM to use instead of calculating the size itself

These options are discussed in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*

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
- Identity of an I/O error-handling routine (the exception exit routine) that is branched to before the program's SYNAD exit is taken
- Whether to preformat control areas during data record insertion
- 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 Protection" and in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*

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.

Defining an Alternate Index

An alternate index can be defined over a key-sequenced cluster or over an entry-sequenced cluster. An alternate index cannot be defined to support a reusable cluster, a relative-record cluster, a catalog, another alternate index, or a nonVSAM data set.

Before you can build an alternate index, you must define and load its base cluster (see the DEFINE CLUSTER command), then define the alternate index and relate it to the base cluster. The base cluster and alternate index are described by entries in the same catalog.

You can issue the DEFINE ALTERNATEINDEX command to define an alternate-index entry and allocate space for an alternate index. VSAM uses more than one catalog entry to describe the alternate index (an alternate index is conceptually a key-sequenced cluster):

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

Attributes of the alternate-index's components can be specified separately from the attributes of the alternate index. When attributes are specified for the alternate index as a whole and are not specified for the components, the attributes of the alternate index (except for its passwords and other protection attributes) are propagated to the alternate-index's components. The attribute specified for the component overrides that specified for the alternate index.

When you define the alternate index, you identify it with an entryname. The entryname identifies the alternate index when it is described with a JCL DD statement (that is, the entryname is the JCL DD statement's dsname) and when it is processed with another Access Method Services command. You can also name the alternate-index's components; a user's program can open and process the alternate-index's data or index component as a data set. For example, your program can open the index of an alternate index, then read and write its records. For more details on this kind of processing, see *OS/VS Virtual Storage Access Method (VSAM) Options for Advanced Applications.*

Refer to the chapter: "Command Format" for a description of the command format for DEFINE ALTERNATEINDEX. An example of defining an alternate index is also included.

Specifying Alternate-Index Information

When you define an alternate index, you specify descriptive information and performance, security, and data integrity options. Some of the information you specify can apply to either the alternate-index's data component, or to its index component, or to both—that is, to the alternate index as a whole. You specify information for the alternate index as a whole with the ALTERNATEINDEX parameter and its subparameters. Information for the data component or for the index component is specified with parameters of DATA or INDEX.

Descriptive Information

Information that you specify to describe the alternate index being defined includes:

- The base cluster related to the alternate index.
- Whether the alternate index is reusable (that is, temporary).
- Average and maximum lengths of alternate-index records.
- Length and position of the key field in data records of the alternate index's base cluster.
- Identity of the catalog that is to contain the alternate index's entries (the same catalog that contains the base cluster's entries).
- Identity of the volume(s) on which space is to be allocated for the alternate index.
- Amount of space to allocate for the alternate index and whether the alternate index is to reside in a separate data space.
- The least amount of I/O buffer space that a user's program is to provide when the program processes the alternate-index's data.

You can define an alternate index with its records assigned to various volumes according to alternate-key value. Each volume can contain the alternate-index records whose alternate key values are between a low-key value and a high-key value—that is, records whose alternate key values are within a key range. For example, your alternate index might reside on three volumes. The records with alternate keys A through E are on the first volume, F through M on the second volume, and N through Z on the third volume.

Performance Options Information

Information that you can specify to improve the performance of your alternate index include:

- Control-interval size, instead of allowing VSAM to calculate an optimum size.
- Whether to place the index's sequence set adjacent to the data control areas.
- Whether to place the alternate-index's index on a separate volume (or direct-access device) from the alternate-index records.
- The amount of free space to remain in the data component's control intervals and control areas when the alternate-index records are loaded (applies only to the alternate-index's data component).

These options are discussed in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*

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

Protection and Integrity Information

Information that you can specify to protect your alternate-index's records from other users and from possible system errors include:

- Passwords and related user-verification information.

- Identity of your authorization routine to verify that a user's program is authorized to gain access to the alternate-index's records.
- Identity of your I/O error-handling routine (the exception exit routine) that is branched to before the program's SYNAD exit is taken.
- Whether to preformat control areas before the alternate-index records are loaded.
- Whether, and to what extent, to share your alternate index between subtasks, jobs, and systems.
- Whether to erase the information that your alternate index contains when you delete it.

These options are discussed in "Data Security and Protection" and in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*

In a system with the Mass Storage System, you can also indicate whether an alternate index 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.

Defining a Path

The DEFINE PATH command is used to establish the relationship, the *path,* between an alternate index and its base cluster. The base cluster and its alternate index must already be defined when you define the path that relates them.

When your program opens a path for processing, both the alternate index and its base cluster are opened. When data in the base cluster is read or written using the path's alternate index, keyed processing is used; RBA processing is not allowed.

You can also use the DEFINE PATH command to establish a password-protected alias for a VSAM cluster. The cluster must already be defined when you define a path as its alias. The path entry, as an alias, allows the cluster another set of protection attributes. You can specify NOUPDATE access for the cluster, which allows you to open the cluster without also opening its upgrade set (that is, all of the cluster's alternate indexes that are to be updated whenever the cluster's data is changed.)

Refer to the chapter: "Command Format" for a description of the command format for the DEFINE PATH command. An example for defining a path is also included.

Defining a NonVSAM Data Set

The DEFINE command can be used to catalog a nonVSAM data set in a VSAM catalog. Any already existing nonVSAM data set can be introduced into a master or user catalog through the DEFINE command. The 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.

When you define or delete a nonVSAM data set in a password-protected catalog, the catalog's update (or higher level) password is required.

Refer to the chapter: "Command Format" for a description of the command format and an example of the DEFINE NONVSAM command.

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.

Refer to the chapter "Command Format" for a description of the command format for DEFINE ALIAS. An example is also included.

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).

[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."

Defining a Generation Data Group

The DEFINE command can be used to create a catalog entry for a generation data group. For further information on generation data groups, see *OS/VS Data Management Services Guide.*

Once a catalog entry for a genertion 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.

When you create a generation data group on an OS/VS2 Release 2 (or higher release level) system, it must be cataloged in a VSAM catalog. The model DSCB must exist on the generation data group's catalog volume. To create a generation data group in an OS CVOL, you must use another system (that is, OS/MVT, OS/VS2 Release 1, or OS/VS2 Release 1.6). All JCL specifications available to support generation data groups work on VS2 Release 2 in a way equivalent to OS/MVT, VS2 Release 1, or VS2 Release 1.6, whether the generation data group is cataloged in a VSAM catalog or in an OS CVOL.

Refer to the chapter: "Command Format" for a description of the command format for DEFINE GENERATIONDATAGROUP. An example is also included.

Defining a Page Space

The DEFINE command can be used to define an entry for a page space. A page space is an OS/VS2 system data set that contains pages of virtual storage; the pages are stored into and retrieved from the page space by the Auxiliary Storage Manager. A page space is a nonindexed data set (that is, an entry-sequenced cluster) that is entirely preformatted before it is used. In a system with the Mass Storage System, a pagespace cannot be defined on a mass storage volume. A page space must reside on a single volume. A page space cannot be opened as a user data set.

A page space has a maximum usable size equal to 65,535 paging slots (records). See the description for space size declarations (TRACKS, CYLINDERS, and RECORDS) in the Command Format chapter for further details.

You can define a pagespace in a user catalog, then move the catalog to a new system and establish it as the system's master catalog. When you define a pagespace in a user catalog, you must also code a STEPCAT or JOBCAT statement to identify and allocate the catalog.

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

Each pagespace is represented by two entries in the catalog: a cluster entry and a data entry (that is, a pagespace is conceptually an entry-sequenced cluster.) Both of these entries should be password protected if the pagespace is to be password protected. (Note: the passwords you specify with the DEFINE PAGESPACE command are put in both the pagespace's cluster entry and its data entry.) When you define pagespaces during system generation (SYSGEN), you should use the ALTER command to add passwords to each entry, since passwords cannot be specified during system generation.

In addition, you should ensure that the catalog containing the pagespace entry is password protected. Otherwise, a user can list the catalog's contents and find out each entry's passwords.

A page space is made known to the system as a system data set at system-generation time or through members of a partitioned 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 (that is, a page space cannot be used if its entry is in a user catalog).

A page space's passwords and protection attributes are cataloged in both the page space entry and its data component entry. You should specify master, control, update, and read passwords to prevent the page space from being accessed if its volume is moved to a VS1 system. The VS2 system automatically prevents a user's program from accessing a page space or its data component.

Refer to the chapter: "Command Format" for a description of the command format of DEFINE PAGESPACE. Examples of how to define page spaces are also included.

BUILDING AN ALTERNATE INDEX

The BLDINDEX command is used to build an alternate index. Before you can build an alternate index, you must define and load its base cluster (see the DEFINE CLUSTER command), then define the alternate index and relate it to the base cluster (see the DEFINE ALTERNATEINDEX command). The base cluster and alternate index are described by entries in the same catalog.

The alternate index's volume and the base cluster's volume must be mounted during the build process. Any volumes identified with the WORKFILES parameter must also be mounted.

An alternate index can be built for a key-sequenced cluster or for an entry-sequenced cluster. The base cluster must be nonempty (that is, its high-used RBA value cannot be zero.) An alternate index cannot be built to support an empty or reuseable cluster, a relative-record cluster, a catalog, another alternate index, or a nonVSAM data set.

Each of the base cluster's data records must be long enough to contain the alternate key. Each record's alternate-key value must be unique, unless the alternate index was defined with the NONUNIQUEKEY attribute.

Refer to the chapter: "Command Format" for a description of the command format for BLDINDEX. Examples of the BLDINDEX command are also included.

How an Alternate Index is Built

When an alternate index is built by BLDINDEX processing, Access Method Services opens the base cluster to sequentially read the data records, sorts the information obtained from the data records, and builds the alternate index records:

1. The base cluster is opened for read-only processing. Other users are prevented from accessing the base cluster's records by including the DISP=OLD parameter in the base cluster's DD statement. If INDATASET is specified, Access Method Services will dynamically allocate the base cluster with DISP=OLD.
2. The base cluster's data records are read and certain information is extracted to form the *key-pointer* pair:
   - When the base cluster is entry-sequenced, the alternate-key value and the data record's RBA form the key-pointer pair.
   - When the base cluster is key-sequenced, the alternate-key value and the data record's prime-key value form the key-pointer pair.

   If the base cluster's data records can span control intervals, the alternate key must be in the record's first segment (that is, the first control interval.)

3. The key-pointer pairs are sorted in ascending alternate-key order. If your program provides enough virtual storage, Access Method Services performs an internal sort (that is, the sorting of key-pointer pairs takes place entirely within main, or virtual, storage).

   You can determine the amount of virtual storage required to sort the records internally by using the following process:

   a. Sort record length = alternate key length + (prime key length (for a key-sequenced data set) or 4 (for an entry-sequenced data set)).

   b. Record sort area size = sort record length × number of records in base cluster, rounded up to the next integer multiple of 2048, or 32,768, whichever is greater.

   c. Sort table size = (record sort area size ÷ sort record length) × 4.

   The sum of 2 and 3 above is the required amount of virtual storage for an internal sort. This amount is in addition to the normal storage requirements for processing an Access Method Services command.

   If you do not provide enough virtual storage for an internal sort, or when you specify the EXTERNALSORT parameter, Access Method Services defines and uses two sort workfiles and sorts the key-pointer pairs externally. Access Method Services uses the sort workfiles to contain most of the key-pointer pairs while it sorts some of them in virtual storage. This process is called an external sort. An external-sort workfile is a VSAM entry-sequenced cluster, marked reuseable, with space suballocated from a VSAM data space. The minimum amount of virtual storage you'll need for an external sort is:

   32,768 + ((32,768 ÷ sort record size) × 4)

4. When the key-pointer pairs have been sorted into ascending alternate-key order, Access Method Services builds an alternate-index record for each key-pointer pair. If the alternate index is defined with the NONUNIQUEKEY attribute and more than one key-pointer pair have the same alternate-key values, the alternate-index record that is built contains the alternate-key value followed by the pointer values in ascending order. If the alternate index is defined with the UNIQUEKEY attribute, each alternate-key value is unique.

   When the record is built, it is written into the alternate index as though it is a data record being loaded into a key-sequenced cluster. Attributes and values that apply to the loading of data records can be specified when the alternate index is defined and are RECORDSIZE, CONTROLINTERVALSIZE, BUFFERSPACE, FREESPACE, WRITECHECK, SPEED, RECOVERY, REPLICATE, and IMBED.

5. When all alternate-index records have been built and loaded into the alternate index, both it and its base cluster are closed. Steps 1 through 4 are repeated for each alternate index that is specified with the OUTFILE or OUTDATASET parameter. When all alternate indexes have been built, the external-sort workfiles are deleted. Access Method Services finishes processing and issues messages that indicate the results of the processing.

DD Statements That Describe the SORT Workfiles

You can identify VSAM data space that the sort routine can use by specifying two dnames with the WORKFILES parameter and supplying two DD statements that describe the workfiles to be defined by Access Method Services. Each workfile DD statement should be coded:

//dname **DD DSNAME=** *dsname,* **VOL=SER=** *volser,*
// UNIT= *devtype,* **DISP=OLD,AMP='AMORG'**

dname
As specified in the WORKFILES parameter. If you do not specify the WORKFILES parameter and you intend to provide VSAM data space for external-sort workfiles, you identify the workfile DD statements with the names IDCUT1 and IDCUT2.

dsname
A data set name. The OS/VS Scheduler generates a data set name for the workfile if none is provided.

VOL=SER= *volser*
Required. Identifies the volume owned by the STEPCAT, JOBCAT, or master catalog in which the workfile is to be cataloged. The workfile's space is suballocated from one of the volume's VSAM data spaces. You can specify a maximum of five volumes for each workfile.

UNIT= *devtype*
Type of direct-access device on which the volume is mounted. You can specify a generic device type (that is, 3330, 2314, etc.) or a unit address (that is, 121, 248, etc.). You cannot specify SYSDA.

DISP=OLD
Required

AMP='AMORG'
Required.

If BLDINDEX is being used interactively in a TSO environment, these sort workfile DD statements must be contained in a logon procedure.

Order of Catalog Use: BLDINDEX

The base cluster and alternate-index entries are in the same catalog, because the catalog owns the volume or volumes that contains the cluster and its alternate index.

The order in which catalogs are selected to determine the catalog to contain the external-sort workfile entries (that is, entries that describe each workfile as an entry-sequenced cluster in a suballocated VSAM data space) is:

1. If a catalog is specified with the CATALOG parameter, that catalog is selected to contain workfile entries. Otherwise,
2. The user catalog specified in the current job step (STEPCAT) or, if none is specified for the job step, the user catalog specified for the current job (JOBCAT) is selected to contain the workfile entries. If more than one catalog is specified for the job step or job, the first job-step or job catalog is selected to contain the workfile entries. Otherwise,

3. If the entries (data set name on the DD statement) are identified with qualified entrynames and the first qualifier is the same as:
   - the name of a user catalog, or
   - the alias of a user catalog,

   the user catalog so identified is selected to contain the workfile entries. Otherwise,
4. The master catalog is selected to contain the workfile entries.

MODIFYING CATALOG INFORMATION

The ALTER command is used to alter attributes in catalog entries and to rename members of nonVSAM partitioned data sets. To alter an entry, you need to supply its name and the attributes 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. The entry's volume must also be mounted when the entry is cataloged in a recoverable catalog. A JCL DD statement can be used to cause the data set or volume to be allocated. In VS2, 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.

Key-sequenced VSAM data sets consist of three components: an index component, a data component, and the cluster entry itself. Most attributes of the VSAM data set are associated with its data or index components. Only retention period, owner ID and cluster protection attributes are associated with the cluster entry itself. If you use the cluster name in the ALTER command, only these attributes will be changed. If you use the cluster name in the ALTER command to alter any attribute not directly associated with the cluster entry, Access Method Services will terminate the ALTER request and issue an error message. The converse is also true; if you specify a data or index component name and attempt to alter an attribute directly associated with a cluster entry, Access Method Services will terminate your ALTER command and issue an error message. You must specify the explicit data or index component name in order to alter any of the remaining attributes such as shareoptions, bufferspace, writecheck, the data or index protection attributes, etc. It is recommended, therefore, that you specify data and index component names at DEFINE time in order to facilitate the use of the ALTER command. Otherwise, you will have to use the long system-generated names. Refer to Figure 8 in the chapter: "Command Format" to determine which values or attributes you may alter for a particular entry type.

You cannot ALTER any attributes of a user catalog except those directly associated with the cluster name. Data and index component names specified for a user catalog at DEFINE time are meaningless, because the system will always generate standard names. If you specify these system-generated names for the data or index components in the ALTER command, VSAM will return a catalog return code of "8", claiming the entry is not there. Additionally, if you alter the expiration date to an invalid date, such as "99999", Access Method Services will return a condition code of "0" even though the expiration date remains unchanged.

Refer to the chapter: "Command Format" for a description of the command format for ALTER. Examples of the ALTER command are also included.

Order of Catalog Use: ALTER

The order in which catalogs are searched when an entry is to be located to be altered is:

1. If a catalog is specified in the CATALOG parameter, only that catalog is searched. If the entry is not found, a "no entry found" error is returned to the user.
2. Any user catalog(s) specified in the current job step (with a STEPCAT DD statement) is searched. If more than one catalog is specified for the job step, the catalogs are searched in order of concatenation. If the entry is found, no other catalog is searched.

   If a STEPCAT catalog is specified and the entry is not found, the JOBCAT catalog is not searched. The catalog search continues with step 3 below.

   If no STEPCAT catalog is specified for the job step, and a user catalog is specified for the current job (with a JOBCAT DD statement), the JOBCAT catalog(s) is searched. If more than one catalog is specified for the job, the catalogs are searched in order of concatenation. If the entry is found, no other catalog is searched. Otherwise,
3. If the entry is identified with a qualified entryname and its first qualifier is the same as:
   - the name of a user catalog, or
   - the alias of a user catalog, or
   - the alias of a control volume,

   the user catalog or control volume so identified is searched. If the entry is found, no other catalog is searched. Otherwise,
4. The master catalog is searched. If the entry is not found, a "no entry found" error is returned to the user.

Generic Names and ALTER

To alter entries with qualified names—for example, PAYROLL.74.MAY—you can identify the entry with its full name (that is, all qualifiers) or with all qualifiers but one. The unspecified qualifier is indicated with an asterisk (for example, PAYROLL.*.MAY). This kind of shortened name, PAYROLL.*.MAY, is called a *generic name.* When you specify the full entryname, only the entry so identified is modified. When you specify all qualifiers of the entryname but one, the entries whose entrynames match the supplied qualifiers are altered, if they contain the type of information specified with the ALTER command's parameters.

For example, when you specify PAYROLL.*, the entries that might be altered are those whose entrynames contain two qualifiers, the first qualifier being PAYROLL. When you specify PAYROLL.*.MAY, the entries that might be altered are those whose entrynames contain three qualifiers, the first being PAYROLL and the third and last being MAY.

When you identify a catalog (that is, when you specify the ALTER command's CATALOG parameter), Access Method Services searches only the specified catalog for generically named entries. If no catalog is specified, the catalog is selected as indicated in "Order of Catalog Use: ALTER" and is searched for generic-named entries.

You must always specify the first qualifier of a qualified entryname. You can substitute an asterisk for no more than one of the other qualifiers.

You must identify all qualifiers in the qualified name. For example, PAYROLL.* cannot be used to identify a qualified name that contains three or more qualifiers, even though its first qualifier is PAYROLL. PAYROLL.74.MAY.* cannot be used to identify a qualified name that has more or fewer than four qualifiers.

Renaming Generically-Named Entries

You can use generic names to rename a group of cataloged objects. To do this, you specify both the entryname and the newname as generic names. For example, if each entryname identified with the generic name A.*.B is to be renamed with the generic name A.*.C, all entrynames that have A as the first qualifier and B as the third and last qualifier are renamed. The new name has A as the first qualifier and C as the third and last qualifier:

Old Name	New Name
A.1.B	A.1.C
A.2.B	A.2.C
A.3.B	A.3.C

If each entryname identified with the generic name A.B.*.D is to be renamed, all entrynames that have A and B as the first and second qualifiers, and D as the fourth and last qualifier, are renamed. If the new generic name is C.*.DATA, the entrynames are renamed as follows:

Old Name	New Name
A.B.1.D	C.1.DATA
A.B.2.D	C.2.DATA
A.B.3.D	C.3.DATA

VSAM Volume Recovery Function

The VSAM Volume Recovery function removes all VSAM data spaces and resets volume ownership for a volume that can't be located with its catalog entry (that is, a system failure or I/O error might have damaged the volume entry). The volume's format-4 DSCB (in the VTOC) is reset to remove its ownership from the VSAM catalog. NonVSAM data sets on the volume aren't affected. The VSAM catalog that owns the volume isn't accessed or modified—the (damaged) volume entry is unchanged.

The VSAM Volume Recovery function can be used to remove a user catalog from a volume without first deleting each of the catalog's objects (that is, VSAM objects and nonVSAM data sets described with the catalog's entries). The VSAM Volume Recovery function allows you to remove a VSAM user catalog that has been damaged by a head crash, a standalone DASDI, or a similar accident.

VSAM erases all VSAM data spaces (even when they contain data or a catalog) and rewrites the volume's VTOC to indicate additional free space on the volume. You should use this function only when you can't access the catalog that owns the volume. You should use the function carefully, in order to prevent unwanted loss of data. When improperly used, this function can contriubte to system integrity exposures.

When you use the ALTER command to recover from a damaged volume, you code the command in this format:

ALTER *entryname* [/ *password*]
FILE(*dname* **)**
REMOVEVOLUMES(*volser* [ъ *volser* ...] **)**

entryname [/ *password*]
names the master catalog. If the master catalog is password protected, then its master password must also be supplied.

FILE(*dname* **)**
specifies the name of a DD statement that describes a volume to be reset. If more than one volume is to be reset, they must be of the same device type. Concatenated DD statements are not allowed. This parameter is required.

REMOVEVOLUMES(*volser* [ъ *volser* ...] **)**
identifies the volume(s) on which all VSAM data spaces are to be removed. VSAM ownership of the volume is also relinquished. Volumes owned by the master catalog cannot be specified.

DISPLAYING CATALOG INFORMATION

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. See "Appendix B: Interpreting LISTCAT Output" for an explanation of the output produced as a result of the LISTCAT command.

If entries to be listed are selected by name, the name(s) can be specified in its entirety, can be specified as a generic name, or can be specified with the LEVEL parameter.

Entries are specified by generic name by supplying all but one qualifier of the name. The qualifier omitted is indicated by an asterisk (*). The first qualifier cannot be omitted.

When you specify ENTRIES (A.*), all two-qualifier entrynames that have A for the first qualifier are selected to be listed. When you specify ENTRIES(A.*.B), all three-qualifier entrynames that have A as the first qualifier and B as the last qualifier are selected to be listed. You can further limit the printing by identifying certain entrytypes to be printed (that is, by specifying NONVSAM, CLUSTER, etc.).

When you specify LEVEL(A.*.B), all entrynames that have A as the first qualifier and B as the third qualifier are selected to be listed—and some of the selected entrynames might have four or more qualifiers (each must have at least three qualifiers and satisfy the A.*.B selection criteria).

When you specify LEVEL(A), all entrynames that have A as the first qualifier (regardless of the number of qualifiers) are selected to be listed.

You cannot specify an "*" as the last qualifier when you use the LEVEL parameter (note that when you specify LEVEL(A), more entries might be listed than when you specify ENTRIES(A.*), even though both ways appear, at first glance, to be identical). If you specify LEVEL(A.*), the LISTCAT operation terminates with an error message.

Refer to the chapter: "Command Format" for a description of the command format for LISTCAT. Examples are also included.

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:

1. If a catalog is specified in the CATALOG parameter, only that catalog is listed. Otherwise,
2. 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. Otherwise,
3. 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:

1. If a catalog is specified in the CATALOG parameter, only that catalog is listed. Otherwise,
2. 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. Otherwise,
3. 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. Otherwise,
4. The master catalog is listed.

DELETING CATALOG ENTRIES

The DELETE command is used to delete entries from a catalog or to delete members of a nonVSAM partitioned data set. When the entry represents an object that contains space in a VSAM data space, the object's space is made available for other VSAM objects.

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 or alternate index is to be deleted and erased.
- A nonVSAM data set is to be deleted and scratched.

If the entry is cataloged in a recoverable catalog, its prime catalog recovery area volume must be mounted so that the CRA can be updated to reflect the entry's deletion.

JCL can be used to cause a data set or volume to be allocated. 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. Any volume to be dynamically allocated must be mounted as permanently resident or reserved.

Entries can be deleted from more than one catalog with a single DELETE command. You do this by specifying many entrynames and not coding the CATALOG parameter. Use concatenated JOBCAT or STEPCAT DD statements to identify the catalogs that contain the entries.

Note: When two catalogs in your system might contain the same entryname, you should issue separate DELETE commands that reference each catalog.

The kinds of catalog entries you can delete are:

- ALIAS—The alias entry is deleted.

  Password required: the catalog's update (or higher level) password.

- ALTERNATEINDEX—The alternate index entry, its data component's entry, and its index component's entry are deleted. Space allocated to the alternate index's components is made available for other VSAM objects. Each path entry that relates the alternate index to its base cluster is also deleted.

  When ERASE is coded, or when the alternate index has the ERASE attribute, the volume containing the alternate-index's data component must be mounted.

  The alternate-index's components cannot be deleted separately. You must delete the alternate index as a whole. When the alternate index's components reside in unique data spaces, the volumes containing these components must be mounted.

  If the alternate index is cataloged in a recoverable catalog, the prime catalog recovery area volume for the alternate index must be mounted. This volume is the first index volume of the base cluster over which the alternate index is defined if the cluster is a KSDS. Otherwise, it is the data volume of the cluster. Its volume serial number is printed by Access Method Services when the alternate index is defined.

Password required: the alternate-index's master password or its catalog's master password.

- CLUSTER—The cluster entry, its data component's entry, and, if the cluster is key-sequenced, its index component's entry are deleted. Space allocated to the cluster's components is made available to other VSAM components.

  Alternate indexes that are associated with the cluster are deleted. The cluster's upgrade-set entry, if it exists, is also deleted. Path entries that relate the cluster to its alternate indexes are also deleted.

  Space allocated to the alternate index's components is made available to other VSAM objects. When the alternate index's components reside in unique data spaces, the volumes containing these components must be mounted.

  When ERASE is coded, or when the cluster has the ERASE attribute, the volume containing the cluster's data component must be mounted.

  The cluster's components cannot be deleted separately. You must delete the cluster as a whole. When the cluster's components reside in unique data spaces, the volumes containing these components must be mounted.

  When the cluster is cataloged in a recoverable catalog, the prime catalog recovery area of the volume of the cluster must be mounted. This volume is the first volume containing the index component of the cluster if the cluster is a KSDS. Otherwise, it is the first data volume of the cluster. Its volume serial number is printed by Access Method Services when the cluster is defined.

  Password required: the cluster's master password or its catalog's master password.

- GENERATIONDATAGROUP—The generation data group entry is deleted. The generation data group must be empty (that is, it cannot contain generation data sets) unless you specify FORCE.

  Password required: the catalog's master password.

- MASTERCATALOG—In a VS2 system, the VSAM master catalog is the system's primary catalog. You are not allowed to delete your system's master catalog.

- NONVSAM—The nonVSAM entry is deleted. When SCRATCH is coded, the data set's DSCB is removed from its volume's VTOC—the volume containing the nonVSAM data set must be mounted.

  All alias entries that point to the nonVSAM entry are also deleted.

  When the nonVSAM data set is partitioned, you can delete one of its members by specifying the entryname as pdsname(membername)

  Password required: the catalog's update (or higher-level) password.

- PAGESPACE—The pagespace entry and its data component entry are deleted. Space allocated to the pagespace is made available to other VSAM objects.

  The page space being deleted must be inactive when it is deleted, and all other page spaces on the same volume must also be inactive.

  Password required: The pagespace's master password or the catalog's master password.

  Password required: the path's master password or its catalog's master password.

- PATH—The path entry is deleted.
  Entries for the path's alternate index and base cluster are undisturbed.

- SPACE—All empty data spaces on the volume are deleted. The DSCB entry that describes each VSAM data space in the volume's VTOC is removed. The space is made available to other OS/VS2 system objects.

  When all VSAM data spaces on the volume have been deleted, or when you specify FORCE, its volume entry in the catalog is deleted. The volume ownership indicator in the volume's VTOC is reset so that a VSAM catalog no longer owns the volume.

  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.

  The volume must be mounted.

  You cannot specify the PURGE parameter when you delete VSAM data spaces.

  When you identify data spaces to be deleted, you cannot identify any other type of cataloged object to be deleted with the same command. You can, however, follow the DELETE SPACE command with other Access Method Services commands.

  Password required: the catalog's update (or higher-level) password. When you also specify FORCE, you must supply the catalog's master password.

- USERCATALOG—The catalog's self-describing entries and its user-catalog entry in the master catalog are deleted. The DSCB entry that describes the user catalog's data space is removed from the volume's VTOC. The volume ownership indicator in the volume's VTOC is reset so that the volume is not owned by a VSAM catalog.

  The catalog must be empty—it cannot contain any catalog entries, unless you specify FORCE.

  All alias entries in the master catalog that point to the user-catalog connector entry are also deleted.

  You cannot code the ERASE or CATALOG parameters when you delete a user catalog.

  The catalog's volume must be mounted.

  Password required: the user-catalog's master password is specified with its entryname.

You cannot delete a data or index entry separately from its associated cluster, alternate-index, or user-catalog entry.

Refer to the chapter: "Command Format" for a description of the command format for DELETE. Examples of deleting VSAM and nonVSAM data sets are also included.

Order of Catalog Use: DELETE

The order in which catalogs are searched when an entry is to be located to be deleted is:

1. If a catalog is specified in the CATALOG parameter, only that catalog is searched. If the entry is not found, a "no entry found" error is returned to the user.
2. Any user catalog(s) specified in the current job step (with a STEPCAT DD statement) is searched. If more than one catalog is specified for the job step, the catalogs are searched in order of concatenation. If the entry is found, no other catalog is searched.

   If a STEPCAT catalog is specified and the entry is not found, the JOBCAT catalog is not searched. The catalog search continues with step 3 below.

   If no STEPCAT catalog is specified for the job step, and a user catalog is specified for the current job (with a JOBCAT DD statement), the JOBCAT catalog(s) is searched. If more than one catalog is specified for the job, the catalogs are searched in order of concatenation. If the entry is found, no other catalog is searched. Otherwise,
3. If the entry is identified with a qualified entryname and its first qualifier is the same as:
   - the name of a user catalog, or
   - the alias of a user catalog, or
   - the alias of a control volume,

   the user catalog or control volume so identified is searched. If the entry is found, no other catalog is searched. Otherwise,
4. If the entry is not found, the master catalog is searched. If the entry is not found in the master catalog, a "no entry found" error is returned to the user.

Generic Names and DELETE

To delete entries with qualified names—for example, PAYROLL.74.MAY—you can identify the entry with its full name (that is, all qualifiers) or with all qualifiers except one. The unspecified qualifier is indicated with an asterisk (for example, PAYROLL.*.MAY). This kind of shortened name, PAYROLL.*.MAY, is called a *generic name.* When you specify the full entryname, only the entry so identified is modified. When you specify all qualifiers of the entryname except one, the entries whose entrynames match the supplied qualifiers are deleted, if they satisfy other parameters specified with the DELETE command's parameters.

For example, when you specify PAYROLL.*, the entries that might be deleted are those whose entrynames contain two qualifiers, the first qualifier being PAYROLL. When you specify PAYROLL.*.MAY, the entries that

might be deleted are those whose entrynames contain three qualifiers, the first being PAYROLL and the third and last being MAY.

When you identify a catalog (that is, when you specify the DELETE command's CATALOG parameter), Access Method Services searches only the specified catalog for generically named entries. If no catalog is specified, the catalog is selected as indicated in "Order of Catalog Use: DELETE" and is searched for generic-named entries.

You must always specify the first qualifier of a qualified entryname. You can substitute an asterisk for no more than one of the other qualifiers.

You must identify all qualifiers in the qualified name. For example, PAYROLL.* cannot identify a qualified name that contains three or more qualifiers, even though its first qualifier is PAYROLL. PAYROLL.74.MAY.* cannot identify a qualified name that has more or fewer than four qualifiers.

MOVING ENTRIES

The EXPORT and IMPORT commands allow you to create backup copies of alternate indexes and clusters, to transport user catalogs, alternate indexes and clusters from one system to another, and to prevent (and subsequently, with the IMPORT command, to allow) 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, a catalog connector entry is created for it in the new system's master catalog and the user catalog is physically transported to the new system. When a user catalog is exported, it is not copied; the user catalog remains on its volume in its original form. 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 demountable volume, you can physically move the volume to another system. 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.

When a cluster or alternate index is exported, its catalog entry is copied to a movable volume along with the object's contents. The entries and components are subsequently copied into a new system.

Exportation of a cluster or alternate index 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 object, but the copy in the original system is marked to indicate that there is a copy elsewhere. If you want to export permanently a base cluster and its associated alternate indexes, the alternate indexes must be exported first since deletion of a base cluster causes deletion of any alternate indexes defined over it. A base cluster must always be imported prior to importing any of its alternate indexes.

When a cluster or alternate index is exported, its catalog information (that is, the attributes specified with the DEFINE command—except the catalog's name—and modified with subsequent ALTER commands) is moved with the cluster's data records. You cannot export an empty cluster or alternate index.

The permanent exportation of a RACF-protected cluster or alternate index causes the deletion of the data set profiles. A temporary exportation does not. Also, when a RACF-protected data set is exported, the RACF indicators are moved to the portable data set, but the profile is not.

When a previously exported RACF-protected object is imported, you have an option to reuse the old profiles or to establish new ones. The IMPORT command in the chapter "Command Format" will give you more information on RACF options when importing a VSAM object.

Exporting or importing a RACF-protected catalog will cause no change in the status of the RACF indicators in the catalog or the profiles associated with the catalog.

The portable copy of a cluster or alternate index is a variable-blocked, spanned, sequential data set. The user may specify, with the DCB parameter

of the OUTFILE DD statement, a blocksize for the portable data set other than the default of 2048 bytes.

Refer to the chapter: "Command Format" for a description of the command formats for the EXPORT and IMPORT commands. Examples showing how data sets are exported and imported are also included.

Exporting an Entry

The EXPORT command is used in conjunction with the IMPORT command to move alternate indexes, clusters, and user catalogs from one system to another, to provide backup copies of clusters and alternate indexes, 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, page space, or path. Access Method Services automatically exports along with a cluster or alternate index any paths defined over it. The paths are redefined when the object is subsequently imported.

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 all users and its aliases are deleted. The person who is responsible for the catalog can issue IMPORT CONNECT to make the catalog available again.

You should take care to list the user catalog's aliases before you issue the EXPORT command. You can use the LISTCAT command to list the aliases:

```
LISTCAT -
   CATALOG(mastercatname /password )-
   ENTRIES(usercatname )-
   ALL
```

The user catalog's aliases are listed under the heading "ASSOCIATIONS."

When you use the IMPORT command to make the catalog available to another system, you can issue the DEFINE ALIAS command to reestablish each of the catalog's aliases.

When a VSAM cluster or alternate index that has paths defined over it is exported, the catalog information for those paths is also exported. IMPORT uses this information to reestablish the paths in the target catalog. Therefore, EXPORT can be used to export path entries, but only as parts of a larger structure being exported (that is, a cluster or alternate index).

Importing an Entry

The IMPORT command is used in conjunction with the EXPORT command to move a cluster, alternate index, or user catalog from one system to another. When a cluster or alternate index is imported, it is automatically reorganized. When you move an object into the new system, you can change some of its attributes (as described with the OBJECTS subparameters).

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, alternate index, 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. There are two exceptions to this:

- When IMPORT is used to replace a cluster or alternate index with its backup copy. When the object's backup copy was made, its catalog entry was marked temporary (that is, TEMPORARY was coded when the object was exported). The object's temporary entry is deleted when it is imported; a new entry is built and imported in place of the temporary entry.
- When the cluster or alternate index has been predefined in the receiving catalog. The predefined entry must be for an empty data set. If a duplicate entryname exists for an empty data set, IMPORT assumes that this is a predefined entry whose attributes and space allocation are to be used in place of those in effect when the data set was exported.

If duplicate entrynames are found and the entry in the receiving catalog is not marked as temporary and is not empty, the IMPORT 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.

A check is also made for an empty target data set. If such a data set is present, IMPORT makes no attempt to define a replacement data set. Instead, IMPORT copies the data records from the portable data set into the empty target data set. If you wish to import into an empty target data set, you must specify the INTOEMPTY parameter.

Note: If you do not want the cluster or alternate index reorganized, use OS/VS IEHDASDR to copy the object. See *OS/VS Utilities* for a description of the IEHDASDR program.

CONVERTING AN OS CATALOG'S ENTRIES TO VSAM CATALOG ENTRIES

When you want to use the entries in an OS catalog with your VS2 system, you can either establish a connector in the master catalog to the OS catalog or you can convert the OS catalog entries to VSAM catalog entries. (**Note:** The OS catalog is called a *control volume* or CVOL.) You can convert the OS CVOL entries to VSAM catalog entries using one of two methods:

- Use the DEFINE command to convert some of the entries in an OS CVOL to VSAM catalog entries.
- Use the CNVTCAT command to convert all of the entries in an OS CVOL to VSAM catalog entries in a user or master catalog.

Refer to chapter "Command Format" for a description of the command format for CNVTCAT. Examples are also included.

Connecting an OS CVOL to the Master Catalog

When you want to use an OS CVOL in an OS/VS2 system, you must define the CVOL in the system's master catalog. For a description of how this is done, and how OS CVOLs are used in an OS/VS2 system, see *OS/VS2 Using OS Catalog Management with the Master Catalog: CVOL Processor.*

Using the DEFINE Command to Convert Some of the CVOL's Entries

When you issue the DEFINE command to redefine, in a VSAM catalog, objects cataloged in an OS catalog (that is, OS CVOL), you can:

- Selectively catalog entries from the OS catalog
- Convert a large OS catalog to two or more VSAM catalogs, without first splitting the OS catalog

However, this method requires that you issue (that is, keypunch or type in) many DEFINE commands. To lessen the amount of keypunching, you might use the following procedure:

1. Execute the IEHLIST utility program to list the OS catalog. The output of the IEHLIST program is the input to step 2. The IEHLIST program is described in detail in *OS/VS Utilities.*
2. Write and execute:
   - A program to generate DEFINE ALIAS statements for each CVOL pointer entry and alias entry in the OS CVOL listing.
   - A program to generate DEFINE ALIAS statements for each CVOL pointer entry that relates the CVOL pointer names to the appropriate catalog name.
   - A program to generate DEFINE NONVSAM statements for each nonVSAM data set in the OS CVOL listing.
   - A program to generate DEFINE GENERATIONDATAGROUP statements for each generation index pointer entry in the OS CVOL listing.

   The output of step 1 (the IEHLIST listing) is the input to your program. The output of your program is a series of DEFINE ALIAS, DEFINE NONVSAM, and DEFINE GENERATIONDATAGROUP commands.
3. Select the DEFINE statements that represent the entries you want cataloged in one of the VSAM catalogs.
4. Code and execute the appropriate JCL statements to execute the Access Method Services program, IDCAMS, using the selected DEFINE statements from step 3 as SYSIN input statements.
5. Repeat steps 3 and 4 until you have selected and cataloged as many entries as you want in the appropriate VSAM catalogs. CVOL aliases must be put in the VSAM master catalog.

Using the CNVTCAT Command to Convert All of the OS CVOL's Entries to VSAM Catalog Entries

The CNVTCAT command is used to convert entries in an OS catalog to entries in an existing master or user catalog. CNVTCAT converts data set entries, generation data group entries, alias entries, and CVOL-pointer entries to appropriate VSAM catalog entries.

CNVTCAT was designed to be executed only once for an OS catalog. The execution of CNVTCAT is time-consuming for any catalog greater than one hundred entries. Therefore, you should not plan to repeatedly convert the entries of a large OS catalog.

When an OS catalog structure of a system catalog and many OS CVOLs are converted to a VSAM master catalog and many user catalogs, you can use Access Method Services commands to perform the entire operation. You can also merge an OS CVOL's entries into a VSAM catalog, using the CNVTCAT command. However, you cannot easily divide the entries of a large OS catalog among many VSAM user catalogs. When the OS catalog is organized on the basis of qualified data set names, you can split the OS catalog into two or more OS catalogs using the IEHMOVE function of a selective COPY CATALOG command. You can then convert the entries of the smaller OS catalogs into selected VSAM catalogs. When the OS catalog is not organized by qualified data set name, you can catalog each individual entry using the DEFINE command as illustrated in the section "Using the DEFINE Command to Convert Some of the CVOL's Entries."

OS control volumes (CVOLs) can be interchanged within different operating systems: OS, VS2-1,VS2-2, and VS2-3. When you intend to continue production on another operating system while converting to OS/VS2, you might want to use OS CVOLs because they can be transferred between operating systems.

When you catalog an OS CVOL in the system's master catalog, you can use the CVOL to point to its cataloged entries. You cannot use a STEPCAT or JOBCAT DD statement to identify an OS CVOL, however. An OS CVOL and a VSAM user catalog might reside on the same volume, since the CVOL is considered a nonVSAM data set.

You can issue the DELETE ALIAS command to delete a CVOL's alias in the master catalog. When you issue the DELETE NONVSAM command to uncatalog the OS CVOL, all of its aliases are also deleted.

Figure 4 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 4. OS Catalog Entry Types and VSAM Equivalents

When you use the CNVTCAT command, you can cause all the entry types shown in Figure 0 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 1 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 5 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 catalogs. The bottom half of Figure 5 shows the master catalog and two user catalogs after SYSCTLG and SYSA and SYSB have been converted.

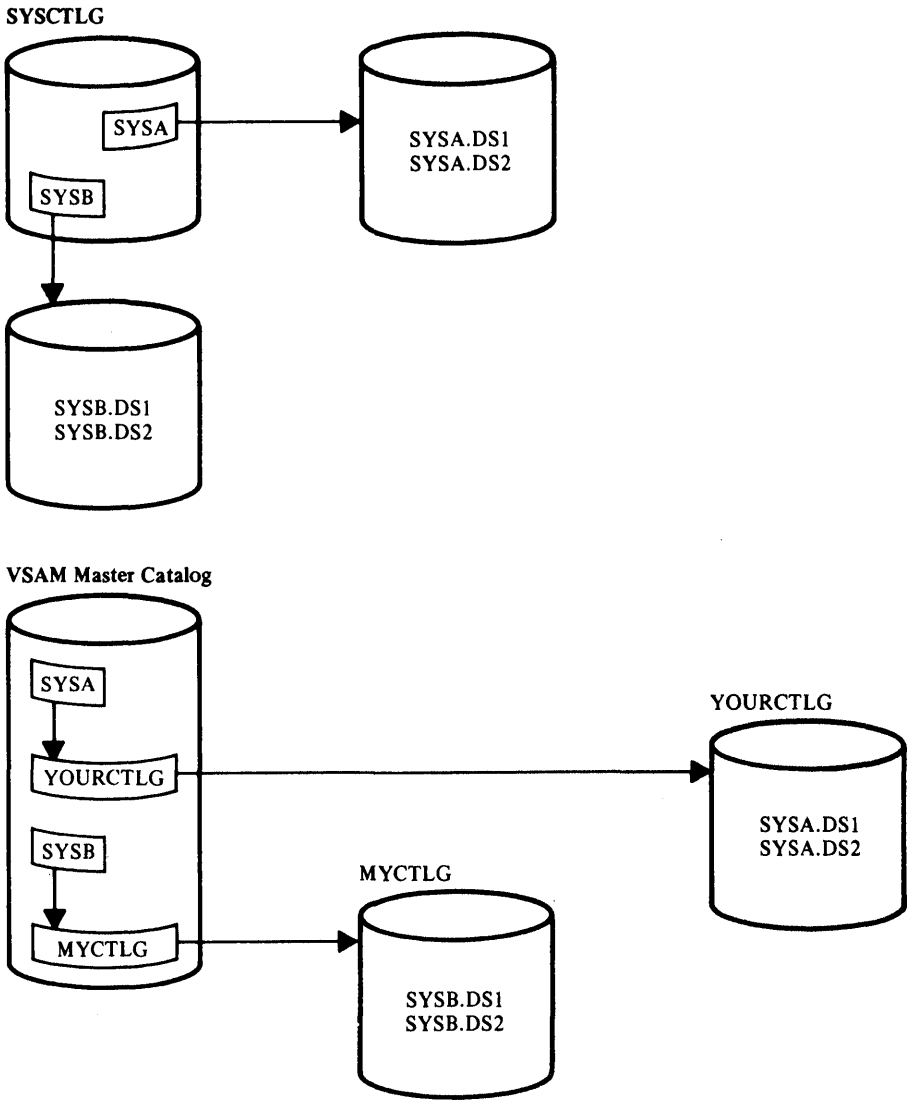


Figure 5. Converting Control Volume Entries

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 "Estimating the Catalog's Space Requirements" to help you estimate the VSAM catalog's size.

Note: Control volumes can be used in a VS2 system without converting entries. See *OS/VS2 Using OS Catalog Management with the Master Catalog: CVOL Processor* for information about the use of control volumes in a VS2 system.

Order of Catalog Use: CNVTCAT

The order in which VSAM catalogs are evaluated to select the catalog that receives the converted entries is:

- If a catalog is specified in the CATALOG parameter, that catalog receives the converted entries. Otherwise,
- The user catalog specified for the current job step (STEPCAT) or, if none is specified for the job step, the user catalog specified for the current job (JOBCAT) receives the converted entries. (If more than one catalog is specified for the job step or job, the first catalog in order of concatenation is selected.) Otherwise,
- The master catalog receives the converted entries.

The master catalog always receives alias entries that point to user catalogs.

RESTORING A CLUSTER'S END-OF-FILE VALUES

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.

Refer to the chapter: "Command Format" for a description of the command format for the VERIFY command.

RESTORING CATALOG ENTRIES AFTER SYSTEM FAILURE

When you create a user catalog or a master catalog, you can define it as recoverable. A recoverable catalog has, on each volume it owns, an area that contains a copy of some of the catalog's entries. Each volume's area is called the *catalog recovery area* or CRA. When a system or device failure occurs that damages the catalog or some of its entries, each volume's catalog recovery area contains the information needed to restore the damaged entries. The contents of a volume's catalog recovery area depend on the types of objects the volume contains. Figure 6 identifies the volume whose catalog recovery area contains a copy of the cataloged object:

Type of entry	Volume whose catalog recovery area contains a copy of the catalog entry
Volume entry	Its own volume
Key-sequenced cluster entry and its data and index entries	The volume that contains the (first part of the) cluster's index component
Alternate index entry and its data and index entries, when the alternate index is for a key-sequenced cluster	The volume that contains the (first part of the) alternate index's base cluster's index component
Path entry, when the path is related to a key-sequenced cluster	The volume that contains the (first part of the) path's base cluster's index component
Entry-sequenced cluster's entry and its data entry	The volume that contains the (first part of the) cluster's data component
Alternate index entry and its data and index entries, when the alternate index is for an entry-sequenced cluster	The volume that contains the (first part of the) alternate index's base cluster's data component
Path entry, when the path is related to an entry-sequenced cluster	The volume that contains the (first part of the) path's base cluster's data component
Relative-record cluster entry and its data entry	The volume that contains the (first part of the) cluster's data component
NonVSAM data set	The volume that contains the nonVSAM entry's catalog
Generation Data Group	The volume that contains the entry's catalog
Alias Entry	The volume that contains the catalog in which the nonVSAM association for the alias is defined
User-catalog connector entry in the master catalog	The volume that contains the master catalog
Catalog's self-describing entries	These entries are not duplicated in any catalog recovery area

Figure 6. Catalog Recovery Area Contents

If you should discover that your recoverable catalog is damaged such that its entries are inaccessible, are downlevel, or contain erroneous information, you have the option of choosing one of two different methods to restore your catalog to a usable condition.

- The EXPORTRA/IMPORTRA Method: You would most likely use this method to selectively repair specific catalog entries. Also, remember that reorganization of your catalog and your data is a by-product of this approach since it involves the movement of data. You may use the following procedure to restore the usability of your catalog.

  1. Issue the LISTCRA command to list and, optionally, compare the catalog recovery area entries (that is, the catalog recovery area that contains copies of the damaged catalog entries).
  2. Issue the EXPORTRA command to obtain a copy of the damaged entries from the catalog recovery area and, when a VSAM data set entry is moved, obtain a copy of the data set's contents.
  3. Clear the damaged volume or reset it so that it is usable by doing one or more of the following:
     - Issue the DELETE SPACE command with the FORCE parameter to remove VSAM data spaces from the volume.
     - Execute the IEHDASDR program with the DUMP and RESTORE statements to restore nonVSAM data sets on the volume. (See *OS/VS Utilities* for details.)
     - Execute the IEHDASDR program with the ANALYZE statement to initialize the volume. (See *OS/VS Utilities* for details.)
     - Issue the ALTER REMOVEVOLUMES command to remove the VSAM catalog's ownership of the volume.
  4. Issue the IMPORTRA command to reload the data copied and moved during step 2 above.

- The RESETCAT Method: If you don't want your data to be moved and if you wish to confine all updating to the catalog (and CRAs), you should consider this approach. RESETCAT does not permit selective reset of specific catalog entries. An entire volume's worth of catalog entries are reset. You would use RESETCAT if a catalog or one or more of its owned volumes become inaccessible. Restore the volume(s) from a backup copy and issue RESETCAT to provide the necessary consistency between the catalog and the restored volume(s).

The LISTCRA, EXPORTRA, IMPORTRA, and RESETCAT commands are described in detail in the sections that follow.

Listing the Catalog Recovery Area's Contents

If your catalog was created with the RECOVERABLE option, you can determine the damage (if any occurred) that was done to your catalog when a system or hardware failure occurred. This is because a recoverable catalog maintains a copy of each catalog entry in a separate part of the volume, called the *catalog recovery area*.

When you issue the LISTCRA command with the COMPARE parameter, Access Method Services compares each catalog entry to its copy in the catalog recovery area. You can use the LISTCRA output listing to determine which catalog entries are no longer accurate, and to help you code the ENTRIES subparameter of the EXPORTRA command.

To use the LISTCRA command, Access Method Services must be authorized. See "Authorized Program Facility (APF)" in *OS/VS2 System Programming Library: Supervisor* for information about program authorization.

The types of output listing that the LISTCRA command can produce, depending on the optional parameters, are:

- A list of the name and *volser* of each entry, and the name and *volser* of each related entry, in the catalog recovery area. The entries are listed in alphanumeric order by group type (NOCOMPARE and NAME).
- A full dump (that is, hexadecimal and character listing) of each entry and its related entries in the catalog recovery area. The entries are listed in alphanumeric order by group type (NOCOMPARE and DUMP).

- A list of the name of each catalog entry and its *volser* whose data doesn't compare equally with the entry's copy in the catalog recovery area, and an indication of the type of information that miscompares. The miscompared entries are listed in alphanumeric order by group type. The entries that compare equally are not listed (COMPARE and NAME).
- A full dump (that is, hexadecimal and character listing) of each catalog record whose contents doesn't compare equally with the record's copy in the catalog recovery area, and asterisks below each byte that miscompares. The miscompared entries are listed in alphanumeric order by group type. The entries that compare equally are not listed (COMPARE and DUMP).
- A full dump (that is, hexadecimal and character listing) of all entries in the catalog recovery area in sequential order as they occur in the CRA (SEQUENTIALDUMP).

Refer to chapter: "Command Format" for a description of the command format of LISTCRA. An example is also included.

Copying a Catalog Entry From the Catalog Recovery Area

When you discover that your catalog is partially or completely damaged due to a system failure or hardware problem, you want to reconstruct the damaged catalog entries so that you can access the cataloged object's data. You can rebuild a catalog entry by issuing the IMPORT command when you have recently made a copy of the object with the EXPORT command. The IMPORT command replaces the damaged catalog entry with its copy in the exported file. (See "Moving Entries" for details about the EXPORT and IMPORT commands.)

An exported copy of your cluster might not exist, or the one you have might not be current. The volume that contains your data might not be damaged but, because its catalog entry is, you can't use the catalog entry to locate and access your data. What you want to do is replace the damaged catalog entry with its undamaged copy in the catalog recovery area and, when the entry is for a VSAM data set, to move the contents of the data set to a volume owned by an undamaged catalog. You can use the EXPORTRA command to obtain the catalog entry's copy from the catalog recovery area, then use the IMPORTRA command to replace the damaged entry with its copy.

If an entire VSAM volume becomes unusable, and a backup copy of the volume exists, you may want to consider using RESETCAT rather than EXPORTRA to reset your catalog so that it will correctly access the VSAM data sets on the restored volume. See "Resetting Catalog Entries."

To use the EXPORTRA and RESETCAT commands, Access Method Services must be authorized. See "Authorized Program Facility (APF)" in *OS/VS2 System Programming Library: Supervisor* for information about program authorization.

When a VSAM data set is recovered using the EXPORTRA and IMPORTRA commands, the result is the same as if the data set were backed up using the EXPORT and IMPORT commands. The differences in the process used to achieve this result are:

- The data set's catalog entry is copied from a catalog recovery area instead of the VSAM catalog.
- Many data sets can be recovered with one issuance of the EXPORTRA and IMPORTRA commands.

The EXPORTRA command copies VSAM catalog entries and data sets to a movable storage device (that is, a magnetic tape or demountable disk pack). Other capabilities of the EXPORTRA command that aren't available as functions of the EXPORT command are:

- You can obtain a copy of each entry in the catalog recovery area by issuing the EXPORTRA command once. Subsequently, you can replace all entries exported with the EXPORTRA command by issuing the IMPORTRA command once.
- You can obtain the copy of one or more entries, as you specify, without obtaining the rest of the catalog recovery area.
- When the catalog owns more than one volume, you can obtain the copy of each entry from one volume's catalog recovery area without obtaining entries from the catalog recovery areas of other volumes.
- You can obtain a copy of nonVSAM entries and their alias associations, if any.
- You can obtain a copy of generation data group entries and their nonVSAM associations, if any.

If the object has been defined with a maximum record length greater than 32,760, EXPORTRA processing terminates with an error message.

Refer to the chapter: "Command Format" for a description of the command format for EXPORTRA. Examples of EXPORTRA are included with the EXPORTRA command format in the same chapter.

Restoring the Catalog Entry That Was Obtained Using the EXPORTRA Command

The IMPORTRA command is used to reestablish in a catalog those objects that reside in the portable data set created by a previously issued EXPORTRA command. When an existing catalog entry is found with the same entryname as the object in the portable data set, the existing entry is first deleted. The object is redefined in the catalog, using information from the portable data set.

VSAM data sets, their associated data and index components, and any paths over them, alternate indexes, their associated data and index components, and any paths over them, nonVSAM data sets, and generation data group entries are automatically defined in the catalog selected by the user. User catalog connector entries (which can be exported only from the system's master catalog) are connected to the master catalog. (Existing user catalog connector entries with the same entryname as the imported entry are disconnected rather than deleted; the imported user catalog connector entry is then reconnected.) The aliases of user catalog and nonVSAM entries are also redefined by IMPORTRA.

Refer to the chapter "Command Format" for a description of the command format for IMPORTRA. Examples of EXPORTRA and IMPORTRA are included.

Resetting Catalog Entries

When you define a catalog as recoverable, each volume owned by the catalog contains a catalog recovery area (CRA). The CRA contains duplicate information for catalog entries associated with that volume. You can use the RESETCAT command when a recoverable catalog or one or more of its owned volumes becomes inaccessible. You can restore the inaccessible volume(s) from a backup copy and execute the RESETCAT command. The CRAs contain enough information to reset the catalog entries, and VSAM data sets owned by that catalog can again be accessed correctly.

Unlike the EXPORTRA/IMPORTRA command, the RESETCAT command is a one-step operation that enables you to recover your catalog without movement of data. The RESETCAT command does not check or process the data itself, but compares catalog entries with CRA entries and resets as necessary. You are responsible for ensuring that the data is at the correct level for your use.

If a VSAM volume becomes inaccessible, and a backup copy of the volume is used to restore the volume to a previous level, the volume and the catalog may no longer be synchronized. A list created by the LISTCRA command (with COMPARE option) can indicate mismatches that require the RESETCAT command to be run. The RESETCAT command can synchronize the catalog with the volume. After access to the data has been regained, the data sets on that volume can be brought up to the current level by rerunning the jobs that were run after the backup was taken.

If a recoverable catalog becomes unusable, use the LISTCRA command to help analyze the problem (also see "Catalog Recovery"). If you are unable to access your data, restore the catalog volume. Then run the RESETCAT command to synchronize the catalog with its owned volumes. If volumes have been added since the catalog backup was made, RESETCAT can build these entries in the catalog from the volume's CRA. If volumes have been deleted since the last backup, use the DELETE SPACE (FORCE) command to delete the volume's space entries in the catalog and delete the data sets that resided on those volumes that are now marked unusable in the catalog.

If your catalog becomes unusable and no backup copy is available, you can use RESETCAT to recover all of your catalog entries:

If the catalog is a user catalog, remove the catalog connector entry of your catalog from the master catalog via EXPORT DISCONNECT, and define a catalog with the same name on a different volume with the RECOVERABLE attribute. Volumes owned by the unusable catalog should *not* be included as owned by the new catalog. The DEFINE operation would flag this as an error condition, because volumes are already owned.

With this new catalog, issue RESETCAT, specifying all volumes owned by the previous catalog (including the unusable catalog's resident volume) for reset. Because the new catalog name is the same as the old catalog name, all entries in the specified CRAs will be added to the new catalog (including volume entries).

While the catalog is always updated during RESETCAT processing, the CRA can also be updated under certain circumstances. If some external event such as a power failure were to cause RESETCAT to fail, partial updates

to the catalog and CRA(s) may have taken place. Therefore the catalog and any CRA volumes being reset should be restored before RESETCAT is rerun. It is advisable, therefore, for you to have backup volumes of your catalog and CRA(s) before you use RESETCAT.

RESETCAT Requirements

In planning to use RESETCAT, you should be aware of the following requirements:

- The catalog being reset must be capable of being opened, and it must have the RECOVERABLE attribute. It may or may not have valid entries.
- The CRAs must be capable of being opened. Entries not related to the CRA itself may be inaccessible.
- The CRAs must have been created by a recoverable catalog with the same name as the catalog being reset.
- The catalog must be extendable in the event that it becomes enlarged as a result of the reset operation.
- If the master catalog is password protected, the master password of that catalog is required.
- The master catalog may not be reset while it is in use as a master catalog.
- You need to use caution when using RESETCAT to recover accessibility of a volume which contains a portion of a multivolume file. Prior to issuing RESETCAT, compatible levels of volumes containing multivolume files should be restored. See "Considerations for Multivolume Files" in this chapter for more details.

WORKFILE Space Requirements

The RESETCAT command requires a temporary work file for use as temporary storage while processing the command. It is defined by the RESETCAT command and deleted at the end of command processing. The space required is suballocated from VSAM data spaces on the volumes assigned on the DD statement for the WORKFILE parameter. Under normal conditions (no extensions), the amount of space required will be no larger than the resultant catalog. You can determine this by a LISTCAT listing of that catalog.

If the catalog must be extended as a result of RESETCAT processing (this may occur when the catalog is restored at a lower level than its owned volumes), enough data space must be provided to allow for this extension. The space required for each extension is 6603 records, where the record size is 505 bytes. An additional 7 bytes per record will be required; the CI size will be set at 512 bytes.

Considerations for Multivolume Data Sets

The contents of a volume's CRA depend on the types of objects the volume contains. It is important for you to understand on which CRA the catalog information resides for a particular object. Figure 6 in this chapter identifies by object type the location of the CRA.

The primary CRA contains all of the catalog records necessary to describe the object. Hence, for an entry sequenced data set on two volumes, the volume that contains the first part of the entry sequenced data set contains all the records that describe the entry sequenced data set (including its allocation on the second volume). The second volume, a secondary CRA for this object, contains information that shows that the entry sequenced data set is allocated on the second volume. If the second volume has an I/O error that renders it useless and a previous version of that volume is restored, the present catalog information may be erroneous; that is, the catalog may reflect the data set's extent on the second volume, whereas there is no longer an extent on the second volume. Prior to issuing a RESETCAT command, you should restore compatible levels of volumes containing multivolume data sets. RESETCAT would then be issued to reset the catalog to reflect the restored level of *all* data sets on all reset volumes.

If, in the above example, the second volume was restored and RESETCAT specified only that volume as a reset volume, the entry sequenced data set may be marked unusable for that volume and the space allocated to it would be either scratched or returned to the catalog for suballocation. The primary description is on the first volume, which was not indicated as one to reset. The description of the data set used would be the description that currently resides in the catalog. If the data set is defined differently on the second volume (for example, extents don't match), the data set is marked unusable for that volume and the allocated space marked free.

For a multivolume entry sequenced data set, a multivolume key-sequenced data set, or an alternate index defined on a volume different from the data set it is based on, minimizing the intersection of different multivolume data sets on a common volume will permit better use of RESETCAT.

When all volumes of a multivolume VSAM data set or structure are *not* specified in the RESETCAT operation, the extent of checking depends on whether the primary CRA volume is specified for reset. If the primary CRA volume is specified for reset, all information in the catalog is replaced for the data set concerned. For all volumes of the multivolume data set, whether specified or not, the following consistency checks are made by RESETCAT:

- Check the current catalog (if the volume is not specified) or the CRA (if the volume is specified) to ensure that the data set is defined on the volume.
- Check the data set specified on each volume. Was it defined at the same time as the one specified in the primary CRA?
- Check the extents described on the volumes. Are they still allocated to the multivolume VSAM data set?

Although the above checks guarantee that the catalog physically describes a data set correctly, these checks cannot guarantee that the level of data in the data set is at a consistent level. For instance, if a multivolume keyed sequential data set was defined with the data on one volume and the index on another, the same Define-time would be associated with both. If, over some time, several additions, deletions, and updates were made without causing an

extension of the data set, RESETCAT would be unable to distinguish among different combinations of volumes taken from this time period. Since the index contains direct VSAM pointers to the data, an inconsistent combination may cause errors.

If the primary CRA volume is not specified for reset, the scope of checking is limited to volumes specified in the reset. The current catalog is checked to ensure that the current catalog entry describes the part of the data set on the reset volume. Hence, only verification (no reset) occurs for these partial entries. The check ensures that the part of the data set on the reset volume resides on the same physical place as described in the current catalog and is part of the same definition as the data set described in the current catalog. RESETCAT cannot guarantee that the level of data *in* the data set is at a consistent level among different volumes.

JCL Requirements

For the catalogs required during RESETCAT processing, the catalog being reset (indicated by the CATALOG parameter of the RESETCAT command) must appear in a STEPCAT or JOBCAT DD statement. The catalog in which the work file is defined (optionally indicated by the WORKCAT parameter) must appear in a STEPCAT or JOBCAT DD statement. If the WORKCAT parameter is not specified, the work file will be defined in the catalog specified in the first concatenation of the STEPCAT or JOBCAT DD statement.

The relationship of the STEPCAT or JOBCAT DD statements is summarized below:

(1) If you want the work file defined in a catalog other than the master catalog, then that catalog must be pecified first in the JOBCAT or STEPCAT DD statement concatenation.

```
//STEPCAT  DD  DSN=workcat,DISP=SHR
//         DD  DSN=resetcat,DISP=SHR
```

The above example specifies the work file catalog and reset catalog, respectively, in the JOBCAT or STEPCAT DD statement for case (1).

(2) If you want the work file defined in the master catalog, then the master catalog must be specified by name via the WORKCAT parameter and, if specified in the STEPCAT or JOBCAT DD statement, must appear last in the concatenation sequence.

```
//STEPCAT  DD  DSN=resetcat,DISP=SHR
//         DD  DSN=mastcat,DISP=SHR
```

The above example specifies the reset catalog and master catalog, respectively, in the JOBCAT or STEPCAT DD statement for case (2).

Further, the catalog being reset must be used by the RESETCAT command as a data set. A separate DD statement may be used for this catalog. It should specify only the catalog being reset and should not be concatenated to another catalog. For example:

```
//DDCAT  DD  DSN=catname,DISP=OLD
```

DISP=OLD should be specified to ensure exclusive use of the catalog. If no DD statement is supplied, it will be dynamically allocated.

For CRAs, a single DD statement is required for each volume containing a CRA, if CRAFILES parameter of the RESETCAT command is specified. For example:

```
//DDCRA1   DD   UNIT=3330,VOL-SER-XYZ,DISP=SHR
//DDCRA2   DD   UNIT=3314.VOL=SER=ABC,DISP=SHR
```

Unit affinity may be specified to reduce unit requirements. No two CRAs will be demanded concurrently by RESETCAT.

For the work file, RESETCAT command requires a list of one or more volumes to define a temporary VSAM data set; no more than five volumes may be specified. If no data set name is provided, a system-generated data set name will be used. The following is an example of a work file DD statement:

```
//FILEW   DD   DSN=A.WFILE,UNIT=(3330,2),VOL=SER=(X,Y),
//             DISP=OLD,AMP='AMORG'
```

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.
- Copy an alternate index as though it were a key-sequenced VSAM data set.
- Copy a SAM, ISAM, or VSAM data set (whose records are fixed-length) into an empty VSAM relative-record data set.
- Convert a sequential or indexed-sequential data set into a VSAM data set.
- Convert a VSAM or indexed-sequential data set into a sequential data set.
- Copy a data set—other than a catalog—to reorganize it. Data set reorganization is an automatic feature.
- Copy a catalog from one volume to another, for example, from a 2314 volume to a 3330 volume.
- Make a backup copy of a catalog.
- Reload a backup copy of a catalog.
- Merge two VSAM data sets.

Throughout the remainder of the REPRO discussion, all of these functions will be referred to as copying.

Refer to the chapter: "Command Format" for a description of the command format for REPRO. Several examples are included.

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 7 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, sequential data set, or relative-record 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. (DISP=MOD must be specified for SAM sequential.)
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, unless you specify the REPLACE option.
Relative-Record	Creates a new data set in relative-record sequence, beginning with 1.	Records from another relative-record data set are merged, keeping their old record numbers. A new record whose number duplicates an existing record number is lost, unless you specify the REPLACE option. Records from any other type of organization cannot be copied into a nonempty relative-record data set.

Figure 7. Adding Records To Various Types of Output Data Sets

Note: If four or more non-recoverable errors are encountered in the copy operation, the copying is terminated.

If a VSAM data set defined with a record length greater than 32,760 is to be copied to a sequential data set, REPRO processing terminates with an error message.

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 record-numbers from a sequential, indexed-sequential, or VSAM data set and store them in VSAM format in key sequence, record-number 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 (defined). JCL can be used to allocate the required data set(s) and to 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.

To name a data set, using JCL, code:

//dname **DD DISP=OLD,DSNAME=** *dsname*

Copying a Catalog

The REPRO command can be used to copy a catalog from one volume to another. When using REPRO to copy a catalog, both the input and the output objects must be catalogs (that is, you must first define a catalog on the device that is to contain the newly-copied catalog). A catalog might be copied to move it to a faster device type or to optimize the catalog's allocation.

You cannot copy the contents of a nonrecoverable catalog into a recoverable catalog, and vice versa, nor can you copy a recoverable catalog into a recoverable catalog. To convert a nonrecoverable catalog into a recoverable catalog, you must export each VSAM data set (from the nonrecoverable catalog), then import it into the newly defined (recoverable) catalog. To convert a recoverable catalog into a nonrecoverable catalog or to copy a recoverable catalog into a recoverable catalog, you can either export each VSAM data set from the source catalog and then import it into the target catalog or use the EXPORTRA/IMPORTRA commands to accomplish the same thing on a volume basis.

To use the REPRO command to copy a catalog, Access Method Services must be authorized. See "Authorized Program Facility (APF)" in *OS/VS2 System Programming Library: Supervisor* for information about program authorization.

Copy-Catalog Preparation

In preparation for copying a 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.

There is no method (besides dumping the catalog) that enables you to determine the size of the catalog's index set or high-keyrange. You can determine the size (that is, number of data records) of the low-keyrange. (When you define a catalog and include DATA(RECORDS...), the number of records you specify is the number of data-records you want the low-keyrange to contain, unless you also specify an amount of space as a subparameter of INDEX.) One of the catalog's self-describing records (in the low-keyrange) is called the catalog control record, or CCR. The CCR describes the free (or unused) control intervals in the catalog, and contains statistics about how the catalog is used. To print the CCR, issue the PRINT command (as shown below), specifying the name of the catalog as the input data set.

```
PRINT     INDATASET( catname ) -
          SKIP( 3 ) -
          COUNT( 1 )
```

When *catname* identifies a user catalog, your job must include a JOBCAT or STEPCAT DD statement to describe and allocate the catalog.

The CCR is printed in the "dump" format (an example of this format is included with the first example in "PRINT Examples"). The record's first 44 bytes are its key value. The next byte is the character "L," which identifies the record as a CCR-type record. The next nine bytes are three 3-byte fields whose hexadecimal values specify:

Displacement	Symbol	Contents
45 (X'2D')	A	The highest control interval number that can be assigned.
48 (X'30')	B	The next control interval number to be assigned.
51 (X'33')	C	The number of deleted records (that is, the number of control intervals that are unused because the entry they contained was deleted).

The number of records in the catalog is the same as the number of control intervals (in the low key range) that currently contain catalog records, or "B–C". When you specify DATA(RECORDS([B–C+15])) (the constant"15" is added to allow enough space for the receiving catalog's self-describing records; after you copy the source catalog into the receiving catlog, you will delete the copy of the source catalog's self-describing records), you are defining the smallest possible receiving catalog because the value you specify doesn't allow for any of the source catalog's free records or for any unassigned control intervals. When you specify the minimum value for your catalog's primary space allocation, you should also specify a secondary allocation amount—the catalog will have to be extended almost immediately when an entry is added to it.

The symbols A, B, and C have no meaning except in the formula "B–C", which yields the number of control intervals that contain catalog records in the low-keyrange. The complete format of the CCR record is included in *OS/VS2 Independent Component: Catalog Management Logic.* See *Planning for Enhanced VSAM Under OS/VS* for more details about the catalog's structure and the low key-range.

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 catalog is to be copied. The receiving 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. A concatenated JOBCAT or STEPCAT DD statement is required to describe and allocate both the source and receiving catalogs. When the master catalog is being copied (that is, the source is the master catalog), use a single JOBCAT or STEPCAT DD statement to describe and allocate the receiving catalog. A DD statement is not required to describe and allocate the master (source) catalog.
3. Use the EXPORT command to disconnect the source user catalog from the master catalog.

You should take care to list the user catalog's aliases before you issue the EXPORT command. You can use the LISTCAT command to list the aliases:

```
LISTCAT -
     CATALOG( master catname / password ) -
     ENTRIES( user catname ) -
     ALL
```

The user catalog's aliases are listed under the heading "ASSOCIATIONS."

When you use the IMPORT command to make the catalog available to another system, you can issue the DEFINE ALIAS command to reestablish each of the catalog's aliases.

4. Use the DELETE command to remove the source catalog from the receiving volume. The source catalog appears in the receiving catalog as a cluster as a result of the copy operation.

The copy-catalog procedure should be performed with caution. Until the source catalog is disconnected from the master catalog, two 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 redefine the catalog and use REPRO to reload the backup to replace it. If the catalog is accessible, you can use REPRO to reload the backup and re-establish catalog entries. Reloading a catalog should be done judiciously since it is difficult to recover VSAM data spaces, page spaces, and data sets that have extended after a backup copy of the catalog was made.

Using REPRO to unload or reload a password-protected catalog requires the catalog's master password.

The parameters that limit the extent of copying are invalid for unload/reload. Parameters that indicate action on the output data set are also invalid. These parameters are: COUNT, FROMKEY, FROMNUMBER, FROMADDRESS, SKIP, TOKEY, TOADDRESS, TONUMBER, REUSE and REPLACE.

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.)

To use the REPRO command to unload or reload a catalog, Access Method Services must be authorized. See "Authorized Program Facility (APF)" in *OS/VS2 System Programming Library: Supervisor* for information about program authorization.

Unloading a Catalog

A sequential, key-sequenced, or entry-sequenced backup copy of a catalog is inaccessible as a catalog. Because there is little 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 must 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. Reloading the master catalog has special requirements (for details, see the section "Backing Up the Master Catalog" in the chapter "Data Security and Protection.")

The target catalog can be either a version of the original catalog (which might have been obtained using IEHDASDR RESTORE), or a newly defined user catalog (after using EXPORT DISCONNECT to remove the user catalog entry in the Master Catalog). The newly defined catalog must be able to hold at least as many entries as the original catalog held at the time the backup copy was made.

Reloading a version of the original catalog 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.

Reloading a newly defined catalog has the same results as reloading a version of the original catalog, with one exception. The newly defined catalog's Volume record contains only self-defining information. A check is initiated when reload opens the catalog and, if the catalog is new, a reload of the original version of the Volume record is bypassed. (The assumption is that, if the catalog is new, no other VSAM data space exists on the volume under normal conditions.) The Volume record of the original version of the catalog contains all the data space information previously on the volume at the time the catalog was unloaded. But, when the reload of the Volume record is bypassed, the data space information is lost. After the reload of a newly defined catalog, the entries in the reloaded catalog reference data sets previously existing on the volume. If these data sets still exist, they can be accessed, but any attempt to extend the data space in which they reside will fail. In this situation, you can restore all needed information to the Volume record by, first, using EXPORT PERMANENT to remove the data set entries from the new catalog, then define a data space large enough to accommodate the data sets, and then use IMPORT to put the data sets into the newly defined data space.

After you reload a catalog, use LISTCAT to list its contents. Run LISTCAT in a separate job 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.

When reloading or restoring a recoverable catalog, the LISTCRA command with COMPARE option should be run to identify mismatches between the catalog and the catalog recovery area (CRA). These mismatches should be resolved as necessary before the catalog can be used. See the section "Catalog Recovery" in the "Data Security and Protection" chapter.

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

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

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

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

See the section "Updating a Backup Catalog" in the chapter "Data Security and Protection" 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.

Printing Data Sets

You use the PRINT command to list part or all of a key-sequenced, relative-record, or entry-sequenced VSAM data set, an alternate index, or a nonVSAM data set. The components of a key-sequenced data set or alternate index can be listed individually. To list a component of a key-sequenced data set or alternate index, specify the component name as the data set name. An alternate index is printed as though it were a key-sequenced cluster.

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. A base cluster can be listed in alternate-key sequence by specifying a path name as the data set name for the cluster.

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

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

Note: If four or more non-recoverable errors are encountered while trying to read the input, the printing is terminated.

To use the PRINT command to print a VSAM catalog, Access Method Services must be authorized. See "Authorized Program Facility (APF)" in *OS/VS2 System Programming Library: Supervisor* for information about program authorization.

Refer to the chapter: "Command Format" for a description of the command format for PRINT. Examples of the PRINT command are included.

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.

Refer to the chapter: "Command Format" for a description of the command format for CHKLIST. Examples of the CHKLIST command are also included.

COMMAND FORMAT

This chapter sets out the functional command formats. Parameters are grouped into categories such as name, data organization, allocation, and protection and integrity. The format of each command is then shown, followed by a discussion of each parameter. Examples of each command follow the parameter discussion.

Functional Command Format

This section provides complete reference information about all functional commands of Access Method Services. The commands discussed in the section are:

- ALTER command, which is used to alter attributes of data sets and other objects that have already been defined.
- BLDINDEX command, which is used to build an alternate index over a base cluster.
- CHKLIST command, which lists tape data sets opened during a checkpoint.
- CNVTCAT command, which converts OS catalog entries to VSAM catalog entries and merges them into a VSAM catalog.
- DEFINE ALIAS command, which is used to define an alternate name for a nonVSAM data set or a user catalog.
- DEFINE ALTERNATEINDEX command, which is used to define an alternate index.
- DEFINE CLUSTER command, which is used to define a cluster for a key-sequenced, entry-sequenced, or relative-record data sets.
- DEFINE GENERATIONDATAGROUP command, which is used to create a catalog entry for a generation data group.
- DEFINE NONVSAM command, which is used to define a catalog entry for a nonVSAM data sets.
- DEFINE PAGESPACE command, which is used to define an entry for a page space data set.
- DEFINE PATH command, which is used to define a path directly over a base cluster or a path over an alternate index and its related base cluster.
- DEFINE SPACE command, which is used to define a VSAM data space.
- DEFINE USERCATALOG command, which is used to define a VSAM user catalog.
- DELETE command, which is used to delete data sets and other objects, including catalogs and nonVSAM data sets.
- EXPORT command, which is used to export VSAM files and to disconnect user catalogs.
- EXPORTRA command, which is used to recover VSAM and nonVSAM catalog entries from catalog recovery areas and, for VSAM objects

(clusters and alternate indexes), to recover the data itself by means of catalog recovery areas.

- IMPORT command, which is used to import VSAM data sets and to connect user catalogs.
- IMPORTRA command, which is used to reconstruct multiple VSAM data sets from a data set created by the EXPORTRA command.
- LISTCAT command, which is used to list catalog entries.
- LISTCRA command, which is used to list or compare the contents of a given catalog recovery area.
- PRINT command, which is used to print both VSAM and nonVSAM data sets.
- REPRO command, which is used to copy and load both VSAM and nonVSAM files and VSAM catalogs.
- VERIFY command, which is used to verify and correct certain problems that have made your data set unusable.

See "Notational Conventions" in the "Preface" for an explanation of the symbols used in the command formats.

ALTER

The format of the ALTER command is:

ALTER	*entryname* [/ *password*] **[FILE(** *dname* **)]** **[NEWNAME(** *newname* **)]** **[NULLIFY(** **[MASTERPW]** **[CONTROLPW]** **[UPDATEPW]** **[READPW]** **[OWNER]** **[AUTHORIZATION(MODULE | STRING)]** **[EXCEPTIONEXIT]** **[RETENTION]** **[CODE])]** **[MASTERPW(** *password* **)]** **[CONTROLPW(** *password* **)]** **[UPDATEPW(** *password* **)]** **[READPW(** *password* **)]** **[UNINHIBIT | INHIBIT]** **[CODE(** *code* **)]** **[ATTEMPTS(** *number* **)]** **[AUTHORIZATION(** *entrypoint* [ƀ *string*])]** **[EXCEPTIONEXIT(** *entrypoint* **)]** **[OWNER(** *ownerid* **)]** **[TO(** *date* **) | FOR(** *days* **)]** **[SHAREOPTIONS(** *crossregion* [ƀ *crosssystem*])]** **[ERASE | NOERASE]** **[WRITECHECK | NOWRITECHECK]** **[DESTAGEWAIT | NODESTAGEWAIT]** **[FREESPACE(** *CI-percent* [ƀ *CA-percent*])]** **[BUFFERSPACE(** *size* **)]** **[ADDVOLUMES(** *volser* [ƀ *volser* ...])]** **[REMOVEVOLUMES(** *volser* [ƀ *volser* ...])]** **[STAGE | BIND | CYLINDERFAULT]** **[RECORDSIZE(** *average* ƀ *maximum* **)]** **[KEYS(** *length* ƀ *offset* **)]** **[UNIQUEKEY | NONUNIQUEKEY]** **[UPDATE | NOUPDATE]** **[UPGRADE | NOUPGRADE]** **[EMPTY | NOEMPTY]** **[SCRATCH | NOSCRATCH]** **[CATALOG(** *catname* [/ *password*])]

ALTER Parameters: Summary

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. These parameters are also used to rename a member of a nonVSAM partitioned data set.
- Protection and integrity, which describes the NULLIFY, MASTERPW, CONTROLPW, UPDATEPW, READPW, UNINHIBIT, INHIBIT, CODE, ATTEMPTS, AUTHORIZATION, EXCEPTIONEXIT, OWNER, TO, FOR, SHAREOPTIONS, ERASE, NOERASE, 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, RECORDSIZE, KEYS, 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 of how a data or index component that is stored on a mass storage volume is to be staged.

  When a data set doesn't contain any data records, RECORDSIZE can be specified to modify the data set's average and maximum recordsizes. When a key sequenced data set or alternate index is empty, KEYS can respecify the the length and position of each data record's key field.
- Alternate index and path, which describes the UNIQUEKEY, NONUNIQUEKEY, UPDATE, NOUPDATE, UPGRADE, and NOUPGRADE parameters. These parameters specify attributes that apply only to alternate index and path entries.
- 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 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.

"Appendix F: Command Parameters Summary" contains a table for the ALTER command that shows each parameter, its abbreviation, and an example of its use.

Entry-Types That Can Be Altered

An "X" in Figure 8 indicates you can respecify the value or attribute for that type of VSAM catalog entry. Some attributes, when specified for a cluster or alternate index, apply to its data or index component entry only, and not to the cluster or alternate-index entry. Some attributes can only be specified for the cluster's or alternate-index's data or index component entry—you must identify the component's entry with its entryname. You can use the Access Method Services LISTCAT command to determine the names VSAM generated for the object's components.

When you identify a group of entries with a generic name, entries whose entrynames match the supplied qualifiers are altered, if they contain the type of information specified with the ALTER command's other parameters.

You cannot identify the following types of entries: alias entries, a catalog's data or index component entries, and a master catalog's self-describing entries.

ALTER Parameters

Name (ALTER)

Name parameters are used to identify and rename catalog entries and to rename members of nonVSAM partitioned data sets.

entryname [/ *password*]
is a required parameter that names the entry to be altered and supplies a password.

If you are renaming a member of a nonVSAM partitioned data set, the *entryname* must be specified in the format: *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.

FILE(*dname* **)**
When the entry to be altered is in a recoverable catalog, *dname* identifies a DD statement that describes the volume whose catalog recovery area contains the entry's copy. If FILE is not specified, the catalog recovery area volume is dynamically allocated. It must be mounted as permanently resident or reserved.

FILE can be used to specify the name of a DD statement that identifies the entry to be altered. If a nonVSAM or unique data set, or a unique alternate index or pagespace is to be renamed and if FILE is not specified, the object's volume is dynamically allocated. The object's volume must be mounted as permanently resident or reserved. See "Restoring Catalog Entries After System Failure" for a description of catalog recovery area contents.

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.

If the object being renamed is a unique cluster or component, or a nonVSAM data set, the object's volume must be mounted because the volume's VTOC is modified. You can use a JCL DD statement (see the FILE parameter) to cause the object's volume(s) to be allocated. If you don't supply a DD statement, the object's volume(s) is dynamically

	Type of Catalog Entry: ALTERNATE		INDEX	CLUSTER							
		DATA	INDEX		DATA	INDEX	PGSPC	PATH	UCAT	NonVSAM	GDG
ADDVOLUMES		X	X		X	X					
ATTEMPTS	X	X	X	X	X	X	X	X	X		
AUTHORIZATION	X	X	X	X	X	X	X	X	X		
BIND		X	X		X	X					
BUFFERSPACE		X			X				X		
CODE	X	X	X	X	X	X	X	X	X		
CONTROLPW	X	X	X	X	X	X	X	X	X		
CYLINDERFAULT		X	X		X	X					
DESTAGEWAIT		X	X		X	X					
EMPTY											X
ERASE		X			X						
EXCEPTIONEXIT		X	X		X	X					
FILE	X	X	X	X	X	X	X	X	X	X	
FOR	X			X			X	X	X	X	X
FREESPACE		X			X						
INHIBIT		X	X		X	X					
KEYS	X	X		X	X			X			
MASTERPW	X	X	X	X	X	X	X	X	X		
NEWNAME	X	X	X	X	X	X	X	X		X	
NODESTAGEWAIT		X	X		X	X					
NOEMPTY											X
NOERASE		X			X						
NONUNIQUEKEY		X									
NOSCRATCH											X
NOUPDATE								X			
NOUPGRADE	X										
NOWRITECHECK		X	X		X	X					
NULLIFY	X	X	X	X	X	X	X	X	X	X	X
AUTHORIZATION	X	X	X	X	X	X	X	X	X		
CODE	X	X	X	X	X	X	X	X	X		
CONTROLPW	X	X	X	X	X	X	X	X	X		
EXCEPTIONEXIT		X	X		X	X					
MASTERPW	X	X	X	X	X	X	X	X	X		
MODULE	X	X	X	X	X	X	X	X	X		
OWNER	X	X	X	X	X	X		X	X		
READPW	X	X	X	X	X	X	X	X	X		
RETENTION	X			X			X	X	X	X	X
STRING	X	X	X	X	X	X	X	X	X		
UPDATEPW	X	X	X	X	X	X	X	X	X		
OWNER	X	X	X	X	X	X		X	X		
READPW	X	X	X	X	X	X	X	X	X		
RECORDSIZE	X	X		X	X			X			
REMOVEVOLUMES		X	X		X	X					
SCRATCH											X
SHAREOPTIONS		X	X		X	X					
STAGE		X	X		X	X					
TO	X			X			X	X	X	X	X
UNINHIBIT		X	X		X	X					
UNIQUEKEY		X									
UPDATE								X			
UPDATEPW	X	X	X	X	X	X	X	X	X		
UPGRADE	X										
WRITECHECK		X	X		X	X	X				

Figure 8. ALTER Parameters and the Entry-Types to Which Each Applies

allocated. The volume(s) must be mounted as either permanently resident or reserved.

If you specify generic names, you specify both the entryname and the newname as generic names. See "Generic Names and ALTER" for details on generic names.

If you are renaming a member of a nonVSAM partitioned data set, the *newname* must be specified in the format: *pdsname (membername).*

If you are renaming a VSAM data set which is RACF protected, the existing RACF data set profile will be renamed. If a data set profile already exists for the new data set name prior to the ALTER command, the command is terminated and the data set name and/or protection attributes remain unchanged. If the old profile cannot be found or cannot be altered to the NEWNAME, the NEWNAME action will not be completed in the catalog, and an error message will indicate the reason for non-completion.

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 subparameters of NULLIFY are to be nullified. Attributes are 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.

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.

CONTROLPW
specifies that the control password is to be nullified.

UPDATEPW
specifies that the update password is to be nullified.

READPW
specifies that the read password is to be nullified.

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. When you nullify the authorization record, the corresponding module is not nullified.

EXCEPTIONEXIT
specifies that the entry's exception exit is to be nullified. The module name is removed from the catalog record, but the exception exit routine itself is not deleted.

RETENTION
specifies that the retention period, specified in a TO or FOR parameter, is to be nullified.

CODE
specifies that the code name used for prompting is to be nullified.

MASTERPW(*password*)
specifies a master password for the entry being altered. For more details about the object's master password, see the DEFINE command that defines the object.

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.

CONTROLPW(*password*)
specifies a control password for the entry being altered.

UPDATEPW(*password*)
specifies an update password for the entry being altered.

READPW(*password*)
specifies a read 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 (X'password'), where two hexadecimal characters represent an EBCDIC character.

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.

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 or TSO terminal user 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 (X'code'), where two hexadecimal characters represent an EBCDIC character.

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 or TSO terminal user 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. If 0 is specified, the operator is not prompted and is not allowed to enter a password from the console. This parameter can be coded, but only has effect when the entry's master password is not null. A prompting message is issued only when the user hasn't already supplied the appropriate password.

number
is an integer from 0 to 7 and can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Note to TSO users: At a TSO terminal, the logon password is checked first before the user is prompted to supply a password. 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 password because the default is 2.

AUTHORIZATION(*entrypoint* [b *string*])
specifies that a user-security-verification routine (USVR) is available for additional security verification.

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 up to 255 bytes of EBCDIC information that is to be passed to the user's security verification routine when the USVR receives control.

The string can contain up to 255 characters. It 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. The string can be coded in hexadecimal form X'string', where two hexadecimal characters represent an EBCDIC character.

EXCEPTIONEXIT(*entrypoint* **)**
specifies the name of the user-written routine that receives control when an exception (usually an I/O error) occurs while the entry's object is being processed. An exception is any condition that causes a SYNAD exit to be taken. The object's exception exit routine is processed first, then the user's SYNAD exit routine receives control.

You cannot specify this parameter to add an exception exit routine to a VSAM cluster that is cataloged in a VSAM catalog created in an OS/VS2-2 (or lower release level) system.

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 (X'ownerid'), where two hexadecimal characters represent an EBCDIC character.

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the entry being altered. This cannot be specified for the data or index components of clusters, alternate indexes, 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. 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.

days
can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

SHAREOPTIONS(*crossregion* [*crosssystem*] **)**
specifies how a data or index component of a cluster or alternate index can be shared. This can be specified only for the data or index components of clusters and alternate indexes.

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 alternate index 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 alternate index 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 alternate index 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 alternate index 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:

1
Reserved

2
Reserved

3
specifies that any number of users can share the component, cluster, or alternate index 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 alternate index 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.

WRITECHECK | NOWRITECHECK
specifies whether a 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.

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. These parameters can be specified only for the data or index component of a cluster or alternate index.

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.

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.

The allocation parameters permit you to respecify the amount of distributed free space, 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, respecify the length and position of the key field of a key-sequenced data set or alternate index, and respecify the data set's average and maximum recordsizes.

FREESPACE(*CI-percent* [ƀ *CA-percent*])
specifies the amount of space that is to be left free after any allocation and after any split of control intervals *(CI-percent)* and control areas *(CA-percent).*

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.

CI-percent and *CA-percent* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

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 for a cluster's or alternate index's data component.

size
is the amount of space to be provided for buffers. The decimal values you can specify are 3072, 4096, 5120, 6144, 7168, and 8192.

size can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. 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.

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 REMOVEVOLUMES—only volumes of the same device type can be processed.) See the section "VSAM Volume Cleanup" in the chapter "Data Security and Protection" for 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.

STAGE | BIND | CYLINDERFAULT
specifies how a data or index component that is stored on a mass storage volume is to be staged. These parameters can be specified only for the data or index component of a cluster or alternate index.

STAGE
indicates that the component is to be staged from mass storage to a direct-access storage staging drive when the 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.

RECORDSIZE(*average* **b** *maximum* **)**
specifies the new average and maximum lengths for the data records in the object. If the object whose entry is being altered is an alternate index path, the alternate index itself is altered; if the path points directly to its base cluster, then the base cluster is altered. If the object whose entry is being altered is an alternate index, the alternate key must be within the limit specified by *maximum*. For more information, see *entryname*.

Restrictions: RECORDSIZE can be specified only if all of the following are true:

- The object whose entry is being altered contains no data records.
- The value for maximum RECORDSIZE in the catalog must be the default value. However, if a nondefault value is altered and the new value matches the old, processing continues for any other parameters specified on the command. For default values, see the DEFINE command that defines the object.
- The new maximum record length must be at least seven bytes less than control interval size, unless the record is a spanned record.

- The new record length must be large enough to contain all prime and alternate keys previously defined.
- For an alternate index, if NONUNIQUEKEY is specified, RECORDSIZE must account for the increased record size resulting from the multiple prime key pointers in the alternate index data record.
- The object whose entry is being altered is an alternate index, a cluster, a path, or a data component.

KEYS(*length* **ъ** *offset* **)**
specifies the length and offset of the object's key. If the entry being altered defines an alternate index, *offset* applies to the alternate key in the data records in the base cluster. For more information, see *entryname*.

Restrictions: Can be specified only if all of the following are true:

- The object whose entry is being altered contains no data records.
- The values for KEYS in the catalog must be default values. However, if nondefault keys are altered and the new value matches the old, processing continues for any other parameters specified on the command. For default values, see the DEFINE command that defines the object.
- The key must fit within the record whose length is specified by the RECORDSIZE parameter.
- The key must fit in the first record segment of a spanned record.
- The new values for KEYS must not conflict with the control interval size specified when the object was defined.
- The object whose entry is being altered is an alternate index, a path, a key-sequenced cluster, or a data component of a key-sequenced cluster or alternate index.

length **ъ** *offset*
specifies the length of the key (between 1 and 255), in bytes, and its displacement from the beginning of the data record, in bytes. You can express *length* and *offset* in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Alternate Index and Path Attributes (ALTER)

The alternate index and path parameters are used to modify attributes of a previously-defined alternate index or path.

UNIQUEKEY | NONUNIQUEKEY
specifies whether the alternate-key value can be found in more than one of the base cluster's data records. UNIQUEKEY specifies that each alternate-key value is unique. If the same alternate-key value is found in more than one of the base cluster's data records, an error results.

NONUNIQUEKEY allows you to have an alternate-index record that, for a given alternate-key value, might point to more than one data record in the cluster.

UNIQUEKEY can be specified only for an empty alternate index (that is, an alternate index that is defined but not yet built).

NONUNIQUEKEY can be specified for an alternate index at any time. You should also respecify RECORDSIZE to ensure that each

alternate-index record is large enough to contain more than one data-record pointer.

UPDATE | NOUPDATE
specifies whether a base cluster's alternate index upgrade set is to be allocated when the path's name is allocated with a DD statement.

NOUPDATE specifies that the path's cluster is to be allocated, but that the cluster's alternate index upgrade set isn't to be allocated.

UPGRADE | NOUPGRADE
specifies whether an alternate index is to be upgraded (that is, kept up to date) when its base cluster is modified. UPGRADE specifies that when the cluster's records are added to, updated, or erased, the cluster's alternate index is upgraded to reflect the changed data.

If UPGRADE is specified when the cluster is open, the upgrade attribute doesn't apply to the alternate index until the cluster is closed and next opened (that is, a new set of VSAM control blocks describes the cluster and its attributes).

UPGRADE can be specified only for an empty alternate index (that is, an alternate index that is defined but not built.) However, the UPGRADE attribute is not effective for the alternate index until the alternate index is built (see the BLDINDEX command).

NOUPGRADE can be specified for an alternate index at any time.

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 specifies that all of the generation data sets are to be uncataloged. NOEMPTY specifies that only the oldest generation data set is to be uncataloged.

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 (that is, whether the data set's format-1 DSCB is removed—scratched—from the VTOC so that the data set can no longer be accessed).

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(*catname* [/ *password*] **)**
specifies the catalog location of the entry to be altered. 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.

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, D50.EXAMPLE.ESDS1. No password for the cluster is required, because the cluster was defined without passwords.

```
//ALTER1   JOB    ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     ALTER -
           D50.EXAMPLE.ESDS1 -
           MASTERPW(DEPT26M) -
           CONTROLPW(DEPT26C) -
           UPDATEPW(DEPT26U) -
           READPW(DEPT26R) -
           AUTHORIZATION(D26AUTH)
/*
```

The job control statement is:

- 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:

- D50.EXAMPLE.ESDS1, the name of the entry-sequenced cluster. It is assumed that an alias entry exists named D50 for the user catalog D27UCAT2. The data set name, D50.EXAMPLE.ESDS1, causes the ALTER request to be directed to D27UCAT2.
- MASTERPW, CONTROLPW, UPDATEPW, READPW, and AUTHORIZATION, which specify passwords and the entryname of the user's security-verification routine.

Alter the Entrynames 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 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.

BLDINDEX

BLDINDEX	{**INDATASET(***entryname* [/ *password*]) | **INFILE(***dname* [/ *password*])} {**OUTDATASET(***entryname* [/ *password*] [ƀ *entryname* [/ *password*]...]) | **OUTFILE(***dname* [/ *password*] [ƀ *dname* [/ *password*]...])} [**CATALOG(***catname* [/ *password*])] [**EXTERNALSORT** | **INTERNALSORT**] [**WORKFILES(***dname* ƀ *dname*)]

"Appendix F: Command Parameters Summary" contains a table for the BLDINDEX command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

BLDINDEX Parameters

Object Names (BLDINDEX)

When you issue the BLDINDEX command, you are required to identify the alternate index and its base cluster. You can identify each object with its entryname (that is, you specify INDATASET and OUTDATASET) or you can point to DD statements that identify each object (that is, you specify INFILE and OUTFILE).

You can build more than one alternate index for the same base cluster by specifying more than one *entryname* or *dname* with the OUTDATASET or OUTFILE parameters.

INDATASET(*entryname* [/ *password*] **)**
names a base cluster or a path that points to the base cluster. If the base cluster or path is password protected, you supply the read (or higher level) password of the named object. Otherwise, the operator or TSO terminal user is prompted to supply the correct password. When you specify INDATASET, the base cluster's volume is dynamically allocated. Therefore, the base cluster's volume must be mounted as permanently resident or reserved.

The base cluster must be defined in the same catalog as the alternate index, and must contain at least one data record.

The alternate index identified with OUTDATASET or OUTFILE must be related to the base cluster.

INFILE(*dname* [/ *password*] **)**
names the DD statement that identifies the base cluster or a path that points to the base cluster. If the base cluster or path is password-protected, you can supply the read (or higher level) password of the object named in the DD statement. Otherwise, the operator or TSO terminal user is prompted to supply the correct password.

The base cluster must be defined in the same catalog as the alternate index, and must contain data records.

The alternate index identified with OUTDATASET or OUTFILE must be related to the base cluster.

OUTDATASET(*entryname* [/ *password*]
[ъ *entryname* [/ *password*] ...])
names the alternate index or a path that points to the alternate index. If the alternate index or path is password-protected, you supply the update (or higher level) password of the named object. Otherwise, the operator or TSO terminal user is prompted to supply the correct password. When you specify OUTDATASET, the alternate index's volume is dynamically allocated. Therefore, the volume must be mounted as permanently resident or reserved.

The alternate index must be defined in the same catalog as the base cluster, and must be empty (that is, its high-used RBA equals zero) or must have been defined with the REUSE attribute.

The alternate index must be related to the base cluster identified with INDATASET or INFILE.

OUTFILE(*dname* [/ *password*] [ъ *dname* [/ *password*] ...])
names the DD statement that identifies the alternate index or a path that points to the alternate index. If the alternate index or path is password-protected, you can supply the update (or higher level) password of the object named in the DD statement. Otherwise, the operator or TSO terminal user is prompted to supply the correct password.

The alternate index must be defined in the same catalog as the base cluster, and must be empty (that is, its high-used RBA equals zero) or defined with the reusable attribute.

The alternate index must be related to the base cluster identified with INDATASET or INFILE.

Catalog (BLDINDEX)

The catalog parameter specifies the name and password, when required, of the catalog used with the BLDINDEX command.

CATALOG(*catname* [/ *password*])
names the catalog that the workfiles are to be defined in. The workfiles are defined and used by the BLDINDEX routine. When all alternate indexes are built and the workfiles are no longer needed by the BLDINDEX routine, they are deleted.

password
If the catalog is password-protected, you can supply its update (or higher level) password. Otherwise, the operator is prompted to supply the correct password.

See "Order of Catalog Use: BLDINDEX" for information about the order in which a catalog is selected when the CATALOG parameter is not specified.

Sort (BLDINDEX)

The sort parameters apply to the two sort workfiles that BLDINDEX uses to put the alternate-key/pointer pairs in ascending order.

EXTERNALSORT | INTERNALSORT
specifies whether the key-pointer pairs are to be sorted entirely within virtual storage.

When you specify EXTERNALSORT, two external-sort workfiles are defined and built as VSAM entry-sequenced clusters.

When you specify INTERNALSORT, or allow it to default, Access Method Services sorts the key-pointer pairs entirely within the user-provided virtual storage if possible. If not enough virtual storage is provided for an internal sort, two external sort workfiles are built and the key-pointer pairs are sorted externally. If the minimum amount of virtual storage is not provided (see "How an Alternate Index is Built"), the BLDINDEX processing terminates with an error message.

When you specify EXTERNALSORT, or if not enough virtual storage is provided for an internal sort, you must provide two DD statements that describe the external-sort workfiles to be defined by BLDINDEX.

You can name the DD statements IDCUT1 and IDCUT2. When you choose other names for the workfile DD statements, you must identify the DD statements with the WORKFILES parameter.

WORKFILES(*dname* **)**
names the DD statements that identify the workfiles you want BLDINDEX to define if an external sort of the key-pointer pairs is required. You can use DD statements to describe two workfiles that will be defined and opened before the BLDINDEX routine begins processing the base cluster's data records.

When you code the DD statements that describe the workfiles and identify them with the standard dnames IDCUT1 and IDCUT2, you don't need to specify the WORKFILES parameter.

For information on how to code the DD statements that describe the workfiles, see "DD Statements that Describe the SORT Workfiles."

BLDINDEX Example

This example builds an alternate index over a previously-defined base cluster, EXAMPLE.KSDS2. Data records have already been loaded into the base cluster, so that it is not empty. The alternate index, its path, and its base cluster are all defined in the same catalog, AMASTCAT.

```
//BUILDAIX JOB  ...
//STEP1    EXEC PGM=IDCAMS
//BASEDD   DD   DSNAME=EXAMPLE.KSDS2,DISP=OLD
//AIXDD    DD   DSNAME=EXAMPLE.AIX,DISP=OLD
//IDCUT1   DD   DSNAME=SORT.WORK.ONE,DISP=OLD,AMP='AMORG',
//     VOL=SER=VSER01,UNIT=2314
//IDCUT2   DD   DSNAME=SORT.WORK.TWO,DISP=OLD,AMP='AMORG',
//     VOL=SER=VSER01,UNIT=2314
//SYSIN    DD   *
     BLDINDEX INFILE(BASEDD) -
              OUTFILE(AIXDD/AIXUPPW) -
              CATALOG(AMASTCAT/MCATUPPW)
/*
```

The job control statements are:

- BASEDD DD, which describes the base cluster.
- AIXDD DD, which describes the alternate index.
- IDCUT1 and IDCUT2 DD, which describe a volume containing VSAM data space to be made available to BLDINDEX for defining and using two sort work data sets in the event an external sort is performed. This data space will not be used by BLDINDEX if enough virtual storage is available to perform an internal sort.

The BLDINDEX command builds an alternate index. The assumption is made that enough virtual storage will be available to perform an internal sort. However, note that DD statements with the default dnames of IDCUT1 and IDCUT2 have been provided for two external sort work data sets in the event that the assumption is incorrect and an external sort must be performed. The BLDINDEX command's parameters are:

- INFILE, which names the base cluster. The *dname* of the DD statement for this object must be identical to this name. Note that a password is not required since the base cluster is not protected.
- OUTFILE, which names the alternate index. The *dname* of the DD statement for this object must be identical to this name. The update password of the alternate index is also required.
- CATALOG, which identifies the master catalog. If it is necessary for BLDINDEX to use external sort work data sets, they will be defined in and deleted from the master catalog. The update password will permit these actions.

CHKLIST

The format of the CHKLIST command is:

CHKLIST	**INFILE(***dname***)** **[OUTFILE(***dname***)]** **[CHECKID(***checkid* **...)]**

"Appendix F: Command Parameters Summary" contains a summary table for the CHKLIST command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

CHKLIST Parameters

CHKLIST
specifies that identification of each tape data set that was open at the time of a checkpoint is to be listed.

INFILE(*dname* **)**
specifies the ddname of the DD statement that identifies the checkpoint data set that contains the checkpoint entries to be processed.

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.

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.

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* C0000001 and C0000002 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 E: 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.

CNVTCAT

The CNVTCAT command can be used to convert OS catalog entries into VS2 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]**

CNVTCAT Parameters

INFILE(*dname* **)**
specifies the name of a DD statement that identifies the OS catalog that is to be converted.

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 OS catalog 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.

CATALOG(*catname* **[** */* *password* **])**
specifies the name of a catalog that is to receive the converted entries. If CATALOG is not specified, see "Order of Catalog Use: CNVTCAT" for information about the order in which a catalog is selected.

password
specifies, for a password-protected catalog, the update or higher level password.

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.

password
specifies the master catalog's update or higher level password.

CVOLEQUATES((*catname* (*volser* [b *volser* ...]))[b(*catname*...)])
identifies the user catalog that converted control volume pointer entries (CVPEs) point to. CVPEs point to control volumes (CVOLs) which contain OS catalog entries. The OS catalog entries in a CVOL might have been (or might soon be) converted to VSAM catalog entries and put into a user catalog. When a CVPE is converted to an alias entry, the alias entry points to the user catalog that contains (or is to contain) the control volume's converted OS catalog entries.

When you specify CVOLEQUATES, you must also specify MASTERCATALOG.

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.

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=SYSCTLG,
//   UNIT=2314
//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 USER12 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,
//   UNIT=2314
//SYSIN     DD     *
     CNVTCAT -
        INFILE(OSCAT2) -
        CATALOG(USER12/MR) -
        CVOLEQUATES( -
            (USER11(VSER08))    ) -
        NOLIST -
        MASTERCATALOG(AMASTCAT)
/*
```

The job control langauage 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 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.

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* **])]**

DEFINE ALIAS Parameters

NAME(*aliasname* **)**
specifies the alias—the alternate entryname for a user catalog or nonVSAM data set.

aliasname can contain 1 to 44 alphanumeric characters (A through A, and 0 through 9), national characters (@, #, and $), and special characters (the hyphen (-), and the 12-0 overpunch (X'C0').)

When the name contains more than eight characters, you divide the name into segments of 1 to 8 characters each and separate the segments with (.) periods.

The name's first character (and each segment's first character) is either an alphabetic character or a national character.

RELATE(*entryname* **)**
specifies the name of the entry (that is, the user-catalog entryname or the nonVSAM data set name) for which the alias is being defined.

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 for a user catalog connector in the master catalog.

CATALOG(*catname* **[/** *password* **])**
identifies the catalog in which the alias is to be defined. If the catalog's volume is physically mounted, it is dynamically allocated. The volume must be mounted as permanently resident or reserved. See "Order of Catalog Search: DEFINE" for information about the order in which a catalog is selected when the catalog's name is not specified.

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.

DEFINE ALIAS Example

Define an Alias for a NonVSAM Data Set: Example 1

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.

DEFINE ALTERNATEINDEX

When you define an alternate index, you can specify attributes for the alternate index as a whole and for the alternate-index's components.

The format of the DEFINE ALTERNATEINDEX command is:

DEFINE	**ALTERNATEINDEX** (**NAME(***entryname* **)** **RELATE(***entryname* [**/** *password*]**)** [**FILE(***dname* **)**] **VOLUMES(***volser* [**ъ** *volser* ...]**)** [**KEYS(***length* **ъ** *offset* **|** **64** **0**)] {**TRACKS(***primary* [**ъ** *secondary*]**)** **|** **CYLINDERS(***primary* [**ъ** *secondary*]**)** **|** **RECORDS(***primary* [**ъ** *secondary*]**)**} [**RECORDSIZE(***average* **ъ** *maximum* **)**] [**FREESPACE(** *CI-percent* [**ъ** *CA-percent*] **|** **0** **0**)] [**UNIQUE** **|** **SUBALLOCATION**] [**REUSE** **|** **NOREUSE**] [**KEYRANGES((***lowkey* **ъ** *highkey* **)** [**ъ(***lowkey* **ъ** *highkey* **)...**])] [**UNIQUEKEY** **|** **NONUNIQUEKEY**] [**ORDERED** **|** **UNORDERED**] [**BUFFERSPACE(***size* **)**] [**CONTROLINTERVALSIZE(***size* **)**] [**STAGE** **|** **BIND** **|** **CYLINDERFAULT**] [**MASTERPW(***password* **)**] [**CONTROLPW(***password* **)**] [**UPDATEPW(***password* **)**] [**READPW(***password* **)**] [**CODE(***code* **)**] [**ATTEMPTS(***number* **|** **2**)] [**AUTHORIZATION(***entrypoint* [**ъ** *string*])] [**EXCEPTIONEXIT(***entrypoint* **)**] [**OWNER(***ownerid* **)**] [**TO(***date* **)** **|** **FOR(***days* **)**] [**UPGRADE** **|** **NOUPGRADE**] [**SHAREOPTIONS(***crossregion* [**ъ** *crosssystem*])] [**ERASE** **|** **NOERASE**] [**SPEED** **|** **RECOVERY**] [**WRITECHECK** **|** **NOWRITECHECK**] [**DESTAGEWAIT** **|** **NODESTAGEWAIT**] [**REPLICATE** **|** **NOREPLICATE**] [**MODEL(***entryname* [**/** *password*] [**ъ** *catname* [**/** *password*]])] [**IMBED** **|** **NOIMBED**] **)**

(Continued on next page.)

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

(Continued on next page.)

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

DEFINE ALTERNATEINDEX Parameters: Summary

The parameters of the DEFINE ALTERNATEINDEX command can be grouped as required and optional parameters.

The required parameters are:

- Entry type, which specifies that an alternate index and its catalog entry are to be created (ALTERNATEINDEX).
- Name, which names the alternate index (NAME).
- Base cluster, which identifies the base cluster related to the alternate index (RELATE).
- Allocation, which specifies the amount of space to be allocated to the alternate index and identifies the volume(s) that is to contain the alternate index (VOLUMES and one of CYLINDERS, RECORDS, or TRACKS).

The optional parameters are:

- Model, which identifies an existing alternate index entry that is to be used as a model for the alternate index being defined (MODEL).
- Allocation, which specifies
  - the name of the DD statement that describes the alternate-index's volume(s) (FILE).
  - the length and offset of the alternate-key field in the base cluster (KEYS).
  - whether the alternate-index records are to be divided by alternate-key value among volumes (KEYRANGES).
  - whether the alternate-key value can identify more than one data record in the base cluster (UNIQUEKEY or NONUNIQUEKEY).
  - the size of control intervals, records, and the user program's buffer area (CONTROLINTERVALSIZE, RECORDSIZE, and BUFFERSPACE).
  - how an alternate index or component that is stored on a mass storage device is to be staged (STAGE, BIND, or CYLINDERFAULT).
  - the amount of free space to be left in control intervals and control areas when the alternate-index records are loaded (FREESPACE).
  - whether the alternate-index is to share its data space with other VSAM objects (SUBALLOCATION or UNIQUE).
  - whether the volume's are to be allocated in the order given with the VOLUMES parameter (ORDERED or UNORDERED).
  - whether the alternate index is to be temporary and reusable (REUSE or NOREUSE).
- Data integrity, which specifies
  - the name of a user-written I/O error routine (EXCEPTIONEXIT).
  - whether alternate-index records are to be overwritten with zeros when the alternate index is deleted (ERASE or NOERASE).
  - whether the alternate index can be shared among subtasks, jobs, and systems (SHAREOPTIONS).
  - whether the alternate-index records are to be checked when the direct-access device writes them into secondary storage (WRITECHECK or NOWRITECHECK).
  - whether an alternate index or component, stored on a mass storage volume, is to be destaged synchronously or asynchronously with respect to the user's program that closes it (DESTAGEWAIT or NODESTAGEWAIT).
  - whether space is to be preformatted when the alternate index is initially loaded (RECOVERY or SPEED).
  - whether VSAM is to upgrade the alternate index each time its base cluster is modified (UPGRADE or NOUPGRADE).
- Protection, which
  - specifies passwords for the alternate index (MASTERPW, CONTROLPW, UPDATEPW, and READPW).

- identifies the alternate index to a console operator without revealing its entryname (ATTEMPTS and CODE).
- identifies a user-written security verification routine for additional authorization verification (AUTHORIZATION).
- identifies the alternate-index's owner (OWNER).
- specifies a retention period for the alternate index (FOR or TO).

- Parameters that apply to the index of an alternate index, which specify
  - whether index records are to be repeated around the track of the index's direct-access device (REPLICATE or NOREPLICATE).
  - whether the index's sequence-set records are to be written adjacent to their data control areas (IMBED or NOIMBED).
- Components, which specifies attributes for the alternate index's data and index components that might differ from the attributes specified for the alternate index as a whole or the attributes established for the alternate index being used as a model (DATA and INDEX).
- Catalog, which identifies the catalog in which the alternate index is to be defined (CATALOG).

"Appendix F: Command Parameters Summary" contains a table for the DEFINE ALTERNATEINDEX command that shows each parameter, its abbreviation, its default value (if any), and an example of its use.

DEFINE ALTERNATEINDEX Parameters

ALTERNATEINDEX
specifies that an alternate index is to be defined and have space allocated.

The ALTERNATEINDEX keyword is followed by the parameters specified for the alternate index as a whole. These parameters are enclosed in parentheses and, optionally, are followed by the DATA and INDEX parameters and their subparameters.

Name (Alternate Index)

The NAME parameter allows you to specify the name of the alternate index 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 alternate index, data component, and index component are individually named, each can be addressed. The RELATE parameter allows you to identify the alternate index's base cluster.

NAME(*entryname* **)**
specifies the alternate-index's entryname. The entryname specified for the alternate index as a whole is not propagated to the alternate-index's components. You can define a separate entryname for the alternate index, its data component, and its index component.

NAME is a required parameter of ALTERNATEINDEX. When NAME is not specified as a parameter of DATA or INDEX, VSAM generates an entryname for the alternate-index's components.

entryname
can contain 1 to 44 alphanumeric characters (A through Z, and 0 through 9), national characters (@, #, and $), and special characters (the hyphen (-), and the 12-0 overpunch (X'C0').)

When the name contains more than eight characters, you divide the name into segments of 1 to 8 characters each and separate the segments with (.) periods. The name's first character (and each segment's first character) is either an alphabetic character or a national character.

RELATE (*entryname* [/ *password*])
names the alternate index's *base cluster*. The base cluster is an entry-sequenced cluster or a key-sequenced cluster that the alternate index is to be organized over.

You can relate an alternate index only to a key-sequenced cluster or to an entry-sequenced cluster. You cannot relate an alternate index to a reusable cluster or to a relative-record cluster.

entryname
names the base cluster.

password
specifies the base cluster's master password.

You must specify the base cluster's master password when the base cluster is password-protected. If you do not specify the base cluster's master password, you can specify the catalog's master password (that is, the master password of the catalog that contains the base cluster's entry—see the CATALOG parameter). If no password is specified with either the RELATE or the CATALOG parameter, VSAM asks the operator for the base cluster's master password.

Required Allocation Parameters (Alternate Index)

The required allocation parameters are used to specify:

- The volume(s) on which an alternate index or its components are to reside.
- The amount of space to be allocated.

TRACKS(*primary* [*secondary*]) |
CYLINDERS(*primary* [*secondary*]) |
RECORDS(*primary* [*secondary*)
specifies the amount of space to be allocated to the alternate index from the volume's available space, when UNIQUE is specified, or from the available space of one of the volume's VSAM data spaces. If you specify RECORDS, the amount of space allocated is the minimum number of tracks that are required to contain the specified number of records. However, if you specify TRACKS or RECORDS and if the minimum number of tracks should exceed a cylinder, space is allocated in terms of cylinders.

When UNIQUE is specified and the data component is to be contained on more than one volume, each volume can contain a maximum of 16 extents.

When the data component is not divided into key ranges and more than one volume is specified, the *primary* amount of space is allocated only on the first volume when the component is defined. When the component increases in size so as to extend on to additional volumes, the first allocation on each overflow volume is the primary amount.

When KEYRANGES is specified and the data component is to be contained on more than one volume, the *primary* amount of space is immediately allocated on each volume required for the key ranges.

secondary amounts can be allocated on all volumes available to contain parts of the alternate index regardless of the key ranges when the alternate index is extended.

You must specify one, and only one, of the following parameters: CYLINDERS, RECORDS, or TRACKS. You can specify the amount of space as a parameter of ALTERNATEINDEX, as a parameter of DATA, or as a parameter of both DATA and INDEX. If the space is specified as a parameter of:

- ALTERNATEINDEX, the amount specified is divided between the data and index components. The division algorithm is a function of control interval size, record size, device type, and other data set attributes.
- DATA, the entire amount specified is allocated to the data component. An additional amount of space, depending on control interval size, record size, device type, and other data set attributes, is allocated to the index component.

To determine the exact amount of space allocated to each component, list the alternate index's catalog entry, using the LISTCAT command.

When you specify UNIQUE and less than one cylinder of space, and the alternate-index's data space is the first data space on a volume that belongs to a recoverable catalog, the *primary* amount defaults to (the equivalent of) one cylinder.

primary
specifies the initial amount of space that is to be allocated to the alternate index. *primary* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

secondary
specifies the amount of space that is to be allocated each time the alternate index extends, as a secondary extent. When *secondary* is specified, space for the alternate-index's data and index components can be expanded to include a maximum of 127 extents. *secondary* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

VOLUMES(*volser* [*volser* ...])
specifies the volume(s) on which an alternate-index's components are to have space.

This parameter must be specified either as a parameter of ALTERNATEINDEX, or as a subparameter of both DATA and INDEX. If the data and index components are to reside on different device types, you must specify VOLUMES as a parameter of both DATA and INDEX.

If more than one volume is listed with a single VOLUMES parameter, the volumes must be of the same device type. The volume serial number may be repeated in the list only if the KEYRANGE parameter is specified. You may wish to do this in order to have more than one keyrange on the same volume. Even in this case, repetition is only valid if all duplicate occurrences are used for the primary allocation of some keyrange.

In a system with the Mass Storage System, an alternate index or component can be defined on a mass storage volume.

volser
is a 1 to 6 alphanumeric or national character volume serial number.

When the *volser* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *volser* in single-quotation marks (for example, VOLUMES('*DORIS')).

When the *volser* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, VOLUMES('*CA''RO')).

You can code *volser* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, VOLUMES(X'E2E4C5') is the same as VOLUMES(SUE).

The VOLUMES parameter interacts with other DEFINE ALTERNATEINDEX parameters. You should take care to ensure that the volume(s) you specify for the alternate index can satisfy the alternate-index's other attributes:

- SUBALLOCATION: If UNIQUE is not specified, the volume must already contain a VSAM data space.
- KEYRANGES: If KEYRANGES and UNIQUE are specified, a VSAM data space is built and allocated on a separate volume for each key range.
- ORDERED: If ORDERED is specified, the volumes are used n the order listed. If ORDERED and KEYRANGES are specified, there is a one-for-one correspondence between the key ranges in the key-range list and the volumes in the *volser* list.
- CYLINDERS, RECORDS, TRACKS: The volume(s) contains enough available space to satisfy the component's primary space requirement.
- FILE: When UNIQUE is specified for the alternate index, the volume information supplied with the DD statement pointed to by FILE is consistent with the information specified for the alternate index and its components.
- CATALOG: If the volume already has a data space on it, subsequent data spaces can be defined for the alternate index and must be cataloged in the catalog that owns the volume.

Optional Allocation Parameters (Alternate Index)

The optional allocation parameters allow you to specify:

- The name of a DD statement that describes the alternate index's volume(s).
- The length and offset of the base cluster's alternate key field.
- Whether the alternate-index records are to be divided by alternate-key value among volumes.
- Whether an alternate-key value can identify more than one data record in the base cluster.

- The size of the alternate index's control intervals, records, and user-program's buffer area.
- The amount of free space to be left in control intervals and control areas when the alternate index is built.
- Whether the alternate-index is to share its data space with other VSAM objects.
- Whether the alternate-index's volumes are to be allocated in the specified order.
- Whether the alternate index is to be temporary and reusable.

FILE(*dname* **)**
names the DD statement that identifies the direct-access device(s) and volume(s) on which space is to be allocated to the alternate index.

If more than one volume is specified in a volume list, all volumes must be the same device type.

When the data component and index component are to reside on separate devices, you can specify a separate FILE parameter as a subparameter of DATA and INDEX to point to different DD statements.

When the alternate index is defined in a recoverable catalog, and FILE is specified as a parameter of ALTERNATEINDEX, the FILE statement must identify all volumes on which space is to be allocated. You must also use FILE to identify the volume containing the base cluster's index component (when the base cluster is key-sequenced) or data component (when the base cluster is entry-sequenced) when defining into a recoverable catalog. If the alternate index or one of its components is unique, and if the FILE parameter is not specified and the volume(s) is physically mounted, the volume(s) identified with the VOLUMES parameter is dynamically allocated. The volume(s) must be mounted as permanently resident or reserved.

When FILE refers to more than one volume, the DD statement that describes the volumes cannot be a concatenated DD statement. When the alternate index is cataloged in a recoverable catalog, the DD statement can be concatenated: part of the concatenated DD statement describes the alternate index's or component's volumes (of one device type); the other part of the DD statement describes the volume that contains the catalog recovery area (of another device type). The DD statement you specify must be in the form:

```
//dname DD UNIT=( devtype [ , unitcount ] ),
//    VOL=SER=( volser1, volser2, volser, ...), ...
```

KEYS(*length* **b** *offset* **|** **64** **0** **)**
describes the alternate-key field in the base cluster's data record. The key field of a key-sequenced cluster is called the *prime key*, to distinquish it from other key fields, called *alternate keys*. The data record's alternate key can overlap or be contained entirely within another (alternate or prime) key field.

The sum of *length* plus *offset* cannot be greater than the length of the base cluster's data record.

When the base cluster's data record is allowed to span control intervals, the record's alternate-key field is within the record's first segment (that is, in the first control interval.)

length b *offset*
specifies the length of the alternate key (between 1 and 255), in bytes, and its displacement from the beginning of the base cluster's data record, in bytes. You can express *length* and *offset* in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

KEYRANGES((*lowkey* b *highkey* **)**
[b **(** *lowkey* b *highkey* **) ...])**
specifies that portions of the alternate-index's data component are to be put on different volumes. Each portion of the alternate index is called a *key range*.

The maximum number of alternate-key ranges is 123. Key ranges must be in ascending order, and aren't allowed to overlap. However, a gap can exist between two key ranges—you are not allowed to insert records within the gap.

Keys can contain 1 to 64 characters; 1 to 128 hexadecimal characters if coded as X' *lowkey* ' b X' *highkey* '. 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.

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

The KEYRANGES parameter interacts with other DEFINE ALTERNATEINDEX parameters. You should take care to ensure that, when you specify KEYRANGES, the alternate-index's other attributes can be satisfied.

- VOLUMES: There should be as many volume-serial-numbers in the *volser* list as there are key ranges. When a volume serial number is duplicated in the *volser* list, more than one key range is allocated space on the volume. When more than one key range is contained on a volume, UNIQUE cannot be coded for the alternate-index's data component.

  When there are more volumes in the *volser* list than there are key ranges, the excess volumes are used for overflow records from any key range without consideration for key-range boundaries.

  When there are fewer volumes in the *volser* list than there are key ranges, the excess key ranges are allocated on the last volume specified—UNIQUE cannot also be specified.

- UNIQUE: When UNIQUE is specified, each key range resides in its own VSAM data space on a separate volume. Other key ranges for the alternate index cannot also reside on the volume.

- ORDERED: There is a one-for-one correspondence between the volumes in the *volser* list and the key ranges: the first volume on the volume list contains the first key range, the second volume contains the second key range, and so on. If a volume can't be allocated in the order specified by the *volser* list, your alternate-index-definition job terminates with an error message.
- KEYS: the length of the key values must not exceed the key length specified in the KEYS parameter.

UNIQUEKEY | NONUNIQUEKEY
specifies whether more than one data record (in the base cluster) can contain the same alternate-key value.

UNIQUEKEY specifies that each alternate key points to only one data record. When the alternate index is built (see the BLDINDEX command) and more than one data record contains the same alternate-key value, the BLDINDEX processing terminates with an error message.

NONUNIQUEKEY specifies that an alternate-key value might point to more than one data record in the base cluster. The alternate-key record can point to a maximum of 32,768 records with nonunique keys.

When you specify NONUNIQUEKEY, the value you specify as the maximum record size should be large enough to allow for alternate-index records that point to more than one data record.

RECORDSIZE(*average* **b** *maximum* **| 4086 32600)**
specifies the average and maximum length, in bytes, of an alternate-index record.

An alternate-index record can span control intervals, so RECORDSIZE can be larger than CONTROLINTERVALSIZE. *average* and *maximum* is any integer value that doesn't exceed the capacity of a control area, and can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. You can identify the records as fixed-length by specifying the same value for *average* and *maximum*.

You can use the following formulas to determine the size of the alternate-index record when the alternate index supports:

- a key-sequenced base cluster

  RECSZ = 5 + AIXKL + n × BCKL

- an entry-sequenced base cluster

  RECSZ = 5 + AIXKL + n × 4

where:

- RECSZ is the average record size.
- AIXKL is the alternate-key length (see the KEYS parameter).
- BCKL is the base cluster's prime-key length (you can issue the Access Method Services LISTCAT command to determine the base cluster's prime-key length.)
- n = 1 when UNIQUEKEY is specified (RECSZ is also the maximum record size.)

- n = the number of data records that contain the same alternate-key value, when NONUNIQUEKEY is specified.

When you also specify NONUNIQUEKEY, the record size you specify should be large enough to allow for as many key-pointers or RBA pointers as you anticipate. The record length values apply only to the alternate index's data component.

CONTROLINTERVALSIZE(*size* **)**
specifies the size of the alternate-index's control intervals. The size of the control interval depends on the maximum size of data records, and on the amount of bufferspace specified.

When you don't specify the control-interval size, VSAM determines the control-interval size. If you haven't specified BUFFERSPACE and the size of your records permits, VSAM selects the optimum size for the data control interval size and 512 bytes for the index control interval size.

size
for the alternate-index's data component.

The sizes you can specify—between 512 and 32,768 bytes—are a multiple of 512 or 2048:

CISZ = (n × 512) or (n × 2048)

where *n* is a positive integer from 1 to 16

size
for the alternate-index's index component.

You can specify the following values for the index component's control interval size:

CISZ = [512 | 1024 | 2048 | 4096]

When you specify a size that is not a multiple of 512 or 2048, VSAM chooses the next higher multiple.

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

STAGE
indicates that the alternate index or component is to be staged from mass storage to a direct-access storage staging drive when the alternate index or component is opened. If the alternate index 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 alternate index 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 alternate index 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 alternate index 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.

When the alternate index or component isn't stored on a mass storage

volume, the attribute is ineffective until the direct-access storage volume the alternate index 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.

FREESPACE(*CI-percent* [*CA-percent*] | **0 0)**

specifies the amount of space that is to be left empty after any allocation (that is, the initial, or primary, allocation and each extension, or secondary allocation) and any split of control intervals (*CI-percent*) and control areas (*CA-percent*) when the alternate index is built (see the BLDINDEX command). The amount of empty space in the control interval and control area is available for data records that are updated and inserted after the alternate index is initially built.

The amounts are specified as percentages. CI-percent translates into a number of bytes that is equal to, or slightly greater than, the percentage value of *CI-percent.* CA-percent translates into a number of control intervals that is equal to, or greater than, the percentage value of *CA-percent.* The percentages, which must be equal to or less than 100, can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

When you specify 100 percent of freespace, one data record is placed in the first control interval of each control area when the alternate index is built.

UNIQUE | SUBALLOCATION

specifies whether the alternate-index's components are allocated an amount of space from the volume's available space (UNIQUE) or from a VSAM data space's available space (SUBALLOCATION.)

When UNIQUE is specified, a VSAM data space is built and is assigned exclusively to each component of the alternate index. The data space is created when the alternate index is defined. The alternate-index's volume(s) must be mounted. VSAM builds a DSCB in the volume's table of contents (VTOC) to describe the data space. The name of the data space, which is the same as the component's name, is put in the DSCB. A subentry is added to the volume entry (in the VSAM catalog) to describe the VSAM data space.

When SUBALLOCATION is specified, space from one of the VSAM data spaces on the volume is assigned to the alternate-index's component.

The space-allocation attribute interacts with other DEFINE ALTERNATEINDEX parameters. You should take care to ensure that the attribute you specify for the alternate index is consistent with other attributes:

- REUSE: You cannot specify REUSE when you specify UNIQUE for an alternate index or its components.
- KEYRANGES: When UNIQUE is specified, a data space is built and allocated on a separate volume for each key range.

- VOLUMES: When UNIQUE is not specified, a VSAM data space must exist on the volume that is to contain the alternate-index's component. When UNIQUE is specified, and more than one volume is specified, VSAM must already own all the volumes except the first. If there is no VSAM space on a volume, you must execute a DEFINE SPACE CANDIDATE before your DEFINE UNIQUE.

BUFFERSPACE(*size* **)**
specifies the minimum space that your program's address space is to provide for buffers. The bufferspace size you specify helps VSAM determine the data component's and index component's control interval size.

If the specified size is less than VSAM requires for the buffers needed to run your job, VSAM terminates your job and provides an appropriate error message.

When you don't specify BUFFERSPACE, VSAM determines the buffer-space size. VSAM determines the control interval size first, then sets the buffer-space amount equal to two data control intervals and one index control interval.

When you specify BUFFERSPACE, you must specify at least enough space to contain two data control intervals and one index control interval.

size
is the amount of space to be provided for buffers. The value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

ORDERED | UNORDERED
specifies whether volumes are to be used in the order in which they are listed in the VOLUMES parameter.

When you want each key range to reside on a separate volume, you can use ORDERED so that the first key range goes on the first volume, the second key range goes on the second volume, and so on.

If ORDERED is specified and the volumes cannot be allocated in the order specified, the command is terminated.

REUSE | NOREUSE
specifies whether the alternate index can be used over and over again as a (temporary) new alternate index. When a reusable alternate index is opened, its high-used RBA can be set to zero if you open it with an ACB which specifies the RESET attribute. When you use Access Method Services command, BLDINDEX, to build a reusable alternate index, the high-used RBA is always reset to zero when the alternate index is opened for BLDINDEX processing.

You cannot specify REUSE when you also specify KEYRANGES or UNIQUE for the alternate index or its components.

Protection (Alternate Index)

The protection parameters permit you to specify:

- Passwords to be associated with the alternate index and its components.
- A prompting code and number of attempts allowed to provide the correct password in response to a prompting message at the operator's console.

- The name of a user-written verification routine for additional authorization verification.
- The name of the alternate-index's owner.
- A retention period for the alternate index.

MASTERPW(*password* **)**
specifies a master password for the alternate index. The master password allows all Access Method Services operations against the alternate index entry and its data and index entries. The master password allows the user's program to access the alternate-index's contents without restriction. For more details on how passwords can be used, see "Data Security and Protection."

If all passwords are null, ATTEMPTS, AUTHORIZATION, and CODE can be specified but have no effect until the master password is specified.

When specified as a parameter of DATA or INDEX, the master password allows the user's program to open the data component or index component and process it without restriction.

When you don't specify MASTERPW, but do specify another password, the object's highest-level password propagates upward and becomes the password for all higher levels, including the master password.

CONTROLPW(*password* **)**
specifies a control password for the alternate index and its data and index components. The control password allows read and write operations against the object's control intervals.

When specified as a parameter of DATA or INDEX, the control password allows the user's program to open the data component or index component for read and write processing of the component's control intervals (that is, the entire control interval, including the data portion of stored records and the control fields VSAM inserts into stored records and control intervals.)

If a read or update password is the only password specified for the object, it (the highest-level password) propagates upward and becomes the password for all higher unspecified levels.

UPDATEPW(*password* **)**
specifies the update password for the alternate index or its data or index component. The update password permits read and write operations against the alternate-index's data records.

When specified as a parameter of DATA or INDEX, the update password allows the user's program to open the data component or index component for read and write processing of the component's VSAM records (that is, the data portion of the stored record, not the control fields VSAM inserts into stored records).

If a read password is the only password specified for the object (that is, it is the highest-level password), it propagates upward and becomes the password for all higher levels. If you specify a higher-level password and don't specify an update password, the update or read password is null.

READPW(*password* **)**
specifies a read password for the alternate index or its data or index component. The read password permits read operations against the alternate-index's data records.

When specified as a parameter of DATA or INDEX, the read password allows the user's program to open the data component or the index component for read-only processing of the component's data records (that is, the data portion of the sorted record, not the control fields VSAM inserts into stored records). For more details on how passwords can be used, see "Data Security and Protection."

password
is a 1 to 8 EBCDIC-character password.

When the *password* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *password* in single-quotation marks (for example, READPW('*DORIS*').)

When the *password* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, READPW('*CA''ROL*')).

You can code *password* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, READPW(X'E2E4C5') is the same as READPW(SUE).

ATTEMPTS(*number* **| 2)**
specifies the maximum number of times the operator can try to enter a correct password in response to a prompting message.

This parameter can be coded, but only has effect when the alternate-index's master password is not null. A prompting message is issued only when the user hasn't already supplied the appropriate password.

number
is an integer from 0 to 7 and can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

When ATTEMPTS(0) is coded, the operator is not prompted and is not allowed to enter a password from the console.

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 alternate index. 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 alternate index's password because the default is 2.

CODE(*code* **)**
specifies a code name for the alternate index. If an attempt is made to access a password-protected alternate index without first supplying an appropriate password, a prompting message might be issued to the operator's console (see ATTEMPTS above). The prompting message includes the code name, which identifies the alternate index without revealing its entryname.

This parameter can be coded, but only has effect when the alternate-index's master password is not null. A prompting message is

issued only when the user hasn't already supplied the appropriate password.

When you don't specify a code name for the alternate index, the prompting message identifies the alternate index with its entryname.

code
can contain 1 to 8 EBCDIC characters. When the *code* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *code* in single-quotation marks (for example, CODE('*DORIS*')).

When the *code* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, CODE('*CA''ROL*')).

You can code *code* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, CODE(X'E2E4C5') is the same as CODE(SUE).

AUTHORIZATION(*entrypoint* [b *string*] **)**
specifies that a user-security-verification routine (USVR) is available for additional security verification. When a protected alternate index is opened and the user supplies a correct password other than the alternate-index's master password, the USVR receives control. For details on coding a USVR, see the *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide*.

If a USVR is loaded from an unauthorized library during Access Method Services processing, an abnormal termination will occur. See the section "Authorized Program Facility" in the chapter "Introduction."

This parameter can always be coded, but only has effect when the master password is not null.

entrypoint
specifies the name of the USVR. *entrypoint* can contain 1 to 8 alphanumeric or national characters. The name's first character is either an alphabetic character or a national character.

string
specifies information to be passed to the USVR when it receives control to verify authorization. *string* can contain 1 to 255 EBCDIC characters.

When the *string* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *string* in single-quotation marks (for example, AUTHORIZATION(*entrypoint,* '*DORIS*')).

When the *string* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, AUTHORIZATION(*entrypoint,* '*CA''ROL*')).

You can code *string* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, AUTHORIZATION(*entrypoint,* X'E2E4C5') is the same as AUTHORIZATION(*entrypoint,* SUE). The string can contain up to

255 hexadecimal characters when expressed in this form, resulting in up to 128 bytes of information.

OWNER(*ownerid* **)**
specifies the identification of the alternate-index's owner.

ownerid
can contain 1 to 8 EBCDIC characters. When the *ownerid* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *ownerid* in single-quotation marks (for example, OWNERID('*DORIS*')).

When the *ownerid* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, OWNERID('*CA''ROL*')).

You can code *ownerid* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, OWNERID(X'E2E4C5') is the same as OWNERID(SUE).

Note to TSO users: If the owner is not identified with the OWNER parameter, the TSO user's *userid* becomes the *ownerid.*

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the alternate index. The alternate index is not automatically deleted when the expiration date is reached. When you don't specify a retention period, the alternate index can be deleted at any time.

The maximum number that can be specified for *days* is 9999. If the number specified is 0 through 1830, the retention period is the number of days specified. If the number specified is between 1831 and 9999, the retention period is through the year 1999.

FOR(*days* **)**
specifies the number of days for which the alternate index is to be kept before it is allowed to be deleted. *days* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

TO(*date* **)**
specifies the date, in the form *yyddd*, where *yy* is the year and *ddd* is the Julian date (001, for January 1, through 365, for December 31), through which the alternate index is to be kept before it is allowed to be deleted.

Data Integrity (Alternate Index)

The integrity parameters permit you to specify:

- Whether VSAM is to upgrade the alternate index each time its base cluster's data records are modified.
- The sharing options to be associated with the alternate index and its components.
- Whether the alternate-index records are to be checked when VSAM writes them to the direct-access device.
- How the alternate-index records are to be loaded when the alternate index is built.

- Whether the alternate-index records are to be overwritten with binary zeros when the alternate index is deleted.
- The name of a user-written I/O error routine.

UPGRADE | NOUPGRADE
specifies whether the alternate index is to be upgraded (that is, kept up to date) when its base cluster is modified. UPGRADE specifies that when the base-cluster's records are added to, updated, or erased, the cluster's alternate index is upgraded to reflect the changed data.

When UPGRADE is specified, the alternate index's name is cataloged with the names of other alternate indexes for the base cluster. The group of alternate-index names identify the *upgrade set* — all of the base cluster's alternate indexes that are opened when the base cluster is opened for write operations.

The UPGRADE attribute is not effective for the alternate index until the alternate index is built (see the BLDINDEX command). If the alternate index is defined when the base cluster is open, the UPGRADE attribute takes effect the next time the base cluster is opened.

SHAREOPTIONS(*crossregion* [ъ *crosssystem*] **)**
specifies how an alternate-index's data or index component can be shared among users. See *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* for a full description of data set sharing.

Unless you specify sharing options, the alternate index cannot be shared.

crossregion
specifies the amount of sharing allowed among regions (within the same system). The values that can be specified are:

1
specifies that any number of users can share the component if only read operations are being performed. A user is not allowed to open the component for write operations unless the component is not in use. When the component is opened for write operations, no other user is allowed to share it.

2
specifies that any number of users can share the component for read operations. One of the users is allowed to open the component for write operations.

3
specifies that any number of users can share the component 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 (that is, the control interval's contents can be processed only by the user updating the contents) 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 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:

1
Reserved

2
Reserved

3
specifies that any number of users can share the component 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 for both read and write operations. VSAM provides some assistance to ensure data integrity.

The assistance VSAM provides each user's program to ensure data integrity in a shared environment is:

- Each PUT request results in the appropriate buffer being written immediately into the VSAM object's direct-access device space. VSAM writes out the buffer in the user's address space that contains the new or updated data record.
- Each GET request results in all of the user's input buffers being refreshed. Each buffer's contents (that is, each data and index buffer being used by the user's program) is retrieved from the VSAM object's direct-access device.

Additional information about shared data can be found in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*

WRITECHECK | NOWRITECHECK
specifies whether an alternate-index's data or index component is to be checked (by a direct-access device operation called write-check) when a record is written to the device. If WRITECHECK is specified, the record is written to the device, then read without data transfer to test for the data-check condition.

DESTAGEWAIT | NODESTAGEWAIT
specifies whether an alternate index or its 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.

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.

When the alternate index or component isn't stored on a mass storage volume, the attribute is ineffective until the direct-access storage volume the alternate index 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.

SPEED | RECOVERY
specifies whether the data component's control areas are preformatted before alternate-index records are loaded into them. When RECOVERY is specified, the data component's control areas are written with records that indicate end-of-file. When an alternate-index record is written into a control interval, it is always followed by a record that identifies the just-written record as the last record in the alternate index. If the initial load fails, you can resume loading alternate-index records after the last correctly-written record (because an end-of-file indicator identifies it as the last record—that is, no more alternate-index records follow).

When SPEED is specified, the data component's space is not preformatted. Its space might contain data records from a previous use of the space, or it might contain binary zeros—its contents are unpredictable. If the initial load fails, you must load the alternate-index records again from the beginning (because VSAM is unable to determine where your last correctly-written record is—VSAM can't find a valid end-of-file indicator when it searches your alternate-index records).

When you specify RECOVERY, your initial load takes longer because the control areas are written initially with end-of-file indicators, and again with your alternate-index records. When you specify SPEED, your initial load is quicker.

ERASE | NOERASE
specifies whether the alternate-index data component's records are to be erased when the alternate index is deleted. If ERASE is specified, the alternate-index data component's records are overwritten with binary zeros when the alternate index is deleted. This attribute applies only to the alternate-index's data component.

EXCEPTIONEXIT(*entrypoint* **)**
specifies the name of the user-written routine, called the *exception exit routine*, that receives control when an exceptional I/O error condition occurs during the transfer of data between your program's address space and the alternate-index's direct-access storage space. (An exception is any condition that causes a SYNAD exit to be taken.) The component's exception-exit routine is processed first, then the user's SYNAD exit routine receives control.

If an exception exit routine is loaded from an unauthorized library during Access Method Services processing, an abnormal termination will occur. See the section "Authorized Program Facility" in the chapter "Introduction."

Index (Alternate Index)

The optional attributes you can specify for the index of an alternate index include:

- Whether an index record is to be replicated (that is, written repeatedly 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 alternate index's data component.

The index attributes can be used in these combinations:

- The sequence set records are adjacent to the data control areas, and only the sequence set records are replicated (IMBED and NOREPLICATE).
- The sequence set records are adjacent to the data control intervals, and all levels of index records are replicated (IMBED and REPLICATE).
- All index records are together, and all index records are replicated (REPLICATE and NOIMBED).
- All index records are together, and no index records are replicated (NOREPLICATE and NOIMBED).

For some applications, specifying index options can improve the application's performance. See *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* for information on how the index's optional attributes affect performance.

IMBED | NOIMBED
specifies whether the sequence set (that is, the lowest level of the index—the level that points to alternate index records) is to be put adjacent to the alternate-index's data component.

When you specify IMBED, the sequence-set record for each control area is written as many times as it will fit on the first track adjacent to the control area.

REPLICATE | NOREPLICATE
specifies whether each index record (that is, each record in the alternate-index's index component) is to be written on a direct-access device track as many times as it will fit.

When you specify REPLICATE, rotational delay is reduced and performance is improved. However, the alternate-index's index usually takes more direct-access device space.

Model (Alternate Index)

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

You can use some attributes of the model entry and override other attributes by explicitly specifying them in the alternate index's definition. For details about how a model is used, see "How to Use One Object as a Model for Another Object." When you do not want to add or change any attributes, you specify only the entry type (ALTERNATEINDEX), the alternate index's name, its base cluster's name, and the model entry's name:

DEFINE	**ALTERNATEINDEX-** **(NAME(***entryname***)-** **RELATE(***entryname / password***)-** **MODEL(***entryname / masterpassword***))**

If the new alternate index's passwords are to be copied from the model entry, you must specify the model-entry's master password.

MODEL (*entryname* [/ *password*]
[ъ *catname* [/ *password*]])
identifies an existing alternate-index entry that is to be used as a model for the alternate index being defined.

You can use an existing data entry as a model for the alternate-index's data component, or an existing index entry as a model for the alternate-index's index component, without specifying a model for the alternate index itself.

When you use an alternate-index entry as a model for the alternate index, the model-alternate-index's data and index entries are used as models for the to-be-defined alternate-index's data and index components, unless another entry is specified with the MODEL parameter as a subparameter of DATA or INDEX.

entryname
names the entry to be used as a model. *entryname* names an alternate index entry when MODEL is a parameter of ALTERNATEINDEX.

entryname names an alternate-index's data entry when MODEL is a parameter of DATA, and it names an alternate-index's index entry when MODEL is a parameter of INDEX.

password
specifies a password. If the model entry is password protected and it is cataloged in a password-protected catalog, you supply the read (or higher level) password of either the model entry or its catalog. If you supply both passwords, the catalog's password is used.

If you are not specifying new protection attributes for the alternate index (that is, the model's passwords and protection attributes are being copied), you must supply the master password of either the model entry or its catalog.

catname
names the model-entry's catalog. You must identify the catalog that contains the model entry when:

- You want to specify the catalog's password instead of the model-entry's password.
- The model-entry's catalog is not identified with a JOBCAT or STEPCAT DD statement, and is not the master catalog.

If the catalog's volume is physically mounted, it 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 a catalog is selected when the ccatalog's name is not specified.

Components (Alternate Index)

Attributes can be specified separately for the alternate index's data and index components.

DATA(*options*)
specifies attributes for the alternate-index's data component. Attributes specified for the data component override similar, but conflicting,

attributes specified for the alternate index as a whole. Attributes specified as parameters of DATA can also be used to override similar, but conflicting, attributes of a data entry that is used as a model.

Parameters that can be specified for the alternate-index's data component are:

- ATTEMPTS, AUTHORIZATION, and CODE: Specifies protection attributes for the data component when it is opened as a separate VSAM object by the user's program.
- BUFFERSPACE: Specifies the amount of buffer space that the user's program is to provide from its address space when the program opens the data component as a separate VSAM object.
- CONTROLINTERVALSIZE: Specifies the size of the data component's control interval. The sizes you can specify are:

  CISZ = (n x 512) or (n x 2048)

  where n is a positive integer from 1 to 16.
- CYLINDERS, RECORDS, or TRACKS: Specifies an amount of space to be allocated to the data component. You can also specify an amount of space for the alternate index as a whole. When you specify space as a parameter of DATA and INDEX, however, you cannot also specify space for the alternate index as a whole.
- DESTAGEWAIT or NODESTAGEWAIT: Specifies whether the data component, stored on a mass storage volume, is to be destaged synchronously or asynchronously with respect to the user's program that closes it.
- ERASE or NOERASE: Specifies whether the data records are to be erased when the alternate index is deleted.
- EXCEPTIONEXIT: Identifies an exception exit routine for the alternate index's data component.
- FILE: Names the DD statement that describes a direct-access device and volumes that are to contain the data component. When more than one volume is to contain the data component, all volumes must be of the same device type.
- FREESPACE: Specifies the amount of space that is to be left free after any allocation and after any split of control intervals and control areas when the alternate index is built (see the BLDINDEX command.)
- KEYRANGES: Specifies that portions, or *key ranges*, of the alternate index are to be put on different volumes, and specifies the boundaries of each key range.
- KEYS: Describes the alternate-key field of the base cluster's data record.
- MASTERPW, CONTROLPW, UPDATEPW, and READPW: Specifies passwords for the data component when it is opened as a separate VSAM object by the user's program.
- MODEL: Identifies a data entry that is to be used as a model for the data component only.
- NAME: Names the data component.

- ORDERED or UNORDERED: Specifies the order in which the volumes listed with the VOLUMES parameter are allocated.
- OWNER: Identifies the data component's owner, when the owner is different from the owner of the alternate index as a whole.
- RECORDSIZE: Specifies the average and maximum sizes of the alternate-index's data records.
- REUSE or NOREUSE: Specifies whether the alternate index's data component is to be reusable.
- SHAREOPTIONS: Specifies share options for the data component when it is opened as a separate VSAM object by the user's program.
- STAGE, BIND, or CYLINDERFAULT: Specifies whether the data component, stored on a mass storage volume, is to be staged.
- UNIQUE or SUBALLOCATION: Specifies whether the data component is to occupy exclusively a VSAM data space or share an existing data space with other VSAM objects.
- UNIQUEKEY or NONUNIQUEKEY: Specifies whether more than one data record (in the base cluster) can contain the same alternate key value.
- VOLUMES: Identifies the volume(s) that is to contain the data component, and identifies other volumes that can be used as overflow volumes for the data component.
- WRITECHECK or NOWRITECHECK: Specifies whether the data records are to be checked by the direct-access device when they are written into the data component.

Restrictions are noted with each parameter's description. All of the DATA subparameters are described in more detail as parameters of the alternate index as a whole.

INDEX(*options* **)**

specifies attributes for the index component of an alternate index.

Attributes specified for the index override similar, but conflicting, attributes specified for the alternate index as a whole. Attributes specified as parameters of INDEX can also be used to override similar, but conflicting, attributes of an index entry that is used as a model.

Parameters that can be specified for the alternate-index's index component are:

- ATTEMPTS, AUTHORIZATION, and CODE: Specifies protection attributes for the index when it is opened as a separate VSAM object by the user's program.
- CONTROLINTERVALSIZE: Specifies the index component's control interval size. The sizes you can specify are: 512, 1024, 2048, or 4096 bytes.
- CYLINDERS, RECORDS, or TRACKS: Specifies an amount of space to be allocated to the index component. When you specify one of the space-quantity parameters as a parameter of INDEX, you must also specify the same type of space-quantity parameter as a parameter of DATA.

- DESTAGEWAIT or NODESTAGEWAIT: Specifies whether the index component, stored on a mass storage volume, is to be destaged synchronously or asynchronously with respect to the user's program that closes it.
- EXCEPTIONEXIT: Identifies an exception exit routine for the index component.
- FILE: The DD statement describes a direct-access device that is to contain the index component. When more than one volume is to contain the index component, all volumes must be of the same device type.
- IMBED or NOIMBED: Specifies whether the sequence set records are to be written adjacent to the data.
- MASTERPW, CONTROLPW, UPDATEPW, and READPW: Specifies passwords for the index component when it is opened as a separate VSAM object by the user's program.
- MODEL: Identifies an index entry that is to be used as a model for the index component only.
- NAME: Names the index component.
- ORDERED or UNORDERED: When ORDERED is specified, the first volume in the *volser* list (of the VOLUMES parameter) is to contain the index. Other volumes are to be available as overflow volumes. Otherwise, any volume specified in the *volser* list can be used for the index component.
- OWNER: Identifies the index component's owner, when the owner identification is different from the owner of the alternate index as a whole.
- REPLICATE or NOREPLICATE: Specifies whether the index component's records are to be repeated around the track.
- REUSE or NOREUSE: Specifies whether the alternate index's index component is to be reusable.
- SHAREOPTIONS: Specifies share options for the index component when it is opened as a separate VSAM object by the user's program.
- STAGE, BIND, or CYLINDERFAULT: Specifies how the index component, stored on a mass storage volume, is to be staged.
- UNIQUE or SUBALLOCATION: Specifies whether the index component is to occupy exclusively a VSAM data space or share an existing data space with other VSAM objects.
- VOLUMES: Identifies the volume that is to contain the index component, and identifies other volumes that can be used as overflow volumes for the index component.
- WRITECHECK or NOWRITECHECK: Specifies whether the index component's records are to be checked by the direct-access device when they are written into the index.

Restrictions are noted above with each parameter's description. All of the INDEX subparameters are described in more detail as parameters of the alternate index as a whole.

Catalog (Alternate Index)

The catalog parameter allows you to supply the name and password of the catalog in which the alternate index is to be defined. When you specify CATALOG, you identify the catalog that contains the base cluster's entry.

CATALOG(*catname* [*/ password*] **)**
identifies the catalog in which the alternate index is to be defined. The catalog also contains the base cluster's entry (see the RELATE parameter above). See "Order of Catalog Use: DEFINE" for information about the order in which a catalog is selected if the catalog's name is not specified.

catname
specifies the catalog's name.

password
specifies the catalog's update or higher level password. If the catalog is password-protected, you must supply the catalog's update or higher level password. If no password is specified with the CATALOG parameter, VSAM asks the operator for the catalog's update password.

If the catalog's volume is physically mounted, it is dynamically allocated. The volume must be mounted as permanently resident or reserved.

If the base cluster's master password is not specified with the RELATE parameter, then the catalog's master password must be specified.

DEFINE ALTERNATEINDEX Example

Define an Alternate Index: Example 1

In this example, an alternate index is defined. A previous example illustrated the definition of the alternate index's base cluster, EXAMPLE.KSDS2. A subsequent example illustrates the definition of a path, EXAMPLE.PATH, that allows you to process the base cluster's data records using the alternate key to locate them. The alternate index, path, and base cluster are defined in the same catalog, AMASTCAT.

```
//DEFAIX   JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
      DEFINE ALTERNATEINDEX -
             (NAME(EXAMPLE.AIX) -
             RELATE(EXAMPLE.KSDS2) -
             MASTERPW(AIXMRPW) -
             UPDATEPW(AIXUPPW) -
             KEYS(3 0) -
             RECORDSIZE(40 50) -
             VOLUMES(VSER04) -
             CYLINDERS(3 1) -
             NONUNIQUEKEY -
             UPGRADE) -
          CATALOG(AMASTCAT/MCATUPPW)
/*
```

The job control statement is:

- 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 to be sent.

The DEFINE ALTERNATEINDEX command creates an alternate-index entry, a data entry, and an index entry to define the alternate index EXAMPLE.AIX. The DEFINE ALTERNATEINDEX command also obtains space for the alternate index from one of the VSAM data spaces on volume VSER04, and allocates three cylinders for the alternate index's use. Since the alternate index is being defined into a recoverable catalog, the catalog recovery volume will be dynamically allocated. The command's parameters are:

- NAME, which specifies that the alternate index's name is EXAMPLE.AIX.
- RELATE, which identifies the alternate index's base cluster, EXAMPLE.KSDS2.
- MASTERPW and UPDATEPW, which specifies the alternate index's master password, AIXMRPW, and update password, AIXUPPW.
- KEYS, which specifies the length and location of the alternate key field in each of the base cluster's data records. The alternate key field is the first three bytes of each data record.
- RECORDSIZE, which specifies that the alternate index's records are variable-length, with an average size of 40 bytes and a maximum size of 50 bytes.
- VOLUMES, which specifies that the alternate index is to reside on volume VSER04. This example assumes that the volume is already cataloged in the master catalog, AMASTCAT.
- CYLINDERS, which specifies that three cylinders are allocated for the alternate index's space. When the alternate index is extended, it is to be extended in increments of one cylinder.
- NONUNIQUEKEY, which specifies that the alternate key value might be the same for two or more data records in the base cluster.
- UPGRADE, which specifies that the alternate index is to be opened by VSAM and upgraded each time the base cluster is opened for processing.
- CATALOG, which specifies that the alternate index is to be defined in the master catalog, AMASTCAT. The example also supplies the master catalog's update password, MCATUPPW.

DEFINE CLUSTER

When you define a cluster, you can specify attributes for the cluster as a whole and for the cluster's components. A general format of the DEFINE CLUSTER command is:

```
DEFINE CLUSTER (parameters) -
    [DATA (parameters)] -
    [INDEX (parameters)] -
    [CATALOG(subparameters)]
```

DEFINE CLUSTER Command Format for a Key-Sequenced Cluster

When you define a key-sequenced cluster, the format of the DEFINE CLUSTER command is:

```
DEFINE    CLUSTER
          (  NAME(entryname)
             [FILE(dname)]
             VOLUMES(volser[ b volser...])
             [INDEXED]
             [KEYS(length b offset | 64 0)]
             {TRACKS(primary[ b secondary]) |
                CYLINDERS(primary[ b secondary]) |
                RECORDS(primary[ b secondary])}
             [RECORDSIZE(average bmaximum)]
             [SPANNED | NONSPANNED]
             [FREESPACE(CI-percent[ b CA-percent] | 0 0)]
             [UNIQUE | SUBALLOCATION]
             [REUSE | NOREUSE]
             [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 | 2)]
             [AUTHORIZATION(entrypoint[ b string])]
             [EXCEPTIONEXIT(entrypoint)]
             [OWNER(ownerid)]
             [TO(date) | FOR(days)]
             [SHAREOPTIONS(crossregion[ b crosssystem])]
             [ERASE | NOERASE]
             [SPEED | RECOVERY]
             [WRITECHECK | NOWRITECHECK]
             [DESTAGEWAIT | NODESTAGEWAIT]
             [REPLICATE | NOREPLICATE]
             [MODEL(entryname[/password]
                [ b catname[/password]])]
             [IMBED | NOIMBED]      )
```

(Continued on next page.)

```
[DATA
(   [NAME(entryname)]
    [FILE(dname)]
    [VOLUMES(volser [ b volser...])]
    [TRACKS(primary [ b secondary]) |
       CYLINDERS(primary [ b secondary]) |
       RECORDS(primary [ b secondary])]
    [RECORDSIZE(average b maximum)]
    [SPANNED | NONSPANNED]
    [FREESPACE(CI-percent [ b CA-percent])]
    [REUSE | NOREUSE]
    [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])]
    [EXCEPTIONEXIT(entrypoint)]
    [OWNER(ownerid)]
    [SHAREOPTIONS(crossregion [ b crosssystem])]
    [ERASE | NOERASE]
    [SPEED | RECOVERY]
    [WRITECHECK | NOWRITECHECK]
    [DESTAGEWAIT | NODESTAGEWAIT]
    [KEYS(length b offset)]
    [MODEL(entryname [/password]
       [ b catname [/password]])]      )]
```

(Continued on next page.)

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

DEFINE CLUSTER Command Format for an Entry-Sequenced or Relative-Record Cluster

When you define an entry-sequenced cluster or a relative-record cluster, the format of the DEFINE CLUSTER command is:

DEFINE	**CLUSTER** **(** **NAME(***entryname***)** **[FILE(***dname***)]** **VOLUMES(***volser* **[ƀ** *volser* **...])** **[NONINDEXED | NUMBERED]** **{TRACKS(***primary* **[ƀ** *secondary* **]) |** **CYLINDERS(***primary* **[ƀ** *secondary* **]) |** **RECORDS(***primary* **[ƀ** *secondary* **])}** **[RECORDSIZE(***average* **ƀ***maximum***)]** **[SPANNED | NONSPANNED]** **[UNIQUE | SUBALLOCATION]** **[REUSE | NOREUSE]** **[ORDERED | UNORDERED]** **[BUFFERSPACE(***size***)]** **[CONTROLINTERVALSIZE(***size***)]** **[STAGE | BIND | CYLINDERFAULT]** **[MASTERPW(***password***)]** **[CONTROLPW(***password***)]** **[UPDATEPW(***password***)]** **[READPW(***password***)]** **[CODE(***code***)]** **[ATTEMPTS(***number* **| 2)]** **[AUTHORIZATION(***entrypoint* **[ƀ** *string* **])]** **[EXCEPTIONEXIT(***entrypoint***)]** **[OWNER(***ownerid***)]** **[TO(***date***) | FOR(***days***)]** **[SHAREOPTIONS(***crossregion* **[ƀ** *crosssystem* **])]** **[ERASE | NOERASE]** **[SPEED | RECOVERY]** **[WRITECHECK | NOWRITECHECK]** **[DESTAGEWAIT | NODESTAGEWAIT]** **[MODEL(***entryname* **[/** *password* **]** **[ƀ** *catname* **[/** *password* **])]** **)**

(Continued on next page.)

	[DATA ([NAME(*entryname*)] [FILE(*dname*)] [VOLUMES(*volser* [ƀ *volser*...])] [TRACKS(*primary* [ƀ *secondary*]) | CYLINDERS(*primary* [ƀ *secondary*]) | RECORDS(*primary* [ƀ *secondary*])] [RECORDSIZE(*average* ƀ*maximum*)] [SPANNED | NONSPANNED] [UNIQUE | SUBALLOCATION] [REUSE | NOREUSE] [ORDERED | UNORDERED] [BUFFERSPACE(*size*)] [CONTROLINTERVALSIZE(*size*)] [STAGE | BIND | CYLINDERFAULT] [MASTERPW(*password*)] [CONTROLPW(*password*)] [UPDATEPW(*password*)] [READPW(*password*)] [CODE(*code*)] [ATTEMPTS(*number*)] [AUTHORIZATION(*entrypoint* [ƀ *string*])] [EXCEPTIONEXIT(*entrypoint*)] [OWNER(*ownerid*)] [SHAREOPTIONS(*crossregion* [ƀ *crosssystem*])] [ERASE | NOERASE] [SPEED | RECOVERY] [WRITECHECK | NOWRITECHECK] [DESTAGEWAIT | NODESTAGEWAIT] [MODEL(*entryname* [/ *password*] [ƀ *catname* [/ *password*]])])] [CATALOG(*catname* [/ *password*])]

DEFINE CLUSTER Parameters: Summary

The parameters of the DEFINE CLUSTER command can be divided into required and optional parameters.

- Entry type, which specifies that a cluster and its catalog entry are to be created (CLUSTER).
- Name, which names the cluster (NAME).
- Allocation, which specifies the amount of space to be allocated to the cluster and identifies the volume(s) that is to contain the cluster (VOLUMES and one of CYLINDERS, RECORDS, or TRACKS).

The optional parameters are:

- Type, which specifies the kind of data organization for the cluster (INDEXED, NONINDEXED, or NUMBERED).
- Model, which identifies an existing cluster entry that is to be used as a model for the cluster being defined (MODEL).

- Allocation, which specifies
  - whether the cluster's data space is to be shared with other VSAM objects (SUBALLOCATION or UNIQUE).
  - the name of the DD statement that describes the cluster's volume(s) (FILE).
  - for a key-sequenced cluster, the amount of free space to be left in control intervals and control areas when the data records are loaded (FREESPACE).
  - the size of control intervals, records, and the user program's buffer area (CONTROLINTERVALSIZE, RECORDSIZE, and BUFFERSPACE).
  - how a cluster or component that is stored on a mass storage device is to be staged (STAGE, BIND, or CYLINDERFAULT).
  - whether the volume's are to be allocated in the order given with the VOLUMES parameter (ORDERED or UNORDERED).
  - whether the cluster is intended for temporary storage of data (REUSE or NOREUSE).
  - whether the cluster's records can cross control interval boundaries (SPANNED or NONSPANNED).
- Data integrity, which specifies
  - the name of a user-written I/O error routine (EXCEPTIONEXIT).
  - whether data records are to be overwritten with binary zeros when the cluster is deleted (ERASE or NOERASE).
  - whether the data records are to be checked when the direct-access device writes them into secondary storage (WRITECHECK or NOWRITECHECK).
  - whether a cluster or component, stored on a mass storage volume, is to be destaged synchronously or asynchronously with respect to the user's program that closes it (DESTAGEWAIT or NODESTAGEWAIT).
  - whether space is to be preformatted when the data is initially loaded (RECOVERY or SPEED).
- Protection, which
  - specifies passwords for the cluster (MASTERPW, CONTROLPW, UPDATEPW, and READPW).
  - identifies the cluster to a console operator without revealing the cluster's entryname (ATTEMPTS and CODE).
  - identifies a user-written security verification routine for additional authorization verification (AUTHORIZATION).
  - identifies the cluster's owner (OWNER).
  - specifies a retention period for the cluster (FOR or TO).
- Parameters that apply only to the definition of a key-sequenced cluster, which specify
  - whether index records are to be repeated around the track of the index's direct-access device (REPLICATE or NOREPLICATE).

- whether the index's sequence-set records are to be written adjacent to their data control areas (IMBED or NOIMBED).
- whether the data records are to be divided by key among volumes (KEYRANGES).
- the length and offset of the cluster's key field (KEYS).

- Components, which specifies attributes for the cluster's data and, for a key-sequenced cluster, index components that might differ from those attributes specified for the cluster as a whole or the attributes established for the cluster being used as a model (DATA and INDEX).
- Catalog, which identifies the catalog in which the cluster is to be defined (CATALOG).

"Appendix F: Command Parameters Summary" contains a table for the DEFINE CLUSTER command that shows each parameter, its abbreviation, its default value (if any), and an example of its use.

DEFINE CLUSTER Parameters

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.

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 data or index 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), nonindexed (entry-sequenced), or numbered (relative-record). 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 the FILE statement must identify all volumes on which space is to be allocated.
- 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 *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* for information on how index options affect performance.

INDEXED | NONINDEXED | NUMBERED
specifies the type of data organization that the cluster is to have. When a cluster is defined, you can specify whether the data organization is to be indexed (key-sequenced), sequential (entry-sequenced), or numbered (relative-record). The data organization you select must be consistent with other parameters you specify.

INDEXED
specifies that the cluster being defined is for key-sequenced data. If INDEXED is specified, an index component is automatically defined and cataloged. The data records can be accessed by key or by relative-byte address (RBA). See also the INDEX parameter.

NONINDEXED
specifies that the cluster being defined is for entry-sequenced data. The data records can be accessed sequentially or by relative-byte address (RBA).

NUMBERED
specifies that the cluster's data organization is for relative-record data. A relative-record cluster is similar to an entry-sequenced cluster, and has fixed-length records that are stored in *slots*. Each slot might or might not contain a data record. The data records are accessed by relative-record number (that is, slot number).

Parameters That Apply Only to Key-Sequenced Clusters

KEYS(*length* **b** *offset* **| 64 0)**
specifies information about the key field of a key-sequenced data set's data records.

length **b** *offset*
specifies the length and offset of the key. The sum of length + offset cannot exceed the length of the shortest record.

The length of the key can be from 1 through 255 bytes. *length* and *offset* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

The key field of the cluster's index is called the *prime key* to distinquish it from other key fields, called *alternate keys*. See DEFINE ALTERNATEINDEX for more details on how to specify alternate keys for a cluster.

KEYS cannot be specified when you define an entry-sequenced cluster or a relative-record cluster. When the data record is allowed to span control intervals, the record's key field must be within the part of the record that is in the first control interval.

KEYRANGES((*lowkey* ђ *highkey*)
[ђ (*lowkey* ђ *highkey*) ...])

specifies that portions of key-sequenced data are to be placed on different volumes. Each portion of the data is called a *key range.*

The maximum number of key ranges is 123. Key ranges must be in ascending order, and aren't allowed to overlap. A gap can exist between two key ranges, but you are not allowed to insert records whose keys are within the gap.

The KEYRANGES parameter interacts with other DEFINE CLUSTER parameters. You should take care to ensure that, when you specify KEYRANGES, the cluster's other attributes can be satisfied.

- VOLUMES: There should be as many volume-serial-numbers in the *volser* list as there are key ranges. When a volume serial number is duplicated in the *volser* list, more than one key range is allocated space on the volume. When more than one key range is contained on a volume, UNIQUE cannot be coded for the cluster's data component.

  When there are more volumes in the *volser* list than there are key ranges, the excess volumes are used for overflow records from any key range without consideration for key-range boundaries.

  When there are fewer volumes in the *volser* list than there are key ranges, the excess key ranges are allocated on the last volume specified—UNIQUE cannot also be specified.

- UNIQUE: When UNIQUE is specified, each key range resides on its own volume in its own VSAM data space. Other key ranges for the cluster cannot also reside on the volume.

- ORDERED: There is a one-for-one correspondence between the volumes in the *volser* list and the key ranges: the first volume on the volume list contains the first key range, the second volume contains the second key range, and so on. If a volume can't be allocated in the order specified by the *volser* list, your cluster definition job terminates with an error message.

- KEYS: The length of the key values must not exceed the keylength specified in the KEYS parameter.

Keys can contain 1 to 64 characters; 1 to 128 hexadecimal characters if coded as X' *lowkey* ' ђ X' *highkey* '. 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.

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

REPLICATE | NOREPLICATE
specifies whether each index record is to be written on a track as many times as it will fit. When you specify REPLICATE, rotational delay is reduced and performance is improved. However, the cluster's index usually requires more direct-access device space.

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 the first track adjacent to the control area.

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.
- Whether the cluster's records can cross control interval boundaries.
- 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.
- Whether the cluster is to be temporary and reusable.
- Whether the cluster's 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. If data and index components are to reside on different device types, FILE is specified as a parameter of DATA and INDEX so that separate DD statements can be referenced.

When the cluster is defined in a recoverable catalog, and FILE is specified as a parameter of CLUSTER, the FILE statement can identify all volumes on which space is to be allocated. If the cluster or one of its components is unique, and if the FILE parameter is not specified and the volume(s) is physically mounted, the volume(s) identified with the VOLUMES parameter is dynamically allocated. The volume(s) must be mounted as permanently resident or reserved.

When FILE refers to more than one volume, the DD statement that describes the volumes cannot be a concatenated DD statement. When the cluster is cataloged in a recoverable catalog, the DD statement can be concatenated: part of the concatenated DD statement describes the cluster's or component's volumes (of one device type); the other part of the DD statement describes the volume that contains the catalog recovery area (of another device type). The DD statement you specify must be in the form:

```
//dname DD UNIT=( devtype [ , unitcount ] ),
//       VOL=SER=( volser1,volser2,volser, ...), ...
```

VOLUMES(*volser* [*volser* ...] **)**

specifies the volumes to contain the cluster or component. 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.

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. If more than one volume is listed with a single VOLUMES parameter, the volumes must be of the same device type. In a system with the Mass Storage System, a cluster or component can be defined on a mass storage volume.

A volume serial number may be repeated in the list only if the INDEXED and KEYRANGE parameters are specified. You may wish to do this in order to have more than one keyrange on the same volume. Even in this case, repetition is only valid if all duplicate occurrences are used for the primary allocation of some keyrange.

volser

is a 1 to 6 alphanumeric or national character volume serial number. Special characters you can specify are the comma [,], semicolon [;], parentheses [(and)], slash [/], asterisk [*], period [.], single quotation mark ['], ampersand [&], plus sign [+], hyphen [-], and equal sign [=]. Code a single quotation mark as two single quotation marks.

If the volume serial number includes a special character, enclose *volser* in quotation marks. For example, you can specify VOLUMES('VOL*10' 'VOL"AB') to include volumes VOL*10 and VOL'AB in the cluster's volume list.

The VOLUMES parameter interacts with other DEFINE CLUSTER parameters. You should take care to ensure that the volume(s) you specify for the cluster can satisfy the cluster's other attributes:

- SUBALLOCATION: If UNIQUE is not specified, the volume must contain a VSAM data space.
- KEYRANGES: If KEYRANGES and UNIQUE are specified, a VSAM data space is built and allocated for each key range on a separate volume.
- ORDERED: If ORDERED is specified, the volumes are used in the order listed. If ORDERED and KEYRANGES are specified, there is a one-for-one correspondence between the key ranges in the key-range list and the volumes in the *volser* list.
- CYLINDERS, RECORDS, TRACKS: The volume(s) must contain enough unallocated space to satisfy the component's primary space requirement.
- FILE: The volume information supplied with the DD statement(s) pointed to by FILE must be consistent with the information specified for the cluster and its components.
- CATALOG: The data space on the volume must have been defined in the same catalog that the cluster is being defined in, and must be owned by the catalog.

TRACKS(*primary* [**ƀ** *secondary*]) |
CYLINDERS(*primary* [**ƀ** *secondary*]) |
RECORDS(*primary* [**ƀ** *secondary*])

specifies the amount of space to be allocated to the cluster from the volume's available space, when UNIQUE is specified, or from the available space of one of the volume's VSAM data spaces. If you specify RECORDS, the amount of space allocated is the minimum number of tracks that are required to contain the specified number of records. However, if you specify TRACKS or RECORDS and if the minimum number of tracks should exceed a cylinder, space is allocated in terms of cylinders. When UNIQUE is specified and the data component is to be contained on more than one volume, each volume can contain a maximum of 16 extents.

When the data component is not divided into key ranges and more than one volume is specified, the *primary* amount of space is allocated only on the first volume when the component is defined. When the component increases in size so as to extend to additional volumes, the first allocation on each overflow volume is the primary amount.

When KEYRANGES is specified and the data component is to be contained on more than one volume, the *primary* amount of space is immediately allocated on each volume required for the key ranges.

secondary amounts can be allocated on all volumes available to contain parts of the cluster regardless of the key ranges.

You must specify one, and only one, of the following parameters: CYLINDERS, RECORDS, or TRACKS.

You can specify the amount of space as a parameter of CLUSTER, as a parameter of DATA, or as a parameter of both DATA and INDEX. When a key-sequenced cluster is being defined and the space is specified as a parameter of:

- CLUSTER, the amount specified is divided between the data and index components. The division algorithm is a function of control interval size, record size, device type, and other data set attributes.
- DATA, the entire amount specified is allocated to the data component. An additional amount of space, depending on control interval size, record size, device type, and other data set attributes, is allocated to the index component.

To determine the exact amount of space allocated to each component, list the cluster's catalog entry, using the LISTCAT command.

When you specify UNIQUE and less than one cylinder of space, and the cluster's data space is the first data space on a volume that belongs to a recoverable catalog, an additional amount of (the equivalent, in tracks, of) one cylinder is required for the recovery area data space.

primary

specifies the initial amount of space that is to be allocated to the cluster.

secondary
specifies the amount of space that is to be allocated each time the cluster extends, as a secondary extent. When *secondary* is specified, space for the cluster's data and index components can be expanded to include a maximum of 123 extents.

primary and *secondary* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

If the data space that contains (all or part of) the cluster can't be extended because the cluster's secondary allocation amount is greater than the data space's secondary allocation amount, VSAM builds a new data space. The new data space's primary and secondary allocation amounts are equal to the cluster's secondary allocation amount.

RECORDSIZE(*average* ъ *maximum* | *default* **)**
specifies the average and maximum lengths, in bytes, of the records in the data component.

RECORDSIZE can be specified as a parameter of either CLUSTER or DATA.

average and *maximum* are integer values and can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. The minimum recordsize that you can specify is 1 byte. For nonspanned records, the maximum recordsize + 7 cannot exceed the data component's control interval size (that is, the maximum nonspanned recordsize, 32,761, + 7 equals the maximum data-component control interval size, 32,768.) For spanned records, the maximum recordsize cannot exceed the number of control intervals per control area × (control interval size less ten), as calculated by VSAM. When you specify a recordsize that is larger than one control interval, you must also specify spanned records (SPANNED).

You can identify the records as fixed-length by specifying the same value for *average* and *maximum*. If NUMBERED is also specified, the records must be fixed-length (that is, *average* equals *maximum*.)

When your records are fixed-length, you can use the following formula to find a control-interval size that contains a whole number (n) of records:

$$CISZ = n \times (RECSZ + 3) + 4$$

or

$$n = (CISZ - 4) \div (RECSZ + 3)$$

where:

- n is the number of fixed-length records in a control interval, and is a positive integer.
- CISZ is the control-interval size (see also the CONTROLINTERVALSIZE parameter).
- RECSZ is the average record size.

default
When SPANNED is specified, the default is RECORDSIZE(4086 32600). Otherwise, the default is RECORDSIZE(4089 4089).

Caution: When you specify RECORDS, you should ensure that:

$$REC(sec) \times RECSZ(avg) > RECSZ(max)$$

where:

- REC(sec) is the secondary space allocation quantity, in records.
- RECSZ(avg) is the average recordsize (default = 4086 or 4089 bytes.)
- RECSZ(max) is the maximum recordsize (default = 4089 or 32600 bytes.)

When the SPANNED recordsize default prevails (32600 bytes), the secondary allocation quantity should be at least 8 records.

SPANNED | NONSPANNED
specifies whether a data record is allowed to cross control-interval boundaries. When the maximum length of a data record (as specified with RECORDSIZE) is larger than a control interval, the record can be contained on more than one control interval. To ensure that you want this to happen, specify SPANNED. (This allows VSAM to select a control-interval size that is optimum for the direct-access device. Otherwise, VSAM would select a larger control-interval size that accomodates your largest record.

When a larger-than-control-interval data record is put into a cluster that allows spanned records, the first part of the record completely fills a control interval. Subsequent control intervals are filled until the record is written into the cluster. Unused space in the record's last control interval is not available to contain other data records.

FREESPACE(*CI-percent* [b *CA-percent*]| **0 0)**
specifies the amount of space that is to be left free when the cluster is loaded and after any split of control intervals (CI-percent) and control areas (CA-percent). The amounts are specified as percentages. FREESPACE can be specified only for key-sequenced clusters. *CI-percent* and *CA-percent*, which must be equal to or less than 100, can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

When you specify FREESPACE(100 100), one data record is placed in each control interval used for data and one control interval in each control area is used for data (that is, one data record is stored in each control area when the data set is loaded.) When no FREESPACE value is coded, the default specifies that no free space be reserved when the data set is loaded.

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 separately. Their names appear in the VTOC of the volume(s) within a format-1 DSCB.

If SUBALLOCATION is specified, the name of the data space, not of the component, appears in the VTOC. If SUBALLOCATION is coded, a data space must exist on the volume on which the cluster or components are to reside. When UNIQUE is specified as a parameter of CLUSTER, a data space is built and allocated to each component of the cluster.

The space-allocation attribute interacts with other DEFINE CLUSTER parameters. You should take care to ensure that the attribute you specify for the cluster is consistent with other attributes:

- REUSE: You cannot specify REUSE when you specify UNIQUE for a cluster or its components.
- KEYRANGES: When UNIQUE is specified, a data space is built and allocated for each key range. Each key range is on a separate volume.
- VOLUMES: When UNIQUE is not specified, a VSAM data space must exist on the volume that is to contain the cluster's component. When UNIQUE is specified, and more than one volume is specified, VSAM must already own all the volumes except the first. If there is no VSAM space on a volume, you must execute a DEFINE SPACE CANDIDATE before your DEFINE UNIQUE.

REUSE | NOREUSE
specifies whether the cluster can be opened again and again as a temporary, or reusable, cluster (that is, with its high-used RBA set to zero when you open it with an ACB which specifies the RESET attribute). REUSE allows you to create an entry-sequenced, key-sequenced, or relative-record workfile.

When you create a reusable cluster, you cannot build an alternate index to support it. Also, you cannot create a reusable cluster with keyranges (see the KEYRANGE parameter) or with its own data space (see the UNIQUE parameter).

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.

BUFFERSPACE(*size* **)**
specifies the minimum space to be provided for buffers. The bufferspace size you specify helps VSAM determine the data component's and index component's control interval size. 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.

size
is the amount of space to be provided for buffers. *size* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. 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. If the specified size is less than VSAM requires for the buffers needed to run your job, VSAM terminates your DEFINE and provides an appropriate error message.

CONTROLINTERVALSIZE(*size* **)**
specifies the size of the control interval for the cluster or component.

The size of the control interval depends on the maximum size of the data records and the amount of buffer space you provide. If the size you specify is not an integer multiple of 512 or 2048, VSAM selects the next higher

multiple for your cluster's control-interval size.

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 the optimum size for the data component and 512 for the index component.

size
for a cluster's data component.

The size of a data control interval must be at least 7 bytes larger than the maximum record length if the data set does not have the SPANNED attribute. If the control interval specified is less than maximum record length plus the 7-byte overhead, then VSAM will increase the data control interval size to contain the maximum record length plus the needed overhead. If the data set has the spanned attribute, the control interval size can be less than the maximum record length. The sizes you can specify—between 512 and 32,768 bytes—are an integer multiple of 512 or 2048:

CISZ = (n × 512) or (n × 2048)

where *n* is a positive integer from 1 to 16

size
for a key-sequenced cluster's index component.

You can specify the following values:

CISZ = [512 | 1024 | 2048 | 4096]

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.

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 a user-written I/O error handling routine (the exception exit 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 password for the entry being defined. The master password allows all Access Method Services operations against the cluster entry and its data and index entries. The master password allows the user's program to access the cluster's contents without restriction. For more details on how passwords can be used, see "Data Security and Protection." 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.

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

If a read or update password is the only password specified for the object, it (the highest-level password) propagates upward and becomes the password for all higher unspecified levels.

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

If a read password is the only password specified for the object (that is, it

is the highest-level password), it propagates upward and becomes the password for all higher levels. If you specify a higher-level password and don't specify an update password, the update password is null.

READPW(*password* **)**
specifies a read password for the entry being defined. The read password permits read operations against the cluster's data records.

password
is a one-to-eight EBCDIC character password.

When the *password* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *password* in single-quotation marks (for example, MASTERPW('*DORIS*')).

When the *password* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, MASTERPW('*CA''ROL*')).

You can code *password* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, MASTERPW(X'E2E4C5') is the same as MASTERPW(SUE).

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.

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.

This parameter can be coded, but only has effect when the cluster's or component's master password is not null. A prompting message is issued only when the user hasn't already supplied the appropriate password.

code
can contain 1 to 8 EBCDIC characters. When the *code* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *code* in single-quotation marks (for example, CODE('*DORIS*')).

When the *code* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, CODE('*CA''ROL*')).

You can code *code* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, CODE(X'E2E4C5') is the same as CODE(SUE).

ATTEMPTS(*number* | **2** **)**
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.

number
can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. This parameter can be coded, but only has effect when the entry's master password is not null. A prompting message is issued only when the user hasn't already supplied the appropriate password.

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 cluster'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 cluster is accessed and the user supplies a correct password other than the cluster's master password, the USVR receives control. See *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* for details on the USVR.

If a USVR is loaded from an unauthorized library during Access Method Services processing, an abnormal termination will occur. See the section "Authorized Program Facility" in the chapter "Introduction."

This parameter can be coded, but only has effect when the entry's master password is not null.

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, and can contain 1 to 255 EBCDIC characters. When the *string* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *string* in single-quotation marks (for example, AUTHORIZATION(*entrypoint,* '*DORIS*')).

When the *string* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, AUTHORIZATION(*entrypoint,* '*CA''ROL*')).

You can code *string* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, AUTHORIZATION(*entrypoint,* X'E2E4C5') is the same as AUTHORIZATION(*entrypoint,* SUE). The string can contain up to 255 hexadecimal characters when expressed in this form, resulting in up to 128 bytes of information.

EXCEPTIONEXIT(*entrypoint* **)**
specifies the name of the user-written routine, called the *exception exit routine*, that receives control when an exceptional I/O error condition occurs during the transfer of data between your program's address space and the cluster's direct-access storage space. (An exception is any I/O error condition that causes a SYNAD exit to be taken.) The component's exception exit routine is processed first, then the user's SYNAD exit routine receives control.

If an exception exit routine is loaded from an unauthorized library during Access Method Services processing, an abnormal termination will occur. See the section "Authorized Program Facility" in the chapter "Introduction."

OWNER(*ownerid* **)**
specifies the identification of the cluster's owner.

ownerid
can contain 1 to 8 EBCDIC characters.

When the *ownerid* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *ownerid* in single-quotation marks (for example, OWNER('*DORIS*').)

When the *ownerid* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, OWNER('*CA''ROL*')).

You can code *ownerid* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, OWNER(X'E2E4C5') is the same as OWNER(SUE).

Note to TSO Users: If the owner is not identified with the OWNER parameter, the TSO user's *userid* becomes the *ownerid.*

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the cluster being defined. If neither TO nor FOR is specified, the cluster can be deleted at any time.

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 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.

days
can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

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

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:

1
Reserved

2
Reserved

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.

The assistance VSAM provides each user to ensure data integrity in a shared environment is:

- Each PUT request results in the appropriate buffer(s) being written immediately to the VSAM cluster's direct-access device space (that is, the buffer in the user's address space that contains the new or updated data record, and the buffers that contain new or updated index records when the user's data is key-sequenced.)
- Each GET request results in all of the user's input buffers being refreshed. That is, each buffer's contents (each data and index buffer being used by the user's program) is retrieved from the VSAM cluster's direct-access device.

Additional information about shared data can be found in *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*

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.

SPEED | RECOVERY
specifies whether storage allocated to the data component is to be preformatted before records are inserted. SPEED | RECOVERY applies only to initial loading.

When RECOVERY is specified, the data component's control areas are written with records that indicate end-of-file. When a data record is written (during the initial load) into a control interval, it is always followed by a record that identifies the just-written record as the last record in the cluster. If the initial load fails, you can resume loading data records after the last correctly-written data record (because an end-of-file indicator identifies it as the last record—that is, no more data records follow).

When SPEED is specified, the data component's space is not preformatted. Its space might contain data records from a previous use of the space, or it might contain binary zeros—its contents are unpredictable. If the initial load fails, you must load the data records again from the beginning (because VSAM is unable to determine where your last correctly-written record is—VSAM can't find a valid end-of-file indicator when it searches your data records).

When you specify RECOVERY, your initial load takes longer because the control areas are written initially with end-of-file indicators, and again with your data records. When you specify SPEED, your initial load is quicker.

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.

DESTAGEWAIT | NODESTAGEWAIT
specifies whether a cluster or its 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.

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.

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 entry as a model for the attributes of the cluster being defined. For details about how a model is used, see "How to Use One Object as a Model for Another Object."

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.

When you use a cluster entry as a model for the cluster, the model-cluster's data and index entries are used as models for the to-be-defined cluster's data and index components, unless another entry is specified with the MODEL parameter as a subparameter of DATA or INDEX.

MODEL (*entryname* [/ *password*]
[ъ *catname* [/ *password*]])
specifies that an existing entry is to be used as a model for the entry being defined.

entryname
specifies the name of the cluster or component entry to be used as a model.

password
specifies a password. If the model entry is password protected and it is cataloged in a password-protected catalog, you must supply the read (or higher level) password of either the model entry or its catalog. If both passwords are supplied, the catalog's password is used.

If you are not specifying new protection attributes for the cluster (that is, the model's passwords and protection attributes are being copied), you must supply the master password of either the model entry or its catalog.

catname
names the model-entry's catalog. You identify the catalog that contains the model entry for either of these cases:

- You specify the catalog's password instead of the model-entry's password.
- The model-entry's catalog is not identified with a JOBCAT or STEPCAT DD statement, and is not the master catalog.

If the catalog's volume is physically mounted, it 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 a catalog is selected when the catalog's name is not specified.

Components (Cluster)

Attributes can be specified separately for the cluster's data and index components.

DATA(*options* **)**

specifies attributes of the data component of the cluster. You can specify separate attributes for the data component of any type of cluster (that is, entry-sequenced, key-sequenced, and relative-record).

Attributes specified for the data component override similar, but conflicting, attributes specified for the cluster as a whole. Attributes specified as parameters of DATA also override similar, but conflicting, attributes of a model cluster's data entry. Parameters that can be specified for the cluster's data component are:

- ATTEMPTS, AUTHORIZATION, and CODE: Specifies protection attributes for the data component when it is opened as a separate VSAM object by the user's program.
- BUFFERSPACE: Specifies the amount of buffer space that the user's program is to provide from its address space when the program opens the data component as a separate VSAM object.
- CONTROLINTERVALSIZE: Specifies the size of the data component's control interval. The sizes you can specify are:

  CISZ = (n × 512) or (n × 2048)

  where n is a positive integer from 1 to 16.
- CYLINDERS, RECORDS, or TRACKS: You can specify an amount of space to be allocated to the data component. You can also specify an amount of space for the cluster as a whole. When you specify space as a parameter of DATA and INDEX, however, you cannot also specify space for the cluster as a whole.
- DESTAGEWAIT or NODESTAGEWAIT: Specifies whether the data component, stored on a mass storage volume, is to be destaged synchronously or asynchronously with respect to the user's program that closes it.
- ERASE or NOERASE: Specifies whether the data records are to be erased when the cluster is deleted.
- EXCEPTIONEXIT: Identifies an exception exit routine for the cluster's data component.
- FILE: Names the DD statement that describes a direct-access device and volume that is to contain the data component. When more than one volume is to contain the data component, all volumes for that component must be the same device type.
- FREESPACE: Specifies the amount of space that is to be left free during loading and after any split of control intervals and control areas. This parameter is valid only for a key-sequenced cluster.
- KEYRANGES: Specifies that portions of the data component of a key-sequenced cluster are to be put on different volumes, and specifies the boundaries of each key range.
- KEYS: Describes the key field of a key-sequenced cluster's data record.
- MASTERPW, CONTROLPW, UPDATEPW, and READPW: Specifies passwords for the data component when it is opened as a separate VSAM object by the user's program.

- MODEL: Identifies a data entry that is to be used as a model for the data component only.
- NAME: Names the data component.
- ORDERED or UNORDERED: When ORDERED is specified, the volumes listed with the VOLUMES parameter are allocated in the order listed.
- OWNER: Identifies the data component's owner, when the owner is different from the owner of the cluster as a whole.
- RECORDSIZE: Specifies the average and maximum sizes of the cluster's data records.
- REUSE or NOREUSE: Specifies whether the data component is to be reusable.
- SHAREOPTIONS: Specifies share options for the data component when it is opened as a separate VSAM object by the user's program.
- SPEED or RECOVERY: Specifies whether the data control intervals are to be preformatted with the end-of-file indicator before data records are inserted.
- SPANNED or NONSPANNED: Specifies whether a data record is allowed to cross control-interval boundaries.
- STAGE, BIND, or CYLINDERFAULT: Specifies whether the data component, stored on a mass storage volume, is to be staged.
- UNIQUE or SUBALLOCATION: Specifies whether the data component is to occupy a separate VSAM data space or share an existing data space with other VSAM objects.
- VOLUMES: Identifies the volume that is to contain the data component, and identifies other volumes that can be used as overflow volumes for the data component. All volumes must be of the same device type.
- WRITECHECK or NOWRITECHECK: Specifies whether the data records are to be checked by the direct-access device when they are written into the data component.

DATA and its subparameters can be specified for any type of cluster. All of the DATA subparameters are described in more detail as parameters of the cluster as a whole. Certain of the above subparameters apply only to the data component of a key-sequenced cluster.

INDEX(*options* **)**
specifies attributes of the index component of a key-sequenced cluster. Attributes specified for the index or a key-sequenced cluster override similar, but conflicting, attributes specified for the cluster as a whole. Attributes specified as parameters of INDEX also override similar, but conflicting, attributes of a model cluster's index entry.

Parameters that can be specified for the cluster's index component are:

- ATTEMPTS, AUTHORIZATION, and CODE: Specifies protection attributes for the index when it is opened as a separate VSAM object by the user's program.
- CONTROLINTERVALSIZE: Specifies the size of the index's control interval. The sizes you can specify are: 512, 1024, 2048, or 4096 bytes.

- CYLINDERS, RECORDS, or TRACKS: You can specify an amount of space to be allocated to the index component. When you specify one of the space-quantity parameters as a parameter of INDEX, you must also specify the same type of space-quantity parameter as a parameter of DATA. You cannot also specify a quantity of space for the cluster as a whole.
- DESTAGEWAIT or NODESTAGEWAIT: Specifies whether the index component, stored on a mass storage volume, is to be destaged synchronously or asynchronously with respect to the user's program that closes it.
- EXCEPTIONEXIT: Identifies an exception exit routine for the index component.
- FILE: The DD statement describes a direct-access device that is to contain the index. When more than one volume is to contain the index component, all volumes for that component must be the same device type.
- IMBED or NOIMBED: Specifies whether the sequence set records are to be written adjacent to the data.
- MASTERPW, CONTROLPW, UPDATEPW, and READPW: Specifies passwords for the index when it is opened as a separate VSAM object by the user's program.
- MODEL: Identifies an index entry that is to be used as a model for the index component only.
- NAME: Names the index component.
- ORDERED or UNORDERED: When ORDERED is specified, the first volume in the *volser* list (of the VOLUMES parameter) is to contain the index. Other volumes are to be available as overflow volumes. Otherwise, any volume specified in the *volser* list can be used for the index.
- OWNER: Identifies the index's owner, when the owner identification is different from the owner of the cluster as a whole.
- REPLICATE or NOREPLICATE: Specifies whether the index's records are to be repeated around the track.
- REUSE or NOREUSE: Specifies whether the index component is to be reusable.
- SHAREOPTIONS: Specifies share options for the index when it is opened as a separate VSAM object by the user's program.
- STAGE, BIND, or CYLINDERFAULT: Specifies how the index component, stored on a mass storage volume, is to be staged.
- UNIQUE or SUBALLOCATION: Specifies whether the index is to occupy a separate VSAM data space or share an existing data space with other VSAM objects.
- VOLUMES: Identifies the volume that is to contain the index, and identifies other volumes that can be used as overflow volumes for the index. All volumes must be of the same device type.

- WRITECHECK or NOWRITECHECK: Specifies whether the index records are to be checked by the direct-access device when they are written into the index.

INDEX and its subparameters can be specified only for a key-sequenced cluster. Other restrictions are noted with each parameter's description. All of the INDEX subparameters are described in more detail as parameters of the cluster as a whole.

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*] **)**
identifies the catalog in which the cluster is to be defined. See "Order of Catalog Use: DEFINE" for information about the order in which catalogs are used.

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

password
specifies the catalog's 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 for the correct password.

If the catalog's volume is physically mounted, it is dynamically allocated. The volume must be mounted as permanently resident or reserved.

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 and the cluster's data and index components are explicitly named. This example assumes that a VSAM data space exists on volume VSER02. It also assumes that an alias name D40 has been defined for the catalog D27UCAT1. This naming convention causes D40.MYDATA to be cataloged in D27UCAT1.

```
//DEFCLU2  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE CLUSTER -
               (NAME(D40.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 statement is:

- 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 D40.MYDATA. The parameters specified for the cluster as a whole are:

- NAME, which specifies that the cluster's name is D40.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 specified for the cluster as a whole, DATA and INDEX subparameters specify values and attributes that apply only to the cluster's data or index component. The parameters specified for the data component of D40.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. A large area for I/O buffers can help to improve access time with certain types of processing. For example, with direct processing if the high level index can be kept in virtual storage, access time is reduced. With sequential processing, if enough I/O buffers are available, VSAM can perform a read-ahead thereby reducing system overhead and minimizing rotational delay.
- 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.

The parameters specified for the index component of D40.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 D40.MYDATA is to be defined in D27UCAT1 because this is the catalog that is identified in the catalog parameter.

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, D40.EXAMPLE.KSDS1. The second DEFINE command defines an entry-sequenced VSAM cluster, D50.EXAMPLE.ESDS1. In both examples it is assumed that alias names, D40 and D50, have been defined for D27UCAT1 and D27UCAT2, respectively.

```
//DEFCLU2  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE CLUSTER -
               (NAME(D40.EXAMPLE.KSDS1) -
               MODEL(D40.MYDATA) -
               VOLUMES(VSER02) -
               NOIMBED   ) -
          CATALOG(D27UCAT1/MRPWD27)
     DEFINE CLUSTER -
               (NAME(D50.EXAMPLE.ESDS1) -
               RECORDS(100 500) -
               RECORDSIZE(250 250) -
               VOLUMES(VSER03) -
               NONINDEXED   ) -
          CATALOG(D27UCAT2/MRPWD27)
/*
```

The job control statement is:

- 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 D40.EXAMPLE.KSDS1. Its parameters are:

- NAME, which specifies the name of the key-sequenced cluster, D40.EXAMPLE.KSDS1.
- MODEL, which identifies D40.MYDATA as the cluster to use as a model for D40.EXAMPLE.KSDS1. The attributes and specifications of D40.MYDATA that aren't otherwise specified with the DEFINE command's parameters are used to define the attributes and specifications of D40.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 D50.EXAMPLE.ESDS1. Its parameters are:

- NAME, which specifies the name of the entry-sequenced cluster, D50.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.

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
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE CLUSTER -
          (NAME(ENTRY) -
          RECORDSIZE(80 80) -
          KEYS(10 10) -
          VOLUMES(VSER04) -
          UNIQUE -
          CYLINDERS(5 10)   ) -
       CATALOG(MYCAT)
/*
```

The job control statement is:

- 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. Access Method Services will create two data spaces for the cluster: one for the data component, and one for the index. Both data spaces are to be on volume VSER04. This example assumes that volume VSER04 has enough available space to contain the new data space. This example also assumes that either the volume's entry is

in the MYCAT catalog or that volume VSER04 is not owned by a VSAM catalog at the beginning of the job.

- 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 a Relative-Record Cluster: Example 4

In this example, a relative-record cluster is defined. The cluster is suballocated (that is, it can reside in a VSAM data space with other VSAM objects.) Volume VSER04 does not have to be mounted or allocated at this time.

```
//DEFCLU4  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
      DEFINE CLUSTER -
             (NAME(EXAMPLE.RRDS1) -
             RECORDSIZE(100 100) -
             VOLUMES(VSER04) -
             TRACKS(10 5) -
             NUMBERED) -
          CATALOG(AMASTCAT/MCATUPPW)
/*
```

The job control statement is:

- 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 to be sent.

The DEFINE CLUSTER command builds a cluster entry and a data entry to define the relative-record cluster EXAMPLE.RRDS1. The DEFINE CLUSTER command also obtains space for the cluster from one of the VSAM data spaces on volume VSER04, and allocates ten tracks for the cluster's use. The COMMAND's parameters are:

- NAME, which specifies that the cluster's name is EXAMPLE.RRDS1.
- RECORDSIZE, which specifies that the records are fixed-length, 100 byte records. Average and maximum record length must be equal for a relative-record data set.
- VOLUMES, which specifies that the cluster is to reside on volume VSER04. This example assumes that the volume is already cataloged in the master catalog, AMASTCAT.
- TRACKS, which specifies that ten tracks are allocated for the cluster's space. When the cluster is extended, it is to be extended in increments of 5 tracks.
- NUMBERED, which specifies that the cluster's data organization is to be relative-record. This parameter overrides the INDEXED default.
- CATALOG, which supplies the master catalog's update password, MCATUPPW.

Define a Reusable Entry-Sequenced Cluster: Example 5

In this example, a reusable entry-sequenced cluster is defined. The cluster can be used as a temporary data set. Each time the cluster is opened, its high-used RBA can be reset to zero. The cluster is suballocated (that is, it can reside in a VSAM data space with other VSAM objects).

```
//DEFCLU5  JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
      DEFINE CLUSTER -
             (NAME(EXAMPLE.ESDS2) -
             RECORDSIZE(2500 3000) -
             SPANNED -
             VOLUMES(VSER03) -
             CYLINDERS(2 1) -
             NONINDEXED -
             REUSE -
             MASTERPW(ESD2MRPW) -
             UPDATEPW(ESD2UPPW)   ) -
          CATALOG(D27UCAT2/UCATMRPW)
/*
```

The job control statement is:

- SYSPRINT DD, which is required in all Access Method Services jobs. The SYSPRINT DD statement identifies the output device to which Access Method Services messages to the programmer are to be sent.

The DEFINE CLUSTER command builds a cluster entry and a data entry to define the entry-sequenced cluster EXAMPLE.ESDS2. The DEFINE CLUSTER command also obtains space for the cluster from one of the VSAM data spaces on volume VSER03, and assigns ten tracks for the cluster's use. VSER03 does not have to be mounted or allocated at this time. The command's parameters are:

- NAME, which specifies that the cluster's name is EXAMPLE.ESDS2.
- RECORDSIZE, which specifies that the records are variable-length, with an average size of 2500 bytes and a maximum size of 3000 bytes.
- SPANNED, which specifies that data records can cross control interval boundaries.
- VOLUMES, which specifies that the cluster is to reside on volume VSER03. This example assumes that the volume is already cataloged in the user catalog, D27UCAT2.
- CYLINDERS, which specifies that two cylinders are allocated for the cluster's space. When the cluster is extended, it is to be extended in increments of one cylinder.
- NONINDEXED, which specifies that the cluster's data organization is to be entry-sequenced. This parameter overrides the INDEXED parameter.
- REUSE, which specifies that the cluster is to be reusable. Each time the cluster is opened, its high-used RBA can be reset to zero and it is effectively an empty cluster.
- MASTERPW and UPDATEPW, which specifies the master password, ESD2MRPW, and the update password, ESD2UPPW, for the cluster.

- CATALOG, which specifies that the cluster is to be defined in a user catalog, D27UCAT2. The example also supplies the user catalog's master password, UCATMRPW.

Define a Key-Sequenced Cluster: Example 6

In this example, a key-sequenced cluster is defined. In subsequent examples, an alternate index is defined over the cluster, and a path is defined that relates the cluster to the alternate index. The cluster, its alternate index, and the path entry are all defined in the same catalog, AMASTCAT.

```
//DEFCLU6  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
      DEFINE CLUSTER -
             (NAME(EXAMPLE.KSDS2) -
          DATA -
             (MASTERPW(DAT2MRPW) -
             UPDATEPW(DAT2UPPW) -
             READPW(DAT2RDPW) -
             RECORDS(500 100) -
             EXCEPTIONEXIT(DATEXIT) -
             ERASE -
             FREESPACE(20 10) -
             KEYS(6 4) -
             RECORDSIZE(80 100) -
             VOLUMES(VSER04) ) -
          INDEX -
             (MASTERPW(IND2MRPW) -
             UPDATEPW(IND2UPPW) -
             READPW(IND2RDPW) -
             RECORDS(300 300) -
             IMBED -
             VOLUMES(VSER04) ) -
          CATALOG(AMASTCAT/MCATUPPW)
/*
```

The job control statement is:

- 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 to be sent.

The DEFINE CLUSTER command builds a cluster entry, a data entry, and an index entry to define the key-sequenced cluster EXAMPLE.KSDS2. The DEFINE CLUSTER command also obtains space for the cluster from one of the VSAM data spaces on volume VSER04, and allocates space separately for the cluster's data and index components. Since the cluster is being defined into a recoverable catalog, the catalog recovery area will be dynamically allocated on VSER04.

The command's parameter that applies to the cluster is:

- NAME, which specifies that the cluster's name is EXAMPLE.KSDS2.

The command's parameters that apply only to the cluster's data component are enclosed in the parentheses following the DATA keyword:

- MASTERPW, UPDATEPW, and READPW, which specifies the data component's master password, DAT2MRPW, update password, DAT2UPPW, and read password, DAT2RDPW.

- RECORDS, which specifies that an amount of tracks equal to at least 500 records is to be allocated for the data component's space. When the data component is extended, it is to be extended in increments of tracks equal to 100 records.
- EXCEPTIONEXIT, which specifies the name of the exception exit routine, DATEXIT that is to be processed if an I/O error occurs while a data record is being processed.
- ERASE, which specifies that the cluster's data is to be erased (overwritten with binary zeros) when the cluster is deleted.
- FREESPACE, which specifies the amounts of free space to be left in the data component's control intervals (20 percent) and the control areas (10 percent of the control intervals in the control area) when data records are loaded into the cluster.
- KEYS, which specifies the location and length of the key field in each data record. The key field is the sixth (byte 5) through ninth bytes of each record.
- RECORDSIZE, which specifies that the cluster's records are variable-length, with an average size of 80 bytes and a maximum size of 100 bytes.
- VOLUMES, which specifies that the cluster is to reside on volume VSER04. This example assumes that the volume is already cataloged in the master catalog, AMASTCAT.

The command's parameters that apply only to the cluster's index component are enclosed in the parentheses following the INDEX keyword:

- MASTERPW, UPDATEPW, and READPW, which specifies the index component's master password, IND2MRPW, update password, IND2UPPW, and read password, IND2RDPW.
- RECORDS, which specifies that an amount of tracks equal to at least 300 records is to be allocated for the index component's space. When the index component is extended, it is to be extended in increments of tracks equal to 300 records.
- IMBED, which specifies that the index's sequence set records are to be placed in the data component's control areas (the sequence set records will be replicated automatically).
- VOLUMES, which specifies that the index component is to reside on volume VSER04.

The CATALOG parameter supplies the catalog's update password.

Define an Entry-Sequenced Cluster (The Cluster has a Generic Name): Example 7

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     ...
//STEP 1   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 D27UCAT1, 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.

DEFINE GENERATIONDATAGROUP

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* **])]**

DEFINE GENERATIONDATAGROUP Parameters

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

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 is 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, from 1 to 255, of generation data sets that can be associated with the generation data group to be defined. *number* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

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. The disposition of the data set's DSCB in the volume's VTOC is determined with the SCRATCH | NOSCRATCH parameter.

EMPTY
specifies that all the generation data sets are to be uncataloged when the limit is reached (that is, each data set's nonVSAM entry is automatically deleted from the catalog).

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

SCRATCH | NOSCRATCH
specifies whether a generation data set's DSCB is to be deleted from the volume's VTOC when the data set is uncataloged (that is, when its entry is deleted from the catalog automatically, as a result of EMPTY | NOEMPTY, or specifically, as a result of a user-issued DELETE request). The user can override the SCRATCH | NOSCRATCH attribute when he issues the DELETE command.

SCRATCH
specifies that the generation data set's DSCB is to be deleted from the volume's VTOC when it is uncataloged. Direct-access device space management (DADSM) removes the data set's DSCB from the VTOC, erases the data set's space on the volume, and makes the space available to other system users. The generation data set ceases to exist.

NOSCRATCH
specifies that the generation data set's DSCB is not to be removed from the volume's VTOC when it is uncataloged. The data set's DSCB in the volume's VTOC is intact and is used to locate the data set. The data set can no longer be accessed through the catalog. However, your program can process the data set as it processes any other nonVSAM data sets—that is, by using a JCL DD statement to describe the data set and allocate it.

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 generation 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. If the catalog's volume is physically mounted, it is dynamically allocated. The volume must be mounted as permanently resident or reserved. See "Order of Catalog Search: DEFINE" for information about the order in which a catalog is selected when the catalog's name is not specified.

catname
specifies the name of the catalog.

password
specifies the catalog's 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 1

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.

  The model DSCB must exist on the generation data group's catalog volume.

- 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.
- 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.

DEFINE NONVSAM

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

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

"Appendix F: Command Parameters Summary" contains a table for the DEFINE NONVSAM command that shows each parameter, its abbreviation, its default value (if any), and an example of its use.

DEFINE NONVSAM Parameters

NONVSAM
specifies that a nonVSAM data set is to be defined.

NAME(*entryname* **)**
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.

Relative generation numbers (that is, GDGname(+1)) cannot be used with the entryname when you use the DEFINE NONVSAM command to catalog a generation data set and attach it to a generation data group. You identify a generation data set with its generation-data-group name (GDGname) followed by the data set's generation and version numbers: NAME(GDGname.GxxxxVyy).

The name may consist of 1 through 44 alphanumeric characters, national characters (@, #, and $), and two special characters (the hyphen and the 12-0 overpunch, X'C0'). 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* [b *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.

You can specify a device type for any device that is supported by your VS2 system. You can specify a generic device type (that is, 3330, 2314, etc.) or a specific unit address (that is, 121, 247, etc.). You can specify an esoteric device type (for example, SYSDA or TAPE) only if it has been established in the system device name table during system generation.

VOLUMES(*volser* [b *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.

For example, if your nonVSAM data set is contained in the first three files of magnetic tape volume TAPE10, you specify:

```
DEVICETYPES(2400 2400 2400) -
VOLUMES(TAPE10 TAPE10 TAPE10) -
FILESEQUENCENUMBERS(1 2 3) -
```

volser
can contain 1 to 6 EBCDIC characters.

When the *volser* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *volser* in single-quotation marks (for example, VOLUMES('*DORIS')).

When the *volser* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, VOLUMES('*CA"RO')).

You can code *volser* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, VOLUMES(X'E2E4C5') is the same as VOLUMES(SUE).

FILESEQUENCENUMBERS(*number* [b *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.

For example, if your nonVSAM data set is contained in the first three files of magnetic tape volume TAPE10, you specify:

```
DEVICETYPES(2400 2400 2400) -
VOLUMES(TAPE10 TAPE10 TAPE10) -
FILESEQUENCENUMBERS(1 2 3) -
```

number
is a file sequence number. *number* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

OWNER(*ownerid* **)**
specifies the identification of the nonVSAM data set.

ownerid
can contain 1 to 8 EBCDIC characters. When the *ownerid* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *ownerid* in single-quotation marks (for example, OWNERID('*DORIS*').

When the *ownerid* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, OWNERID('*CA''ROL*')).

You can code *ownerid* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, OWNERID(X'E2E4C5') is the same as OWNERID(SUE).

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. The nonVSAM data set is not automatically deleted when the expiration date is reached. When you don't specify a retention period, the nonVSAM data set can be deleted at any time.

The maximum number that can be specified for *days* is 9999. If the number specified is 0 through 1830, the retention period is the number of days specified. If the number specified is between 1831 and 9999, the retention period is through the year 1999.

FOR(*days* **)**
specifies the number of days for which the nonVSAM data set is to be kept before it is allowed to be deleted. *days* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

TO(*date* **)**
specifies the date, in the form *yyddd*, where *yy* is the year and *ddd* is the Julian date (001, for January 1, through 365, for December 31), through which the nonVSAM data set is to be kept before it is allowed to be deleted.

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

See "Order of Catalog Search: DEFINE" for information about the order in which a catalog is selected when the catalog's name is not specified.

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

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

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
//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 statement is:

- 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 Device.
- 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.

DEFINE PAGESPACE

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

DEFINE	**PAGESPACE** (**NAME(***entryname***)** **VOLUME(***volser***)** **[FILE(***dname***)]** **{TRACKS(***primary***) |** **CYLINDERS(***primary***) |** **RECORDS(***primary***)}** **[UNIQUE | SUBALLOCATION]** **[SWAP | NOSWAP]** **[MASTERPW(***password***)]** **[CONTROLPW(***password***)]** **[UPDATEPW(***password***)]** **[READPW(***password***)]** **[CODE(***code***)]** **[ATTEMPTS(***number* | 2**)]** **[AUTHORIZATION(***entrypoint* [b *string*])]** **[OWNER(***ownerid***)]** **[MODEL(***entryname* [/ *password*] [b *catname* [/ *password*]])]** **[TO(***date***) | FOR(***days***)]** **)** **[CATALOG(***catname* [/ *password*])]

DEFINE PAGESPACE Parameters: Summary

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.
- Parameters that apply only to the page space data sets, which describe the SWAP and NOSWAP 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.

"Appendix F: Command Parameters Summary" contains a table for the DEFINE PAGESPACE parameters that shows each parameter, its abbreviation, its default value (if any), and an example of its use.

DEFINE PAGESPACE Parameters

PAGESPACE
specifies that a pagespace is to be defined.

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.

In a system with the Mass Storage System, a pagespace cannot reside on a mass storage volume. Also, a page space with the SWAP attribute cannot be defined on a 2314 direct access device.

volser
the volume serial number, can contain one to six alphanumeric characters (A through Z, and 0 through 9), national characters (@, #, and $), and special characters. When the volume serial number contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the volume serial number in single-quotation marks (for example, VOLUME ('*DORIS*')).

When the volume serial number contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, VOLUME ('*CA"RO ')).

You can code VOLUME in hexadecimal form. Enclose it in single-quotation marks and precede it with X (for example, VOLUME (X'E2E4C5') is the same as VOLUME (SUE)).

The VOLUME parameter interacts with other DEFINE PAGESPACE parameters. You should take care to ensure that the volume you specify for the page space can satisfy the page space's other attributes:

- SUBALLOCATION: If suballocation is specified, the volume must contain a VSAM data space.
- CYLINDERS, RECORDS, TRACKS: The volume must contain enough unallocated space to satisfy the page space's primary space requirement.
- FILE: The volume information supplied with the DD statement pointed to by FILE must be consistent with the information specified for the page space.
- CATALOG: If the page space is suballocated, the data space on the volume must have been defined in the same catalog that the page space is being defined in, and must be owned by the catalog.

FILE(*dname* **)**
specifies the name of the DD statement that identifies the device and volume to be allocated to the page space. If the page space is unique, and if the FILE parameter 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* **)** |
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.

You must specify one, and only one, of the following parameters: CYLINDERS, RECORDS, or TRACKS.

To determine the exact amount of space allocated, list the page space's catalog entry, using the LISTCAT command.

When you specify UNIQUE and less than one cylinder of space, and the page space's data space is the first data space on the volume that belongs to a recoverable catalog, an additional amount of (the equivalent, in tracks, of) one cylinder is required for the recovery area data space.

primary
specifies the amount of space that is to be allocated to the page space. Once the primary extent is full, the page space is full. The page space cannot extend onto secondary extents.

primary can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

The Auxiliary Storage Manager (ASM) uses a halfword field to contain a relative paging slot (record) number within a given page space (data set). As such, the theoretical maximum number of paging slots for each page space is 65,535. This limitation currently affects only page space allocations on a 3350 device. The maximum usable page space on a 3350 is 504 cylinders, which is 15,120 tracks, or 65,520 slots.

UNIQUE | SUBALLOCATION

specifies whether the page space is allocated an amount of space from the volume's available space (UNIQUE) or from a previously defined VSAM data space (SUBALLOCATION). When UNIQUE is specified, the page space is allocated a VSAM data space of its own. VSAM generates a name for the data space and builds a format-1 DSCB in the volume's VTOC to describe it. If SUBALLOCATION is specified, the name of the data space, not of the page space, appears in the VTOC. A data space must have been defined on the volume on which the page space is defined.

Auxiliary Storage Management recommends that UNIQUE be used for page space data sets. This will cause VSAM to allocate a single non-shared data space for the page data set. Space from another data space will not be used, eliminating one of the possibilities for multiple extent data sets which will cause Auxiliary Storage Management performance degradation.

Parameters that Apply only to Page Spaces

Auxiliary Storage Management separates private area address space pages into two distinct groups: Local System Queue Area (LSQA) pages and pageable private area pages.

SWAP | NOSWAP

specifies for which group of pages the page space will be used. SWAP specifies that the page space is a high speed data set used during a swap operation to store and retrieve the set of LSQA pages owned by an address space. NOSWAP indicates that the page space is a conventional page space used to record pageable private area pages.

A page space with the SWAP attribute cannot be defined on a 2314 direct access device.

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 password for the page space. The master password allows all Access Method Services operations against the page space entry. 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.

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

If a read or update password is the only password specified for the page space, it (the highest-level password) propagates upward and becomes the password for all higher unspecified levels.

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

If a read password is the only password specified for the page space (that is, it is the highest-level password), it propagates upward and becomes the password for all higher levels. If you specify a higher-level password and don't specify an update password, the update password is null.

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

password
is a one-to-eight EBCDIC character password.

When the *password* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *password* in single-quotation marks (for example, MASTERPW('*DORIS*')).

When the *password* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, MASTERPW('*CA''ROL*')).

You can code *password* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, MASTERPW(X'E2E4C5') is the same as MASTERPW(SUE).

CODE(*code***)**
specifies a code name for the page space. 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 or TSO terminal user to be prompted for the password without disclosing the name of the entry. If code is not specified and an attempt is made to access a page space entry that is password protected without supplying a password, the operator or TSO terminal user is prompted with the name of the entry.

This parameter can be coded, but only has effect when the cluster's or component's master password is not null. A prompting message is issued only when the user hasn't already supplied the appropriate password.

code
can contain 1 to 8 EBCDIC characters. When the *code* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *code* in single-quotation marks (for example, CODE('*DORIS*')).

When the *code* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, CODE('*CA''ROL*')).

You can code *code* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, CODE(X'E2E4C5') is the same as CODE(SUE).

ATTEMPTS(*number* | 2 **)**
specifies the maximum number of times the operator or TSO terminal user 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.

number
can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. This parameter can be coded, but only has effect when the entry's master password is not null. A prompting message is issued only when the user hasn't already supplied the appropriate password.

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 page space. 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 page space'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 page space is accessed and the user supplies a correct password other than the page space's master password, the USVR receives control.

See *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* for details on the USVR.

If a USVR is loaded from an unauthorized library during Access Method Services processing, an abnormal termination will occur. See the section "Authorized Program Facility" in the chapter "Introduction."

This parameter can be coded, but only has effect when the entry's master password is not null.

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, and can contain 1 to 255 EBCDIC characters. When the *string* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *string* in single-quotation marks (for example, AUTHORIZATION(*entrypoint,* '*DORIS*')).

When the *string* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, AUTHORIZATION(*entrypoint,* '*CA''ROL*')).

You can code *string* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, AUTHORIZATION(*entrypoint,* X'E2E4C5') is the same as AUTHORIZATION(*entrypoint,* SUE). The string can contain up to

255 hexadecimal characters when expressed in this form, resulting in up to 128 bytes of information.

OWNER(*ownerid* **)**
specifies the identification of the owner of the page space.

ownerid
can contain 1 to 8 EBCDIC characters.

When the *ownerid* contains a special character (that is, a comma [,], semicolon [;], blank [␣], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*], enclose the *ownerid* in single-quotation marks (for example, OWNER('*DORIS*')).

When the *ownerid* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, OWNER('*CA''ROL*')).

You can code *ownerid* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, OWNER(X'E2E4C5') is the same as OWNER(SUE).

Note to TSO users: If the owner is not identified with the OWNER parameter, the TSO user's *userid* becomes the *ownerid.*

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the page space.

If neither TO nor FOR is specified, the page space can be deleted at any time.

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(*day* **)**
specifies the number of days for which the page space is to be kept. The maximum number that can be specified is 9999. If the number specified is 0 through 1830, the page space is retained for the number of days specified; if the number is between 1831 and 9999, the page space is retained through the year 1999.

days
can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

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.

For more details on how a model is used, see "How to Use One Object as a Model For Another Object."

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.

MODEL(*entryname* [/ *password*]
[ъ *catname* [/ *password*]])
specifies that an existing page space entry is to be used as a model for the entry being defined.

entryname
specifies the name of the page space entry to be used as a model.

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 you specify both passwords, the catalog's password is used. 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. You identify the catalog that contains the model entry for either of these cases:

- You specify the catalog's password instead of the model-entry's password.
- The model-entry's catalog is not identified with a JOBCAT or STEPCAT DD statement, and is not the master catalog.

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(*cataname* [/ *password*])
specifies the name and password of the catalog in which the page space is to be defined.

When the CATALOG parameter identifies a user catalog, you must also supply a STEPCAT or JOBCAT DD statement to describe and allocate the user catalog. See "Order of Catalog Use: DEFINE" for information about the order in which catalogs are used.

catname
specifies the name of the 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, VSAM asks the operator or TSO terminal user for the correct password.

DEFINE PAGESPACE Examples

Define a Conventional (NOSWAP) 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 PAGESPACE -
          (NAME(SYS1.PAGE2) -
          CYLINDERS(10) -
          VOLUMES(VSER05) ) -
          CONTROLPW(PASSWD1) -
          UPDATEPW(PASSWD2) -
          READPW(PASSWD3))
/*
```

The job control statements are:

- VOLUME 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.

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 PAGESPACE command defines a page space. This page space will have the UNIQUE attribute and hence a new data space will be created for its exclusive use. 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.
- CONTROLPW, which specifies the control password for the page space is PASSWD1.
- UPDATEPW, which specifies the update password for the page space is PASSWD2.
- READPW, which specifies the read password for the page space is PASSWD3.

Because neither UNIQUE nor SUBALLOCATION has been specified, the page space defaults to UNIQUE. Because neither SWAP nor NOSWAP has been specified, the page space defaults to NOSWAP.

Define a SWAP 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) -
          SWAP -
          CONTROLPW(PASSWD1) -
          UPDATEPW(PASSWD2) -
          READPW(PASSWD3))
/*
```

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.
- SWAP, which specifies the page space will be used to store Local System Queue Area (LSQA) pages.
- CONTROLPW, which specifies the control password for the page space is PASSWD1.
- UPDATEPW, which specifies the update password for the page space is PASSWD2.
- READPW, which specifies the read password for the page space is PASSWD3.

Because neither UNIQUE nor SUBALLOCATION has been specified, the page space defaults to UNIQUE.

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.

DEFINE PATH

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

DEFINE	**PATH** (**NAME(***entryname* **)** **PATHENTRY(***entryname* [/ *password*]) [**FILE(***dname* **)**] [**ATTEMPTS(***number* | **2**)] [**AUTHORIZATION(***entrypoint* [ƀ *string*])] [**CODE(***code* **)**] [**TO(***date* **)** | **FOR(***days* **)**] [**MASTERPW(***password* **)**] [**CONTROLPW(***password* **)**] [**UPDATEPW(***password* **)**] [**READPW(***password* **)**] [**OWNER(***ownerid* **)**] [**MODEL(***entryname* [/ *password*] [ƀ *catname* [/ *password*]])] [**UPDATE** | **NOUPDATE**] **)** [**CATALOG(***catname* [/ *password*])]

The parameters of the DEFINE PATH command can be grouped as required and optional parameters.

The required parameters are:

- Entry type, which specifies that a path catalog entry is to be created (PATH.)
- Name, which names the path and the entry that it points to (NAME and PATHENTRY.)

The optional parameters are:

- Allocation, which identifies the catalog recovery volume that contains the alternate index or cluster that the path points to (FILE). This parameter is required when the path is defined in a recoverable catalog.
- Protection and integrity, which
  - identify the path to a console operator without revealing the path's entryname (CODE and ATTEMPTS).
  - identify a user-written security verification routine for additional authorization verification (AUTHORIZATION).
  - specify a retention period for the path (FOR or TO.)
  - associate passwords with the path (MASTERPW, CONTOLPW, UPDATEPW, and READPW).
  - identify the owner of the path (OWNER).
- Model, which identifies an existing path entry that is to be used as a model for the path being defined (MODEL).

- Update, which specifies whether the cluster's upgrade set is to be opened when the path (and its cluster) is opened (UPDATE or NOUPDATE).
- Catalog, which identifies the catalog that contains the alternate index or cluster entry that the path entry is to point to (CATALOG).

"Appendix F: Command Parameters Summary" contains a table for the DEFINE PATH command that shows each parameter, its abbreviation, its default value (if any), and an example of its use.

DEFINE PATH Parameters

PATH
specifies that a path is to be defined.

The PATH keyword is followed by parameters which name the path and its related object, and which specify the path's attributes.

Name (Path)

The NAME parameter allows you to specify the name of the path and the name of its related alternate index or cluster.

NAME(*entryname* **)**
specifies the path's name.

entryname
can contain 1 to 44 alphanumeric characters (A through Z, and 0 through 9), national characters (@, #, and $), and special characters (the hyphen (-), and the 12-0 overpunch (X'C0')).

When the name contains more than eight characters, you divide the name into segments of 1 to 8 characters each and separate the segments with (.) periods. The name's first character (and each segment's first character) is either an alphabetic character or a national character.

PATHENTRY(*entryname* [/ *password*])
when the path consists of an alternate index and its base clusters, *entryname* identifies the alternate index entry. When the path is opened to process data records, both the alternate index and the base cluster are opened.

When the path consists of a cluster without an alternate index, *entryname* identifies the cluster. You can define the path as though it were an alias for the cluster. This allows you to specify no-update access to the cluster, so that the upgrade set will not be required or updated when the cluster is opened. You can also establish protection attributes for the alternate name, separate from the protection attributes of the cluster.

If the cluster or alternate index entry is password protected, you must supply the entry's master password. When you identify the catalog with the CATALOG parameter, you can supply the catalog's master password instead of the entry's password.

Allocation (Path)

The allocation parameter is used to identify the base cluster's recovery volume.

FILE(*dname* **)**
specifies the name of a DD statement that identifies the recovery volume of the alternate index or cluster named with the PATHENTRY parameter. FILE is only required when the path is defined in a recoverable catalog.

The *recovery volume* is the first volume of the base cluster's index component when PATHENTRY names a key-sequenced cluster or an alternate index over a key-sequenced cluster.

The *recovery volume* is the first volume of the base cluster's data component when PATHENTRY names an entry-sequenced cluster or an alternate index over an entry-sequenced cluster.

When FILE is not specified, the cluster's recovery volume is dynamically allocated. The volume must be mounted as permanently resident or reserved.

Protection and Integrity (Path)

The protection and integrity parameters allow you to:

- Specify passwords to be associated with the path.
- Specify a prompting word and number of attempts allowed to provide the correct password when the operator responds to a prompting message.
- Specify a user-supplied security verification routine.
- Identify the owner of the path.
- Specify a retention period for the path.

ATTEMPTS(*number* | **2** **)**
specifies the maximum number of times the operator can try to enter a correct password in response to a prompting message.

This parameter can be coded, but only has effect when the path's master password is not null. A prompting message is issued only when the user hasn't already supplied the appropriate password.

number
is an integer from 0 to 7 and can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

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 path. 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 path'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 path is accessed and the user supplies a correct password other than the cluster's master password, the USVR receives control. See *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide* for details on the USVR.

If a USVR is loaded from an unauthorized library during Access Method Services processing, an abnormal termination will occur. See the section "Authorized Program Facility" in the chapter "Introduction."

This parameter can always be coded, but only has effect when the path's master password is not null.

entrypoint
specifies the name of the USVR. *entrypoint* can contain 1 to 8 alphanumeric or national characters. The name's first character is either an alphabetic character or a national character.

string
specifies information to be passed to the USVR when it receives control to verify authorization. *string* can contain 1 to 255 EBCDIC characters.

When the *string* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *string* in single-quotation marks (for example, AUTHORIZATION(*entrypoint,* '*DORIS*')).

When the *string* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, AUTHORIZATION(*entrypoint,* '*CA''ROL*')).

You can code *string* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, AUTHORIZATION(*entrypoint,* X'E2E4C5') is the same as AUTHORIZATION(*entrypoint,* SUE). The string can contain up to 255 hexadecimal characters when expressed in this form, resulting in up to 128 bytes of information.

CODE(*code* **)**
specifies a code name for the path. If an attempt is made to access a password-protected path without first supplying an appropriate password, a prompting message is issued to the operator's console. The prompting message includes the code name, which identifies the path without revealing its entryname.

This parameter can be coded, but only has effect when the path's master password is not null. A prompting message is issued only when the user hasn't already supplied the appropriate password. When CODE is not specified, the prompting message identifies the path with its entryname.

code
can contain 1 to 8 EBCDIC characters.

When the *code* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *code* in single-quotation marks (for example, CODE('*DORIS*')).

When the *code* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, CODE('*CA''ROL*')).

You can code *code* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, CODE(X'E2E4C5') is the same as CODE(SUE).

TO(*date* **) | FOR(** *days* **)**
specifies the retention period for the path. The path is not automatically deleted when the expiration date is reached. When a retention period is not speciifed, the path can be deleted at any time.

The maximum number that can be specified for *days* is 9999. If the number specified is 0 through 1830, the retention period is the number of days specified. If the number specified is between 1831 and 9999, the retention period is through the year 1999.

FOR(*days* **)**
specifies the number of days for which the entry is to be kept before it is allowed to be deleted. *days* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

TO(*date* **)**
specifies the date, in the form *yyddd*, where *yy* is the year and *ddd* is the Julian date (001, for January 1, through 365, for December 31), through which the path is to be kept before it is allowed to be deleted.

MASTERPW(*password* **)**
specifies a master password for the path. The master password allows all operations against the path.

If the master password is not specified, the path's highest-level password propagates upward and becomes the password for all higher levels, including the master password.

If all passwords are null, ATTEMPTS, AUTHORIZATION, and CODE can be coded but have no effect until the master password is specified.

CONTROLPW(*password* **)**
specifies a control password for the path. Control-interval processing is not permitted through a path; the control password will permit read and write operations against the base cluster.

If a read or update password is the only password specified for the object, it (the highest-level password) propagates upward and becomes the password for all higher unspecified levels.

UPDATEPW(*password* **)**
specifies the update password for the path. The update password permits read and write operations against the base cluster's data records.

If a read password is the only password specified for the object (that is, it is the highest-level password), it propagates upward and becomes the password for all higher levels. If you specify a higher-level password and don't specify an update password, the update password is null.

READPW(*password* **)**
specifies a read password for the path. The read password permits read operations against the base cluster's data records.

password
is a 1 to 8 EBCDIC-character password.

When the password contains a special character (that is, a comma [,],

semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the password in single-quotation marks (for example, READPW('*DORIS*')).

When the password contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, READPW('*CA"ROL*')).

You can code *password* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, READPW(X'E2E4C5') is the same as READPW(SUE).

OWNER(*ownerid* **)**
specifies the identification of the path's owner.

ownerid
can contain 1 to 8 EBCDIC characters.

When the *ownerid* contains a special character (that is, a comma [,], semicolon [;], blank [b], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *ownerid* in single-quotation marks (for example, OWNER('*DORIS*')).

When the *ownerid* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, OWNER('*CA"ROL*')).

You can code *ownerid* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, OWNER(X'E2E4C5') is the same as OWNER(SUE).

Note to TSO users: If the owner is not identified with the OWNER parameter, the TSO user's *userid* becomes the *ownerid*.

Model (Path)

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

You can use some attributes of the model and override others by explicitly specifying them in the definition of the path. When you do not want to add or change any attributes, you specify only the entry-type (PATH), the path's name, its alternate-index's (or cluster's) name, and the model-entry's name.

DEFINE	**PATH(-** **NAME(***entryname***)-** **PATHENTRY(***entryname* [*/ password*]**)-** **MODEL(***entryname* [*/ password*] [b *catname* [*/ password*]]))

MODEL (*entryname* [*/ password*]
[b *catname* [*/ password*]])
identifies an existing path entry that is to be used as a model for the path being defined. For details about how a model is used, see "How to Use One Object as a Model for Another Object."

entryname
names the entry to be used as a model. *entryname* must name a path entry.

password
specifies a password. If the model entry is password protected and it is cataloged in a password-protected catalog, you must supply the read (or higher level) password of either the model entry or its catalog. If you specify both passwords, the catalog's password is used.

If you are not specifying new protection attributes for the path (that is, the model's passwords and protection attributes are being copied), you must supply the master password of either the model entry or its catalog.

catname
names the model-entry's catalog. You must identify the catalog that contains the model entry for either of these cases:

- You specify the catalog's password instead of the model-entry's password
- The model-entry's catalog is not identified with a JOBCAT or STEPCAT DD statement, and is not the master catalog.

If the catalog's volume is physically mounted, it 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 a catalog is selected when the catalog's name is not specified.

Update (Path)

The UPDATE parameter allows you to specify whether the base cluster's upgrade set (that is, the group of alternate indexes that is updated whenever the base cluster is opened for addition, modification, or erasure of its data records) is allocated when the path is opened for processing.

UPDATE | NOUPDATE
specifies whether the base cluster's upgrade set is to be allocated when the path's cluster is allocated with a DD statement.

The *upgrade set* is a group of alternate indexes associated with the base cluster. The alternate indexes are opened whenever the base cluster is opened. When records in the base cluster are modified or deleted, and when records are added to the base cluster, each alternate index in the base cluster's upgrade set is modified to reflect the change in the cluster's data, just as a key-sequenced cluster's index is modified each time the cluster's data changes.

When a path points to a base cluster that has a large upgrade set (that is, many alternate indexes are associated with the base cluster), and the path is defined with the NOUPDATE attribute, you can open the path, and consequently the base cluster, and none of the alternate indexes will be opened.

NOUPDATE specifies that, when opening the path, the path's base cluster is to be allocated and the base cluster's upgrade set isn't to be allocated. You can specify the NOUPDATE attribute for the path even though the UPGRADE attribute is set for one of the base cluster's alternate indexes.

Catalog (Path)

The catalog parameter allows you to supply the name and password of the catalog in which the path is to be defined.

CATALOG(*catname* [*/ password*] **)**
identifies the catalog that contains the entry of the cluster or alternate index named in the PATHENTRY parameter.

See "Order of Catalog Use: DEFINE" for information about the order in which a catalog is selected if the catalog's name is not specified.

If the cluster's or alternate-index's entry is password protected and its catalog is also password protected, you must specify the master password for either the entry or the catalog.

catname
specifies the catalog's name.

password
specifies the catalog's master password.

If the catalog's volume is physically mounted, it is dynamically allocated. The volume must be mounted as permanently resident or reserved.

DEFINE PATH Example

Define a Path: Example 1

In this example, a path is defined. Previous examples illustrated the definition of the path's alternate index, EXAMPLE.AIX, and the alternate index's base cluster, EXAMPLE.KSDS2. The alternate index, path, and base cluster are defined in the same catalog, AMASTCAT.

```
//DEFPATH  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DEFINE PATH -
               (NAME(EXAMPLE.PATH) -
               PATHENTRY(EXAMPLE.AIX/AIXMRPW) -
               READPW(PATHRDPW) ) -
          CATALOG(AMASTCAT/MCATUPPW)
/*
```

The job control statement is:

- 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 PATH command builds a path entry to define the path EXAMPLE.PATH. A copy of the path entry is placed in the catalog recovery area, since the path entry is being defined into a recoverable catalog. The catalog recovery area on volume VSER04 is dynamically allocated. The command's parameters are:

- NAME, which specifies that the path's name is EXAMPLE.PATH.
- PATHENTRY, which identifies the alternate index, EXAMPLE.AIX, that the path provides access to.
- READPW, which specifies the path's read password, PATHRDPW.
- CATALOG, which in this example supplies the master catalog's update password, MCATUPPW.

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*])]

The parameters of the DEFINE SPACE command can be grouped as follows:

- Allocation, which specifies the amount and type of space to be allocated to the data space, and the volume that is to contain the data space (VOLUMES, CYLINDERS, RECORDS, TRACKS, RECORDSIZE, FILE, and CANDIDATE).
- Catalog, which identifies the catalog in which the data space is to be defined (CATALOG).

"Appendix F: Command Parameters Summary" contains a table for the DEFINE SPACE command that shows each parameter, its abbreviation, its default value (if any), and an example of its use.

DEFINE SPACE Parameters

SPACE
specifies that a data space is to be defined. You can also use the DEFINE SPACE command to reserve a volume for VSAM's future use. The SPACE keyword is followed by the allocation parameters, enclosed in parentheses, that describe the space, and by the catalog parameter that identifies the catalog that owns the volume.

Allocation (Space)

The allocation parameters are used to specify:

- The volume on which the data space is to be allocated, or the volume that is to be identified as a candidate volume.
- The amount of space to be allocated for the data space.

VOLUMES(*volser* [b *volser* ...])
specifies the volumes on which data spaces are to be defined. If two or more volumes are specified and an amount of space is specified (that is, TRACKS, CYLINDERS, or RECORDS and 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. When the volume already has a data space on it, subsequent data spaces can be defined and must be cataloged in the catalog that owns the volume. A data space can be defined on a mass storage volume.

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.

FILE(*dname* **)**
specifies the name of the DD statement that identifies the device type and volumes to be used for space allocation. You cannot use concatenated DD statements to describe more than one volume.

If FILE is not specified and the volume is physically mounted, the volume(s) identified with the VOLUMES parameter is dynamically allocated. Each dynamically allocated volume must be mounted as permanently resident or reserved.

TRACKS(*primary* [*secondary*]) |
CYLINDERS(*primary* [*secondary*]) |
RECORDS(*primary* [*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. However, if you specify TRACKS or RECORDS and if the minimum number of tracks should exceed a cylinder, space is allocated in terms of cylinders.

primary
specifies the initial amount of space (primary extent) to be allocated to the data space. *primary* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

secondary
specifies the amount of space that is to be allocated to the data space each time it is extended. Once the primary extent is filled, the data space can expand to include a maximum of 15 secondary extents if you have specified a secondary-extent amount. *secondary* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

The secondary space allocation is rounded upwards to the nearest whole cylinder regardless of your specification in terms of TRACKS or RECORDS. Consequently, the total amount of space allocated to the data space might exceed the predicted limit.

When the data space is the first data space on a volume that belongs to a recoverable catalog, the *primary* amount is increased by one cylinder.

You must specify one, and only one, of the following parameters: CANDIDATE, CYLINDERS, TRACKS, or RECORDS. When you specify RECORDS, you must also specify RECORDSIZE. When you specify CYLINDERS or TRACKS, you cannot specify RECORDSIZE.

RECORDSIZE(*average* *maximum* **)**
specifies the average and maximum lengths, in bytes, of the records.

average and *maximum* can be any integer value between 1 and 32,761, expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. When you specify the RECORDS parameter, you must also specify

RECORDSIZE. Otherwise, you cannot specify RECORDSIZE.

CANDIDATE
specifies that the volumes listed in the VOLUMES parameter are reserved for future use by VSAM. An ownership indicator is set in a format-4 DSCB in the volume's table of contents (VTOC). The volumes are reserved for use by the catalog, but no space is allocated. Each volume so reserved is called a candidate volume. If CANDIDATE is not specified, you must specify one of the following parameters: CYLINDERS, TRACKS, or RECORDS.

If a VSAM data space already exists on the volume, you cannot specify CANDIDATE. (You can, however, use the ALTER command to identify the volume as a candidate volume for a VSAM object other than the volume's catalog.)

Catalog (Space)

The catalog parameter is used to supply the name and password, when required, of the catalog that contains the volume entry that describes the volume on which the data space is to be allocated. If the DEFINE SPACE command is used to allocate the volume's first VSAM data space or to identify the volume as a candidate volume, the catalog parameter identifies the catalog that is to contain the volume's catalog entry and is to own the volume. (For more information about VSAM volume ownership, see the section "VSAM Volume Ownership" in the "Introduction.")

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

If the catalog's volume is physically mounted, it is dynamically allocated. See "Order of Catalog Use: DEFINE" for information about the order in which catalogs are used.

catname
specifies the name of the catalog.

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

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
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE SPACE -
          (CYLINDERS(100 10) -
          VOLUMES(VSER05))
/*
```

VSAM assigns a name to the data space and keeps track of it for future data set 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 statement is:

- 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.
- 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.

DEFINE USERCATALOG

When you define a catalog, you can specify attributes for the catalog as a whole and for the catalog's components.

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* **[ъ** *secondary***]) |** **CYLINDERS(***primary* **[ъ** *secondary***]) |** **RECORDS(***primary* **[ъ** *secondary***])}** **[BUFFERSPACE(***size* **| 3072)]** **[RECOVERABLE | NOTRECOVERABLE]** **[MASTERPW(***password***)]** **[CONTROLPW(***password***)]** **[UPDATEPW(***password***)]** **[READPW(***password***)]** **[CODE(***code***)]** **[ATTEMPTS(***number* **| 2)]** **[AUTHORIZATION(***entrypoint* **[ъ** *string***])]** **[OWNER(***ownerid***)]** **[TO(***date***) | FOR(***days***)]** **[WRITECHECK | NOWRITECHECK]** **[DESTAGEWAIT | NODESTAGEWAIT]** **[MODEL(***entryname* **[/** *password***]** **[ъ** *catname* **[/** *password***])])** **[DATA** **([TRACKS(***primary* **[ъ** *secondary***]) |** **CYLINDERS(***primary* **[ъ** *secondary***]) |** **RECORDS(***primary* **[ъ** *secondary***])]** **[WRITECHECK | NOWRITECHECK]** **[RECOVERABLE | NOTRECOVERABLE]** **[DESTAGEWAIT | NODESTAGEWAIT]** **[BUFFERSPACE(***size***)])]** **[INDEX** **([TRACKS(***primary***) |** **CYLINDERS(***primary***) |** **RECORDS(***primary***)]** **[WRITECHECK | NOWRITECHECK]** **[DESTAGEWAIT | NODESTAGEWAIT])]** **[CATALOG(***catname* **[/** *password***])]**

DEFINE USERCATALOG Parameters: Summary

The parameters of this command are described in the following groups:

- Entry type, which describes the MASTERCATALOG and USERCATALOG parameters. These parameters specify that a user catalog is to be defined.
- Name, which describes the NAME parameter.
- Allocation, which describes the FILE, VOLUME, TRACKS, CYLINDERS, RECORDS, BUFFERSPACE, RECOVERABLE, and NOTRECOVERABLE 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's data space, and the space to be used for buffers when the catalog is accessed. They also determine whether a catalog recovery area is to be defined along with the catalog.
- Protection and integrity, which describes the MASTERPW, CONTROLPW, UPDATEPW, READPW, CODE, ATTEMPTS, 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 catalog from which attributes are to be copied for the definition of a new user catalog.
- Components, which describe the DATA and INDEX parameters. These parameters allow you to specify attributes separately for the catalog's data and index components.
- Catalog, which describes the CATALOG parameter. This parameter is used to supply the name and password, when required, of the master catalog.

"Appendix F: Command Parameters Summary" contains a table for the DEFINE USERCATALOG command that shows each parameter, its abbreviation, its default value (if any), and an example of its use.

DEFINE USERCATALOG Parameters

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

USERCATALOG(*options* **)**
specifies that a user catalog is to be defined. USERCATALOG is followed by the parameters specified for the catalog as a whole.

(In VS2, when you specify MASTERCATALOG the result is the same: a user catalog is created.)

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.
- Whether the catalog is recoverable.

See "Estimating the Catalog's Space Requirements" 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 catalog's 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. The volume cannot be currently owned by any other VSAM catalog. A user catalog can be defined on a mass storage volume.

A volume serial number, volser, may contain one to six alphanumeric, 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 VOLUME parameter interacts with other DEFINE CATALOG parameters. You should take care to ensure that the volume you specify for the catalog can satisfy the catalog's other attributes:

- CYLINDERS, RECORDS, TRACKS: The volume contains enough unallocated space to satisfy the catalog's primary space requirement. (Space on the volume might already be allocated to nonVSAM data sets and system data sets. However, the volume cannot be owned by another VSAM catalog—that is, no space on the volume can already be allocated to VSAM objects.)
- FILE: The volume information supplied with the DD statement is consistent with the information specified for the catalog and its components.

TRACKS(*primary* [ð *secondary*]) |
CYLINDERS(*primary* [ð *secondary*]) |
RECORDS(*primary* [ð *secondary*])

specifies the amount of space to be allocated in terms of tracks, cylinders, or number of records. You can specify the amount of space as a parameter of USERCATALOG, as a parameter of USERCATALOG and DATA, or as a parameter of USERCATALOG, DATA, and INDEX.

When you specify less than one cylinder of space, and the catalog is defined as recoverable (see the RECOVERABLE parameter), the *primary* amount defaults to (the equivalent of) one cylinder.

primary [ð *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.

primary and *secondary* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. Secondary allocation should be specified in case the catalog has to be extended.

BUFFERSPACE(*size* | **3072**)

specifies the minimum space, in bytes, to be provided for buffers when using the catalog. Decimal values you can specify are 3072, 4096, 5120, 6144, 7168, and 8192. If a value greater than 8192 is specified, the BUFFERSPACE size defaults to 8192.

size

is the amount of space, in bytes, to be provided for buffers. *size* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

RECOVERABLE | NOTRECOVERABLE

specifies that a catalog recovery space is to be created on each volume owned by the catalog.

On the catalog's volume, the catalog recovery space is allocated from the catalog's data space. On subsequent volumes owned by the catalog, the catalog recovery space is allocated from the first data space defined on the volume.

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 data 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.

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-protected in order to prevent unauthorized users from cataloging data sets. For more details on how passwords can be used, see "Data Security and Protection."

MASTERPW(*password* **)**
specifies a master 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. The master password allows all operations; it is required to open the catalog as a data set.

If a master password is not specified for the catalog, the catalog's highest-level password propagates upward and becomes the password for all higher levels, including the master password. The master password is null when no other passwords are specified.

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

If a read or update password is the only password specified for the catalog, it (the highest-level password) propagates upward and becomes the password for all higher levels. When the master password is specified but the control password is not specified, the control password is null.

UPDATEPW(*password* **)**
specifies an update password for the catalog being defined. The update password permits entries to be added to the catalog being defined. When you anticipate using the catalog to contain many nonVSAM entries, you should consider allowing the catalog's update password to be null (that is, to specify a master password but not an update password). A null update password allows you to delete nonVSAM entries without having to supply the catalog's password.

The phrase "an update or higher-level password" means:

- If the update password exists, you are required to correctly supply either the update, control, or master password before Access Method Services processing continues.
- If the update password is null (that is, it doesn't exist), no prompting message is issued and no password is required.

You can protect the catalog and its VSAM entries by specifying a master password. You can allow the unrestricted deletion of nonVSAM entries and the addition of all types of entries by *not* specifying an update password.

If a read password is the only password specified for the catalog (that is, it is the highest-level password), it propagates upward and becomes the password for all higher levels. If you specify a higher-level password and don't specify an update or read password, the update password is null.

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

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 (X' *password* '), where two hexadecimal characters represent an EBCDIC character.

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 (X' *code* '), where two hexadecimal characters represent an EBCDIC character.

ATTEMPTS(*number* | **2** **)**
specifies the maximum number of times the operator can try to enter a correct password in response to a prompting message.

number
is an integer from 0 to 7 and can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. If 0 is specified, the operator is not prompted and is not allowed to enter a password from the console.

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 catalog. 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 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.

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 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 appears within a string, it must be coded as two single quotation marks if the string is enclosed in single quotation marks.

string can be expressed in hexadecimal (X' *string* '), where two hexadecimal characters represent an EBCDIC character. The string can contain up to 255 hexadecimal characters when expressed in this form, resulting in up to 128 bytes of information.

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 (X' *ownerid* '), where two hexadecimal characters represent an EBCDIC character.

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 catalog 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.

days
can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

WRITECHECK | NOWRITECHECK
specifies whether the catalog is to be checked by a direct-access device operation called write-check when a record is written to the device. If WRITECHECK is specified, a record is written and then read, without data transfer, to test for the data check condition.

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.

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.

If the user catalog isn't stored on a mass storage volume, the attribute is ineffective until the direct-access storage volume the 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 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.

MODEL(*entryname* [/ *password*][ъ *catname* [/ *password*]] **)**
specifies that an existing master or user catalog is to be used as a model for the user catalog being defined. For details about how a model is used, see "How to Use One Object as a Model for Another Object."

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 you specify both passwords (that is, the password following *entryname* and the password following *catname*), the password following *catname* is used for authorization. If the protection attributes are to be copied, enter the master password of the catalog being used as a model. If passwords are not to be copied, any password except the master can be used.

catname
specifies the name of the catalog to be used as a model. This parameter is required if the model catalog is neither the master catalog nor a catalog identified by a JOBCAT or STEPCAT DD statement.

Components (Catalog)

Attributes can be specified separately for the catalog's data and index components.

DATA(*optional parameters* **)**
specifies the data component's attributes. If an optional parameter is not specified as a parameter of DATA, the data component shares the value or attribute specified for the catalog as a whole.

Attributes specified for the data component override similar, but conflicting, attributes specified for the catalog as a whole. Attributes specified as parameters of DATA also override similar, but conflicting, attributes of a model catalog's data component.

Parameters that can be specified for the catalog's data component are:

- BUFFERSPACE: Specifies the amount of buffer space that the user's program is to provide from its address space when the program opens the data component as a separate VSAM object.
- CYLINDERS, RECORDS, or TRACKS: You can specify an amount of space to be allocated to the data component. You can also specify an amount of space for the catalog as a whole.
- WRITECHECK or NOWRITECHECK: Specifies whether the catalog's records are to be checked by the direct-access device when they are written into the data component.
- RECOVERABLE or NOTRECOVERABLE: Specifies whether a catalog recovery area is to be created on each volume owned by the catalog.
- DESTAGEWAIT or NODESTAGEWAIT: Specifies whether the catalog's data component, stored on a mass storage volume, is to be destaged synchronously or asynchronously with respect to the program that closes it.

INDEX (*optional parameters*)
specifies the index component's attributes. If an optional parameter is not specified as a parameter of INDEX, the index component shares the value or attribute specified for the catalog as a whole.

Attributes specified for the index override similar, but conflicting, attributes specified for the catalog as a whole. Attributes specified as parameters of INDEX also override similar, but conflicting, attributes of a model catalog's index component.

Parameters that can be specified for the catalog's index component are:

- CYLINDERS, RECORDS, or TRACKS: You can specify an amount of space to be allocated to the index component. When you specify one of the space-quantity parameters as a parameter of INDEX, you must also specify the same type of space-quantity parameter as a parameter of DATA. TRACKS, CYLINDERS, or RECORDS must have been specified as a parameter of DATA to be specified as a parameter of INDEX.
- WRITECHECK or NOWRITECHECK: Specifies whether the index records are to be checked by the direct-access device when they are written into the index.
- DESTAGEWAIT or NODESTAGEWAIT: Specifies whether the catalog's index component, stored on a mass storage volume, is to be destaged synchronously or asynchronously with respect to the program that closes it.

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, its update (or higher level) password must be provided in this parameter or in response to prompting.

DEFINE USERCATALOG Examples

Define a User Catalog: Example 1

In this example, a user catalog is defined.

```
//DEFCAT3  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE USERCATALOG( -
               NAME(D27UCAT1) -
               MASTERPW(MRPWD27) -
               UPDATEPW(UPPWD27) -
               FOR(365) -
               CYLINDERS(150 5) -
               VOLUME(VSER02) ) -
          DATA( -
               CYLINDERS(8 5) ) -
          INDEX( -
               CYLINDERS(5) ) -
          CATALOG(AMASTCAT/MRCATPW2)
/*
```

The job control statement is:

- 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.
- CYLINDERS, which specifies that 150 cylinders are to be allocated for the 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 catalog is to reside on volume VSER02.
- DATA and INDEX, which specify that VSAM is to allocate 13 cylinders for the 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 VSAM objects that reside in a nonunique data space. 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 entry is to be defined in the master catalog, AMASTCAT.CATALOG provides the catalog's update password, MRCATPW2, so that the operator is not prompted to provide it.
- Values and attributes that apply by default to the catalog are:
  - BUFFERSPACE = 3072 bytes
  - ATTTEMPTS = 2
  - NOTRECOVERABLE
  - NOWRITECHECK
  - NODESTAGEWAIT
- Null attributes and values of the catalog are:
  - CONTROLPW and READPW passwords
  - CODE
  - AUTHORIZATION
  - OWNER

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.

```
//DEFCAT4  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//STEPCAT  DD      DSNAME=D27UCAT1,DISP=SHR
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE USERCATALOG( -
                NAME(D27UCAT2) -
                MASTERPW(MASTD27) -
                UPDATEPW(UPDD27) -
                CYLINDERS(150 5) -
                VOLUME(VSER03) -
                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.
- 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 catalog D27UCAT2. Its parameters are:

- NAME, which names the catalog, D27UCAT2.
- MASTERPW and UPDATEPW, which specify master and update passwords for the catalog: MASTD27 and UPDD27
- CYLINDERS, which specifies that 150 cylinders are to be allocated for the catalog's data space. When the catalog's data space is extended, it is to be extended in increments of 5 cylinders.
- VOLUME, which specifies that the 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. D27UCAT1 contains the catalog entries that describe itself (the master catalog, AMASTCAT, contains a user-catalog connector entry which points to D27UCAT1)—this is why D27UCAT1 is specified as MODEL's *catname* subparameter. Values and attributes that apply to D27UCAT2 as a result of using D27UCAT1 as a model are:
  - FOR = 365 days (retention period)
  - CYLINDERS = 8 (primary) and 5 (secondary)
  - BUFFERSPACE = 3072 bytes
  - ATTEMPTS = 2
  - NOTRECOVERABLE
  - NOWRITECHECK

  - NODESTAGEWAIT
  - CONTROLPW and READPW passwords are null
  - CODE is null
  - AUTHORIZATION is null
  - OWNER is null
- CATALOG, which specifies that the user catalog connector is to be defined in the AMASTCAT catalog. The update password of AMASTCAT is MRCATPW2.

Defining a User Catalog and Determining the Catalog's Space Requirements: Example 3

In this example, a user catalog is defined. 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 key-sequenced clusters with alternate indexes to be upgraded
- 5 entry-sequenced clusters with alternate indexes to be upgraded
- 10 alternate indexes
- 10 paths
- 200 aliases
- 10 generation data sets
- 1000 nonVSAM 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 and tracks, is required for the primary allocation of a to-be-defined catalog in your system, see the previous section, "Estimating the Catalog's Space Requirements." The number of records for the primary allocation of the catalog in this example is shown in the filled-in worksheet in Figure 9.

When you define the catalog's space parameters, use the following format:

```
DEFINE USERCATALOG -
     (TRACKS(prim secn) ... ) -
     DATA -
     (RECORDS(prec srec) ... )
```

where:

- prec = number of entry records for the catalog's data component, or the value of N:

  prec = 1660 records

- srec = number of records for the data component's secondary extent:

  srec = .2N = 332 records

Variable Quantities	Formulas	Estimates
Basic requirement = 10 records		10
A = number of key-sequenced clusters	Ax3	300
A^1 = number of key-sequenced clusters with alternate indexes to be upgraded	A^1x4	20
B = number of entry-sequenced clusters	Bx2	20
B^1 = number of entry-sequenced clusters with alternate indexes to be upgraded	B^1x3	15
C = number of relative-record clusters	Cx2	0
D = number of alternate indexes	Dx3	30
E = number of path entries	E	10
F = number of nonVSAM data set entries	F	1000
G = number of generation data group entries	G	10
H = number of alias entries	H	200
I = number of page spaces	Ix2	0
J = number of volumes, depending on device type, owned by the catalog:		
J^1 = number of 2305 volumes	J^1x2	0
J^2 = number of 2314/2319 volumes	J^2x3	20
J^3 = number of 3330 and 3340/3344 volumes	J^3x4	0
J^4 = number of 3330 Model 11 and 3350 volumes	J^4x6	
K = for each key-sequenced cluster and alternate index (KSIS) with space on more than two volumes, add "1" for each additional group of one to five volumes:		
K^1 = number of KSDSs with 3 to 7 volumes	K^1	0
K^2 = number of KSDSs with 8 to 12 volumes	K^2x2	0
K^3 = number of KSDSs with 13 to 17 volumes	K^3x3	0
L = for each entry-sequenced cluster and relative-record cluster (ESDS) with space on more than five volumes, add "1" for each additional group of one to eight volumes:		
L^1 = number of ESDSs with 6 to 13 volumes	L^1	0
L^2 = number of ESDSs with 14 to 21 volumes	L^2	0
M = for each group of four data spaces on a volume, add "1"	M	25
N = number of entry records required for the catalog's data component (total of above)	N	1660

Figure 9. Completed Worksheet for Determining a Catalog's Space Requirements

- prim = the minimum amount of space for the catalog data space's primary allocation. When you expect to have other VSAM objects in the catalog's data space, the value of prim should be larger than the calculated minimum. The calculated value of prim should be rounded upward to a whole number of tracks.

  Because the catalog resides on a 3330 volume, the value of prim is:

  prim = .105N + 3.3 = 177.6 = 178 tracks

- secn = the minimum amount of space for each secondary extent of the catalog's data space. The calculated value of secn should be rounded upward to a whole number of tracks:

  secn = .1prim = 18 tracks

The number of tracks required for the catalog's primary extent is at least 156. If VSAM extends the catalog, it should extend it in increments of 16 tracks.

```
//DEFCAT5  JOB     ...
//STEP1    EXEC    PGM=IDCAMS
//SYSPRINT DD      SYSOUT=A
//SYSIN    DD      *
     DEFINE USERCATALOG( -
          NAME(MYCAT) -
          VOLUME(VSER04) -
          NOTRECOVERABLE -
          TRACKS(178 18) ) -
          DATA ( -
             RECORDS(1660 332))
/*
```

The job control statement is:

- 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.
- VOLUME, which specifies that the catalog is to reside on the volume whose serial number is VSER04.
- NOTRECOVERABLE, which specifies that no recovery space is to be allocated for this catalog.
- TRACKS, which specifies that the catalog's data space is to be 178 tracks. When the user catalog is extended, VSAM extends it in increments of 18 tracks.
- DATA (RECORDS), which specifies that the catalog's data component is to be allocated an amount of tracks that can contain at least 1660 (512-byte) catalog records.

  When the data component is extended, VSAM extends it in increments of (the number of tracks that is equivalent to) 332 catalog records.
- Values and attributes that apply by default to the catalog are:
  - BUFFERSPACE = 3072 bytes
  - ATTEMPTS = 2
  - NOWRITECHECK
  - NODESTAGEWAIT
- Null attributes and values of the catalog are:
  - All passwords
  - CODE
  - AUTHORIZATION
  - TO and FOR
  - OWNER

DELETE

The format of the DELETE command is:

DELETE	(*entryname* [/ *password*][ъ *entryname* [/ *password*]...]) [**CATALOG**(*catname* [/ *password*])] [**FILE**(*dname*)] [**PURGE** | **NOPURGE**] [**FORCE** | **NOFORCE**] [**ERASE** | **NOERASE**] [**SCRATCH** | **NOSCRATCH**] [**CLUSTER** | **SPACE** | **USERCATALOG** | **NONVSAM** | **ALTERNATEINDEX** | **PATH** | **ALIAS** | **GENERATIONDATAGROUP** | **PAGESPACE**]

"Appendix F: Command Parameters Summary" contains a table for the DELETE command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

DELETE Parameters

DELETE
specifies that an entry is to be removed from a catalog or a member is to be deleted from a nonVSAM partitioned data set.

(*entryname* [/ *password*][ъ *entryname*...])
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.

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 you are deleting a member of a nonVSAM partitioned data set, the *entryname* must be specified in the format: *pdsname (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. Only certain types of catalog entries can be password-protected. To delete a

- Data space, specify the catalog's update (or higher level) password.
- Cluster, specify its master password.
- Alternate index, specify its master password.
- Pagespace, specify its master password.
- User catalog, specify its master password with the entryname (that is, Access Method Services doesn't issue a prompting message to the operator or TSO terminal user in this case).
- Path, specify its master password.

CATALOG(*catname* [*/ password*] **)**
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.

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 with the *entryname.*

FILE(*dname* **)**
specifies the name of the DD statement that identifies the volume that contains the data set to be deleted if it is unique or nonVSAM (with SCRATCH) or identifies the entry to be deleted if erase processing is to be performed. When the data set resides on more than one volume (that is, it is a multi-volume data set), FILE identifies the DD statement which specifies all volumes. For erase processing, a JOBCAT or STEPCAT DD statement is required.

When the entryname is dynamically allocated, the data set's volume must be mounted as permanently resident or reserved. When you delete a catalog and also specify FORCE, you can use FILE and its associated DD statement to describe, allocate, and mount the volume(s) that are owned by the to-be-deleted catalog.

When the entry being deleted is in a recoverable catalog, FILE identifies the DD statement that describes the catalog recovery volume (that is, the volume whose recovery space contains a copy of the entry being deleted). When the data set resides on more than one volume, FILE identifies each volume. See "Restoring Catalog Entries After System Failure" for more details on catalog recovery.

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.

PURGE
specifies that the entry is to be deleted even if the retention period, specified in the TO or FOR parameter, has not expired.

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.

FORCE | NOFORCE
The FORCE parameter allows you to delete data spaces, generation data groups, and VSAM user catalogs without first ensuring that these objects are empty.

- When you delete space using the FORCE parameter, all VSAM data spaces on the volume(s) are scratched from the VTOC. VSAM's

ownership of the volume(s) is given up. Before the data spaces are scratched, VSAM verifies that no VSAM clusters on the volume are open. The clusters remain cataloged, but their entries are marked unuseable. Subsequently, you can delete the cluster from the catalog.

You cannot specify FORCE to delete the space on a volume that contains the VSAM master catalog or a user catalog. When you specify FORCE, you must supply the master password of the catalog that contains the volume's entry.

- When you delete a user catalog using the FORCE parameter, the catalog is deleted even though it contains catalog entries for objects that haven't been deleted. All data spaces are deleted from each volume owned by the catalog. The volumes are available to be used by the OS/VS2 system or to be assigned to other VSAM catalogs. All cataloged objects are automatically deleted, but the contents of each cluster and alternate index are not erased. NonVSAM data set entries in the catalog are deleted, but the nonVSAM data set's space on the volume is undisturbed. The nonVSAM data set can be located with its DSCB in the volume's VTOC. VSAM clusters and alternate indexes, even though they might reside in unique data spaces, cannot be located, because the data space's DSCB is scratched when the catalog is deleted.

  When you delete a user catalog using the FORCE parameter, you must supply the master password of the catalog being deleted.

- When you delete a generation data group using FORCE, the GDG entry is deleted even though it might point to generation data sets (that is, it might point to nonVSAM entries in the catalog). Each nonVSAM data set entry (that is, the entry for a generation data set) pointed to by the GDG base entry is deleted before the GDG base entry is deleted. However, the nonVSAM data set's space and contents on the volume is undisturbed. The data set can be located with its DSCB in the volume's VTOC.

NOFORCE, the default option, causes the DELETE command to terminate when you request the deletion of a nonempty data space, generation data group, or catalog.

ERASE | NOERASE
specifies whether the data component of a cluster or alternate index to be deleted is to be erased, that is, overwritten with binary zeros. This parameter will override whatever was coded when the cluster or alternate index was defined or last altered. This parameter should be specified only when a cluster or alternate index 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 qualifier in the entry's qualified name identifies the catalog (that is, the first qualifier is the catalog's name or alias).

ERASE
specifies that the data component is to be overwritten with binary zeros when the cluster or alternate index is deleted. If ERASE is specified, the volume that contains the data component must be mounted.

NOERASE
specifies that the data component is not to be overwritten with binary zeros when the cluster or alternate index is deleted.

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, ALTERNATEINDEX, 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. NOSCRATCH cannot be specified for VSAM objects in recoverable catalogs. See the section "VSAM Catalog Cleanup" in the chapter "Data Security and Protection" 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 qualifier in the entry's qualified name identifies the catalog (that is, the first qualifier is the catalog's name or alias).

You should specify NOSCRATCH if the Format-1 DSCB of a nonVSAM data set has already been scratched from the VTOC.

CLUSTER | SPACE | USERCATALOG | NONVSAM | ALTERNATEINDEX | PATH | ALIAS | GENERATIONDATAGROUP | PAGESPACE
specifies the type of the object or entry to be deleted. If the object to be deleted is a catalog or data space, 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 object to be deleted is a cluster, its associated data and index entries, and any related paths and alternate indexes.

USERCATALOG
specifies that the object to be deleted is a user catalog. 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.

A user catalog can be deleted when it is empty (that is, it contains only its self-describing entries and its volume's volume entry), or when you specify the FORCE parameter.

PATH
specifies that a path entry is to be deleted. No entries associated with the path are deleted.

ALTERNATEINDEX
specifies that the object to be deleted is an alternate index and its data and index entries. When a path entry is associated with the alternate index, the path entry is also deleted. When the alternate index has the to-be-upgraded attribute and it is the only such alternate index associated with the base cluster, the base cluster's upgrade-set entry is also deleted.

SPACE
specifies that all empty VSAM data spaces on a volume are to be deleted. This parameter is required when you want to delete the volume's empty data spaces. A data space can be deleted when it is empty, or when you specify the FORCE parameter. If all data spaces on the volume have been deleted and the volume is not a candidate volume, its volume entry is also deleted. When the catalog that owns the volume is recoverable, the data space that contains the volume's recovery area is not deleted unless all other data spaces on the volume have been deleted.

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).*

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 when it is empty, or when you specify the FORCE parameter.

PAGESPACE specifies that an inactive page space is to be deleted. A page space is identified as "active" during the operator's IPL procedure.

DELETE Examples

Deleting a Key-Sequenced VSAM Cluster: Example 1

In this example, a key-sequenced cluster is deleted. Alternate indexes and paths related to the key-sequenced cluster are deleted automatically by Access Method Services. Access Method Services will dynamically allocate the key-sequenced data set so that the data can be overwritten (as specified by the ERASE option).

```
//DELET1   JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DELETE -
          D40.EXAMPLE.KSDS1 -
          PURGE -
          ERASE -
          CATALOG(D27UCAT2/MRPWD27)
/*
```

The job control statement is:

- 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:

- D40.EXAMPLE.KSDS1, which is the entryname of the object being deleted. D40.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 2

In this example, an entry-sequenced VSAM cluster is deleted. Alternate indexes and paths related to the entry-sequenced cluster are automatically deleted by Access Method Services. Next, all empty VSAM data spaces on volume VSER03 are deleted and the volume's volume entry is deleted.

```
//DELET2   JOB  ...
//JOBCAT   DD   DSNAME=D27UCAT2,DISP=OLD
//STEP1    EXEC PGM=IDCAMS
//DD2      DD  VOL=SER=VSER03,UNIT=3330,DISP=OLD
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
     DELETE -
          EXAMPLE.ESDS2/DEPT26M -
          PURGE -
          CLUSTER
     DELETE -
          (VSER03) -
          SPACE -
     FILE (DD2)
/*
```

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 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. Alternate indexes and paths related to the

entry-sequenced cluster are also deleted regardless of their retention periods or dates.

- 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 DD2 DD statement ensures that volume VSER03 is mounted when its data spaces are deleted. In this example JCL has been used instead of allowing Access Method Services to dynamically allocate the volume. If the volume were mounted as permanently RESIDENT or RESERVED, Access Method Services could dynamically allocate the volume without the need of JCL or the FILE parameter. 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.
- FILE, which specifies the dname of a DD statement that describes the VSAM volume and causes it to be mounted.

Deleting Two Key-Sequenced Clusters: Example 3

In this example, two key-sequenced clusters, MYDATA and ENTRY, are deleted from a recoverable catalog. This example illustrates how more than one cataloged object is deleted with a single DELETE command.

```
//DELET3   JOB    ...
//JOBCAT   DD     DSNAME=MYCAT,DISP=SHR
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DELETE -
          (D40.MYDATA -
           ENTRY) -
          PURGE -
          CLUSTER
/*
```

The job control statements are:

- JOBCAT DD, which makes the catalog MYCAT available for the job.
- 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 D40.MYDATA and ENTRY. The high-level qualifier of the obect D40.MYDATA causes the catalog D27UCAT1 to be searched, since it is assumed that D40 is an alias of the catalog D27UCAT1. The object ENTRY is found in the catalog MYCAT, which is made available to the job through the JOBCAT DD statement. Both entries will be dynamically allocated so that their respective catalog recovery areas can be updated. The DELETE command's parameters are:

- D40.MYDATA and ENTRY, which identify the objects to be deleted. D40.MYDATA and ENTRY are the entrynames of two key-sequenced clusters. It is assumed that neither cluster is password-protected at this time. If either cluster is password-protected, the operator is prompted to supply the cluster's master password.
- PURGE, which specifies that the cluster is to be deleted regardless of its retention period or date.
- CLUSTER, which specifies that D40.MYDATA and ENTRY are clusters (that is, the names identify cluster catalog records).

Deleting a NonVSAM Data Set's Entry: Example 4

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). A FILE parameter and its associated DD statement are provided to allocate the data set's volume. In this example, dynamic allocation is not used to provide catalog or volume allocation.

```
//DELET4   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT1,DISP=SHR
//STEP1    EXEC   PGM=IDCAMS
//DD1      DD     VOL=SER=VSER02,UNIT=2314,DISP=OLD,
//   DSNAME=EXAMPLE.NONVSAM
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     DELETE -
          (EXAMPLE.NONVSAM) -
          FILE (DD1) -
          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. The DELETE command's parameters are:

- EXAMPLE.NONVSAM, which is the entryname of the object to be deleted.
- FILE, which specifies the dname of a DD statement that describes the nonVSAM data set's volume and causes it to be mounted. When the data set is deleted, its DSCB entry in the volume's VTOC is removed.
- 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.

Deleting All Empty VSAM Data Spaces on a Volume: Example 5

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.

```
//DELET5  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. Since the volume's VTOC must be updated, Access Method Services can dynamically allocate the volume if the volume is mounted as permanently RESIDENT or RESERVED. 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 is 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 6

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 unless the FORCE parameter is specified.

```
//DELET6  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 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.

Deleting an Alias Entry: Example 7

In this example, an alias entry, EXAMPLE.NONVSAM1, is removed from catalog D27UCAT1.

```
//DELET7    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 Generically-Named Entries: Example 8

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.

```
//DELET8   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.
- 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 9

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.

```
//DELET9    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 is:

- 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 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, 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 Member of a Partitioned (NonVSAM) Data Set: Example 10

In this example, the MEM1 member of partitioned data set EXAMPLE.NONVSAM2 is deleted. Next, the nonVSAM data set itself is deleted.

```
//DELET10    JOB   ...
//JOBCAT     DD    DSNAME=D27UCAT1,DISP=SHR
//STEP1      EXEC  PGM=IDCAMS
//SYSPRINT   DD    SYSOUT=A
//SYSIN      DD    *
     DELETE -
          EXAMPLE.NONVSAM2(MEM1) -
          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.
- 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 set 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 the second part 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 11

In this example, page space SYS1.PAGE2 is deleted.

```
//DELET11    JOB   ...
//STEP1      EXEC  PGM=IDCAMS
//SYSPRINT   DD    SYSOUT=A
//SYSIN      DD    *
     DELETE -
          SYS1.PAGE2 -
          PURGE -
          PAGESPACE
/*
```

The job control statement is:

- 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 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.

EXPORT

The format of the EXPORT command, when it is used to move or disconnect a user catalog, is:

EXPORT	*usercatname* [*/ password*] **DISCONNECT**

The format of the EXPORT command, when it is used to move a cluster or alternate index, is:

EXPORT	*entryname* [*/ password*] **OUTFILE(***dname***) | OUTDATASET(***entryname***)** **[INFILE(***dname***)]** **[TEMPORARY | PERMANENT]** **[INHIBITSOURCE | NOINHIBITSOURCE]** **[INHIBITTARGET | NOINHIBITTARGET]** **[ERASE | NOERASE]** **[PURGE | NOPURGE]**

"Appendix F: Command Parameters Summary" contains a table for the EXPORT command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

EXPORT Parameters

EXPORT
specifies that a cluster, alternate index, or user catalog is to be moved from the system in which the command is executed.

entryname [*/ password*]
is a required parameter that names the cluster, alternate index, 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 or alternate index, you must supply its (the object'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. Also, the user catalog's alias entries are deleted from the master catalog.

DISCONNECT is a required parameter if EXPORT is issued for a user catalog. The volume that contains the user catalog can 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.

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. With the exception of blocksize, 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, established by the EXPORT command, are: physical sequential data organization, variable spanned blocked record format, and a record length of (the greater of) 268 | maximum data record length + 4. Blocksize is defaulted by the EXPORT command to 2048. Blocksize may be specified in the DD statement to override this default and improve performance.

When you use a DD statement to describe the output data set, you must identify the data set's catalog with a JOBCAT or STEPCAT DD statement unless:

- The data set's entry is in the master catalog, or
- The first qualifier of the data set's name identifies a catalog (that is, the first qualifier is the catalog's name or alias).

OUTDATASET(*entryname* **)**
specifies the name of the data set that is to receive the data being exported. If OUTDATASET is specified, the entryname is dynamically allocated.

INFILE(*dname* **)**
specifies the name of the DD statement that identifies the cluster or alternate index to be exported. If the cluster or alternate index has been defined with a maximum logical record length greater than 32,760 bytes, EXPORT processing terminates with an error message.

In addition to the DD statement for INFILE, you must identify the entry's catalog with a JOBCAT or STEPCAT DD statement unless:

- The object's entry is in the master catalog, or
- The first qualifier of the object's name identifies a catalog (that is, the first qualifier is the catalog's name or alias).

When INFILE and its DD statement aren't specified for a to-be-exported object, it is dynamically allocated.

TEMPORARY | PERMANENT
specifies whether the cluster or alternate index to be exported is to be deleted from the original system.

TEMPORARY
specifies that the cluster or alternate index is not to be deleted from the original system. The object 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 IMPORT command deletes the original copy, defines the new object, and copies the data from the portable copy into the newly defined object.

PERMANENT
specifies that the cluster or alternate index is to be deleted from the original system. Its storage space is freed. If its retention period has not yet expired, you must also code PURGE.

INHIBITSOURCE | NOINHIBITSOURCE
specifies whether the original data records (that is, the data records of the *source* cluster or alternate index) can be accessed for any operation other than retrieval after its copy is exported. This specification can later be altered through the ALTER command.

INHIBITSOURCE
specifies that the source object in the original system cannot be accessed for any operation other than retrieval. This parameter can be specified only when the object is to be temporarily exported (that is, a backup copy of the object is made; the object itself remains in the original system).

NOINHIBITSOURCE
specifies that the source object in the original system can be accessed for any kind of operation.

INHIBITTARGET | NOINHIBITTARGET
specifies whether the copy's records (that is, the *target* alternate index or cluster) can be accessed for any operation other than retrieval after it has been imported to another system. This specification can be altered through the ALTER command.

INHIBITTARGET
specifies that the target object cannot be accessed for any operation other than retrieval after it has been imported into another system.

NOINHIBITTARGET
specifies that the target object can be accessed for any type of operation after it has been imported into another system.

ERASE | NOERASE
specifies whether the data component of the cluster or alternate index to be exported is to be erased (that is, overwritten with binary zeros). This parameter overrides whatever was specified when the object was defined or last altered. This parameter can be specified only if the object is to be permanently exported (that is, deleted from the original system).

PURGE | NOPURGE
specifies whether the cluster or alternate index to be exported is to be deleted from the original system regardless of the retention period, specified in a TO or FOR parameter when the object was defined. This parameter can be specified only if the object is to be permanently exported, that is, deleted from the original system.

PURGE
specifies that the object is to be deleted even if the retention period has not expired.

NOPURGE
specifies that the object is not to be deleted unless the retention period has expired.

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 is:

- 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 removes the user catalog connector entry for D27UCAT1 from the master catalog. 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 Cluster: Example 2

In this example, a key-sequenced cluster, D40.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
//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 -
          D40.EXAMPLE.KSDS1 -
          OUTFILE(RECEIVE) -
          TEMPORARY -
          INHIBITSOURCE
/*
```

The job control statements are:

- 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, D40.EXAMPLE.KSDS1, to a portable file, TAPE2, and modifies the cluster's catalog entries (its cluster, data, and index entries). Access Method Services dynamically allocates D40.EXAMPLE.KSDS1. Since the high-level qualifier of the cluster name is the name of the alias of D27UCAT2, the cluster can be allocated without specifying a JOBCAT or STEPCAT DD statement. The EXPORT command's parameters are:

- D40.EXAMPLE.KSDS1, which identifies the cluster to be exported. Because the cluster is not password protected, no password is provided with the cluster's entryname.
- 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, "Importing 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 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
//RECEIVE  DD     DSNAME=TAPE1,UNIT=(2400,,DEFER),
//     VOL=SER=001147,LABEL=(1,SL),DISP=NEW
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
     EXPORT -
          D50.EXAMPLE.ESDS1/DEPT26M -
          OUTFILE(RECEIVE) -
          PURGE
/*
```

The job control statements are:

- 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, D50.EXAMPLE.ESDS1, to a portable file, TAPE1. The cluster is deleted from the system after it is copied into the portable file. Access Method Services dynamically allocates D50.EXAMPLE.ESDS1. Since the high-level qualifier of the cluster is the name of the alias of the catalog D27UCAT2, the cluster can be dynamically allocated without specifying a JOBCAT or STEPCAT DD statement. The EXPORT command's parameters are:

- D50.EXAMPLE.ESDS1, which identifies the entry-sequenced cluster to be exported, and which specifies the cluster's master password, DEPT26M. When a cluster is exported, its master password is required if it is password protected.
- 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 INHIBITTARGET 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.

EXPORTRA

The format of the EXPORTRA command is:

EXPORTRA	**OUTFILE(***dname***)** **CRA(** **(***dname1* **{ALL[b INFILE(***dname2***)] |** **ENTRIES(** **(***entryname* **[b** *dname3* **])** **[b (***entryname* **...)]) |** **NONE})** **[b (***dname1* **...)])** **MASTERPW(***password***)** **[FORCE | NOFORCE]**

"Appendix F: Command Parameters Summary" contains a summary table for the EXPORTRA command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

EXPORTRA Parameters

OUTFILE(*dname* **)**
identifies the DD statement that describes the data set (usually a movable disk pack or magnetic tape reel) that is to contain a copy of the VSAM objects and catalog entries identified with the CRA parameter.

CRA (
(*dname1*
{ [ALL [b INFILE(*dname2* **)]] \|**
ENTRIES (
(*entryname* **[b** *dname3* **])**
[b (*entryname* **...)]) \|**
NONE }
[b (*dname1* **...)])**

The CRA parameter and its subparameters describe the volume's catalog recovery area, identify the specific entries to be copied from the catalog recovery area, and specify the volumes that contain the entries' data. The subparameters are repeated to describe each catalog recovery area to be referenced, and whether its entries are to be copied.

dname1
identifies the DD statement that describes a volume whose catalog recovery area is to be referenced. When the cluster to be recovered resides on many volumes, you identify each volume's DD statement with the *dname1* subparameter list. Each *dname1* you specify is followed by one of the following subparameters:

ALL
specifies that each entry in *dname1's* catalog recovery area is to be copied to the OUTFILE data set. When part of the object's data (that is, part or all of the object's data component) is on another volume, you use INFILE to describe the other volume(s).

INFILE(*dname2* [b *dname2* ...])
identifies the DD statement that describes all volumes that contain part of one (or more) of the objects in *dname1's* catalog recovery area.

ENTRIES (
(*entryname* [b *dname3*])
[b (*entryname* ...)])
names each object in the catalog recovery area that is to be copied to the OUTFILE data set. All other objects in the catalog recovery area are not copied.

entryname
names an object whose catalog entry is copied in the *dname1* volume's catalog recovery area. You can specify the entryname of the following entry types: alternate index, cluster, and nonVSAM.

When you specify a cluster entry, the cluster's data and index entries and all associated path entries are also copied. In addition, the cluster's contents (that is, its data records, in logical sequential order) are copied, following the copies of the cluster's catalog entries.

When you specify an alternate index entry, the alternate index's entries and contents and all associated path entries are copied as though it were a key-sequenced cluster.

dname3
identifies the DD statement that describes each volume that contains part of the object. *dname3* is ignored when specified for a nonVSAM entry.

dname3 is required when the object is a VSAM cluster or alternate index whose data component resides on volumes other than the volume identified with the *dname1* DD statement.

NONE
specifies that no object in the volume's catalog recovery area is to be recovered. You must specify *dname1* NONE for each DD statement's *dname* that was included in a previous ALL or ENTRIES subparameter, if no entry's catalog information is to be recovered from the volume identified by *dname1*.

For example, YOUR.LARGE.DATASET resides on three direct-access volumes: VOL01, VOL02, and VOL03. When you issue the EXPORTRA command to obtain a copy of the cluster's catalog entries from the catalog recovery area, you code:

```
//STEP1    EXEC PGM=IDCAMS
//VL0A     DD   UNIT=(3330,3),AMP='AMORG',DISP=OLD,
//      VOL=SER=(VOL01,VOL02,VOL03)
//VL01     DD   UNIT=3330,VOL=SER=VOL01,AMP='AMORG',
//      DISP=OLD
//VL02     DD   UNIT=3330,VOL=SER=VOL02,AMP='AMORG',
//      DISP=OLD
//VL03     DD   UNIT=3330,VOL=SER=VOL03,AMP='AMORG',
//      DISP=OLD
//VOLOUT   DD   UNIT=2400,VOL=SER=TAPE44
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     EXPORTRA -
     OUTFILE(VOLOUT) -
     CRA ( -
          (VL01 ENTRIES ( -
               (YOUR.LARGE.DATASET VL0A)))-
          (VL02 NONE) -
          (VL03 NONE) )
/*
```

Each *dname3* DD statement specifies the volume's serial number, its device type, and the data organization (AMP='AMORG').

MASTERPW(*password* **)**
specifies the master catalog's master password. The master catalog's master password is required when the master catalog is password-protected.

FORCE | NOFORCE
specifies whether the catalog entry's copy is to be copied from the catalog recovery area even though the copy itself might be inaccurate.

When you specify FORCE, EXPORTRA ignores time stamp mismatches and the object's catalog-entry copy is copied from the volume's catalog recovery area, even though a loss of data integrity might result. Otherwise, when the catalog recovery area's copy is not accurate, it is not copied and the EXPORTRA command terminates with an error message.

EXPORTRA Example

"IMPORTRA Examples" illustrates the use of EXPORTRA and IMPORTRA together to recover damaged entries of a catalog. Refer to that section for an example of the EXPORTRA command.

IMPORT

The format of the IMPORT command, when it is used to move or connect a user catalog, is:

```
IMPORT   CONNECT
         OBJECTS((usercatname
            VOLUMES(volser)
            DEVICETYPE(devtype)))
         [CATALOG(mastercatname[/password])]
```

The format of the IMPORT command, when it is used to move or restore a cluster or alternate index, is:

```
IMPORT   {INFILE(dname) | INDATASET(entryname)}
         {OUTFILE(dname[/password]) |
            OUTDATASET(entryname[/password])}
         [CATALOG(catname[/password])]
         [ERASE | NOERASE]
         [INTOEMPTY]
         [PURGE | NOPURGE]
         [SAVRAC | NOSAVRAC]
         [OBJECTS((name
               [ƀNEWNAME(newname)]
               [ƀVOLUMES(volser[ƀ volser...])]
               [ƀFILE(dname)]
               [ƀKEYRANGES((lowkey ƀ highkey)
                  [ƀ(lowkey ƀ highkey)....])]
               [ƀORDERED | UNORDERED])
            [ƀ(name...)])]
```

"Appendix F: Command Parameters Summary" contains a table for the IMPORT command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

IMPORT Parameters

IMPORT
specifies that a cluster, alternate index, or user catalog entry is to be moved into the system in which this command is executed.

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.

INFILE(*dname* **)**
specifies the name of a DD statement that identifies the portable copy of the cluster or alternate index to be imported. If a non-labeled tape or a direct-access data set created by DOS/VS Access Method Services contains the copy, the following DCB parameters must be specified on the referenced DD statement: BLKSIZE must be the same value that was specified for BLKSIZE when the object was exported; LRECL must be

the larger of 268 or maximum record size + 4; and RECFM must be VBS.

In addition to the DD statement for INFILE, you must identify the entry's catalog with a JOBCAT or STEPCAT DD statement unless:

- The data set's entry is in the master catalog, or
- The first qualifier of the data set's qualified name identifies a catalog (that is, the first qualifier is the catalog's name or alias).

INDATASET(*entryname* **)**
names the cluster or alternate index to be imported.

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.

OUTFILE(*dname* [/ *password*] **)**
specifies the name of a DD statement that identifies the data set name and volume(s) that is to receive the cluster or alternate index that is to be imported. If the data set was permanently exported and/or you are importing to a volume other than the original volume, the DD statement specifies the name of the cluster or alternate index as DSNAME, the volume serial number(s), the device type, DISP=OLD, and AMP='AMORG'. In addition, you must use concatenated DD statements when both of the following conditions exist and the data set was permanently exported:

- An alternate index or key-sequenced cluster is being imported.
- Its data and index components are on different device types.

The first DD statement specifies the name of the cluster or alternate index as the DSNAME, the volume serial numbers and device type of the data component, DISP=OLD, and AMP='AMORG'. The second DD statement specifies the name of the index component as the DSNAME, the volume serial numbers and device type of its index component, DISP=OLD, 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 object must be owned by a VSAM catalog.

password
is the update (or higher level) password of the output object. You must supply the object's password when it is empty.

When you use a DD statement to describe the output data set, you must identify the data set's catalog with a JOBCAT or STEPCAT DD statement unless:

- The data set's entry is in the master catalog, or
- The first qualifier of the data set's qualified name identifies a catalog (that is, the first qualifier is the catalog's name or alias).

OUTDATASET(*entryname* [/ *password*] **)**
specifies the name of the data set that is to receive the data being imported. When you specify OUTDATASET, the VSAM data set you identify is dynamically allocated.

password
is the update (or higher level) password of the output object. You must supply the object's password when it is empty.

CATALOG(*catname* [/ *password*])
specifies the name of the catalog in which the imported object is to be cataloged. This parameter is required when the catalog is password protected.

catname
is the name of the catalog in which the entry to be imported is to be defined. If you are importing a user catalog, the specified catalog must be the master catalog.

password
specifies the catalog's update or higher level password. When you import an alternate index whose base cluster is password-protected, you should supply the catalog's master password. Otherwise, you (or the operator) will be prompted to supply the base cluster's update password.

ERASE | NOERASE
specifies whether the data component of the cluster or alternate index 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 the object into the system from which it was previously exported with the TEMPORARY option. This parameter overrides whatever was specified when the object was defined or last altered.

INTOEMPTY
specifies that you are importing into an empty data set. An attempt to import into an empty data set without specifying this parameter will fail. Importing into an empty data set causes the data set to be loaded from the portable data set. The password and RACF profiles associated with the empty data set will be retained. When importing into an empty data set, the SAVRAC | NOSAVRAC parameter applies only to the paths imported and successfully defined over the empty data set. If the define of an exported path fails because a catalog entry with the same name already exists, the path on the portable data set is ignored.

PURGE | NOPURGE
specifies whether the original cluster or alternate index 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 the object into the original system from which it was exported with the TEMPORARY option.

SAVRAC | NOSAVRAC
specifies, for a RACF-protected object, whether existing profiles are to be used or whether new profiles are to be created.

SAVRAC should be specified when RACF data set profiles already exist for objects being imported from the portable data set. Typically, you would specify this option when replacing a data set with a portable copy made

with an EXPORT TEMPORARY operation. SAVRAC will cause the existing profiles to be "saved" and used, rather than letting the system delete old profiles and create new, default profiles.

Caution: You should ensure that valid profiles do exist for all components (for example, cluster, data, index) being imported when SAVRAC is specified. If this is not done, an invalid and possibly improper profile may be "saved" and used inappropriately. Remember, that paths are imported along with their corresponding cluster or alternate index, and the same caution applies to these entries. In particular, keep in mind that additional paths may be brought in during the import.

NOSAVRAC should be specified when you wish new profiles to be created. This is usually the situation when importing a permanently exported cluster or alternate index. A profile will be defined for the imported components if either the Automatic Data Set Protection option has been specified for you or if the exported component had a RACF indication in the catalog when it was exported. If you import into a catalog in which there is a component with a duplicate name which is marked as having been temporarily exported, it and any associated profiles will be deleted before importing the portable data set.

OBJECTS

specifies the attributes for the cluster, alternate index, or user catalog to be imported. Attributes may be specified for the components of a alternate index or cluster by repeating the parameter list. Attributes cannot be specified separately for the components of a user catalog. You can specify OBJECTS

- For a user catalog:

  OBJECTS((*name*
  VOLUMES(*volser*)
  DEVICETYPE(*devtype*)))

- For a cluster or alternate index:

OBJECTS((*name* [ƀ**NEWNAME(** *newname* **)**]
[ƀ**VOLUMES(** *volser* [ƀ *volser*]**)**]
[ƀ**FILE(** *dname* **)**]
[ƀ**KEYRANGES((** *lowkey* ƀ *highkey* **)**
[ƀ**(** *lowkey* ƀ *highkey* **)** ...]**)**]
[ƀ**ORDERED | UNORDERED**] **)**
[ƀ**(** *name* ...**)**]**)**

name
specifies the name of the data component, index component, cluster, alternate index, or user catalog whose attributes are being specified.

NEWNAME(*newname* **)**
specifies the new name of an imported cluster or alternate index or its components. 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 a cluster or alternate index that was exported with the TEMPORARY option and it is being imported back into the original system, you must rename it and each of its components.

VOLUMES(*volser* [ƀ *volser* ...]**)**
specifies the volumes on which the cluster or alternate index is to reside or the volume on which the user catalog resides.

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, the original volume is the receiving volume. This parameter is required when a user catalog is to be imported; when importing a user catalog, specify only one volume.

Caution: Portable data sets created by previous releases of OS/VS2 might not contain volume information. In this case, the VOLUMES parameter is required.

FILE(*dname* **)**
specifies the name of a DD statement that identifies the volumes allocated to the data and index components of a key-sequenced cluster or of an alternate index. 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. The FILE parameter is also used to specify the name of a DD statement that identifies the prime catalog recovery volume of an alternate index or of a path over an alternate index.

KEYRANGES((*lowkey* ƀ *highkey*) [ƀ (*lowkey* ƀ *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 an alternate index or cluster or its data 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 must be in ascending order, and 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 and be enclosed in single quotation marks.

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.

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.

By repeating the OBJECTS parameter set for each component and including VOLUMES in each parameter set, you can have the data and index components on 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 or alternate index, the job may be terminated because of allocation problems. Each space allocation quantity is recorded in a catalog

entry as an amount of cylinders or tracks even when RECORDS was specified in the DEFINE command. When a cluster or alternate index is imported, the number of cylinders or tracks in the catalog entry is not modified, even though the object may be imported to reside on a device type other than that it was exported from. An attempt to import an object that previously resided on a 3330 may fail if it is imported to a 2314. Conversely, if an object is exported from a 2314 and imported to a 3330, more space is allocated than the object needs.

You can avoid space allocation problems when you define an empty cluster or alternate index on the new system or catalog and identify it as the target cluster for the object being imported. An alternative way to avoid space allocation problems is to follow this procedure:

- Using the REPRO command, copy the cluster or alternate index to be exported to a magnetic tape. The copy is called a portable data set.
- Using the DEFINE command, define a new entry for the cluster or alternate index in the catalog to which it is to be moved. Specify all the parameters used when the object 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 in the catalog for the object, you must delete that entry or use a different name in the DEFINE command.
- Using the REPRO command, load the portable data set into the newly defined object (copy it from the tape).

IMPORT Examples

Importing 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 is:

- 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 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, which 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.

Importing a Key-Sequenced Cluster: Example 2

In this example a key-sequenced cluster, D40.EXAMPLE.KSDS1, that was previously exported, is imported. (See the previous EXPORT example, "Exporting a Key-Sequenced Cluster.") OUTFILE and its associated DD statement are provided to allocate the data set.

The original copy of D40.EXAMPLE.KSDS1 is replaced with the imported copy, TAPE2. Access Method Services finds and deletes the duplicate name, D40.EXAMPLE.KSDS1, in the catalog D27UCAT2. (A duplicate name exists because TEMPORARY was specified when the cluster was exported.) Access Method Services then redefines D40.EXAMPLE.KSDS1 using the catalog information from the portable file, TAPE2.

```
//IMPORT2  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//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(D40.EXAMPLE.KSDS1) -
          CATALOG(D27UCAT2/MRPWD27)
/*
```

The job control statements are:

- 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 D40.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 gives the name of the data set being imported. Since the high-level qualifier of the data set is the name of the alias of the user catalog D27UCAT2, Access Method Services can dynamically allocate the cluster without specifying a JOBCAT or STEPCAT DD statement.
- 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.

Importing an Entry-Sequenced VSAM Cluster: Example 3

In this example, an entry-sequenced cluster, D50.EXAMPLE.ESDS1, is imported from a portable file, TAPE1. This example is associated with EXPORT Example 3, "Exporting an Entry-Sequenced Cluster."

```
//IMPORT3  JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SOURCE   DD   DSNAME=TAPE1,UNIT=(2400,,DEFER),DISP=OLD,
//         VOL=SER=001147,LABEL=(1,SL)
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(D50.EXAMPLE.ESDS2) -
          OBJECTS( -
                (D50.EXAMPLE.ESDS1 -
                   NEWNAME(D50.EXAMPLE.ESDS2) -
                   VOLUMES(VSER03) ) -
                 ) -
          CATALOG(D27UCAT2/MRCATPW1)
/*
```

The job control statements are:

- 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.
- 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.
- OUTDATASET gives the name of the data set that is being imported—the name is the renamed cluster. Since the high-level qualifier of the data set name is the name of the alias of the catalog, D27UCAT2, the data set can be dynamically allocated without specifying a JOBCAT or STEPCAT DD statement.

- OBJECTS, which specifies some of the attributes for the object being imported:
  - D50.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 D50.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.

IMPORTRA

The format of the IMPORTRA command is:

IMPORTRA	{**INFILE(***dname***)** | **INDATASET(***entryname***)**} [**OUTFILE(***dname***)**] [**SAVRAC** | **NOSAVRAC**] [**OBJECTS(** (*entryname* [**FILE(***dname***)**] [**VOLUMES(***volser* [ъ *volser*...])] [**DEVICETYPE(***devtype***)**]) [(*entryname*...)...])] [**CATALOG(***catname* [/*password*])]

"Appendix F: Command Parameters Summary" contains a summary table for the IMPORTRA command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

IMPORTRA Parameters

INFILE(*dname* **)**
names the DD statement that describes the portable data set (that is, the data set that resulted when you issued the EXPORTRA command).

INDATASET(*entryname* **)**
specifies the data set name of the portable data set. When you specify INDATASET instead of INFILE, the portable data set (that is, the data set that resulted when you issued the EXPORTRA command) is dynamically allocated.

OUTFILE(*dname* **)**
names the DD statement that contains a data set name and the volume serial number of each volume that is to contain the imported data sets. You must use concatenated DD statements if the data sets are on different device types. If the OUTFILE parameter is not supplied, the required data sets are dynamically allocated as needed.

The dsname specified with the DD statement cannot be one of the names cataloged in the target catalog (that is, the catalog that is to contain the imported catalog entries) and it cannot be one of the entrynames within the portable data set. The dsname you specify is used by IMPORTRA processing and has no importance to VSAM or Access Method Services after your job completes. The DD statement(s) also specify the volume serial number(s), device type, DISP=OLD and AMP='AMORG'.

SAVRAC | NOSAVRAC
specifies, for a RACF-protected object, whether existing profiles are to be used or whether new profiles are to be created.

SAVRAC should be specified when RACF data set profiles already exist for objects being imported from the portable data set. SAVRAC will cause the existing profiles to be "saved" and used, rather than letting the system delete old profiles and create new, default profiles.

Caution: You should ensure that valid profiles do exist for all components (for example, cluster, data, index) being imported when SAVRAC is specified. If this is not done, an invalid and possibly improper profile may be "saved" and used inappropriately. Remember, that paths are imported along with their corresponding cluster or alternate index, and the same caution applies to these entries. In particular, keep in mind that additional paths may be brought in during the import.

NOSAVRAC should be specified when you wish new profiles to be created. A profile will be defined for the imported components if either the RACF Automatic Data Set Protection option has been specified for you or if the exported component had a RACF indication in the catalog at the time it was exported. The mode, SAVRAC or NOSAVRAC, set by these keywords applies to all VSAM components being imported back into the system.

There may be situations where the SAVRAC or NOSAVRAC mode would not be suitable for all entities restored from the catalog recovery area. In such cases, the following procedure is suggested:

- Issue two separate EXPORTRA commands, resulting in two portable data sets. One EXPORTRA will create a portable data set containing entities to which SAVRAC should apply, the other to which NOSAVRAC should apply.
- Issue two IMPORTRA commands, one specifying SAVRAC and the other NOSAVRAC.

OBJECTS(
(*entryname*
[FILE(*dname* **)]**
[VOLUMES(*volser* [b *volser* ...] **)]**
[DEVICETYPE(*devtype* **)])**
(*entryname* ... **)** ... **)**

The OBJECTS parameter group specifies attributes for one of the objects (that is, a VSAM data set, alternate index, user catalog, or nonVSAM data set) on the portable file.

entryname
specifies the object's entryname. You can specify the entryname and associated attributes for up to 255 objects.

You can specify the entryname of these types of objects only: VSAM data set, nonVSAM data set, alternate index, path, user catalog, and a VSAM object's data or index component.

FILE(*dname* **)**
names the DD statement that describes the volume(s) onto which a unique object is to be imported (that is, an object that doesn't share the data space it occupies with other VSAM objects). For a unique alternate index or key sequenced cluster, you must code FILE twice within the OBJECTS parameter, once for the data component and once for the index component. When you specify FILE and its DD statement, the entryname you specify must be the object's data or index component's entryname. The FILE parameter is also used to describe the prime catalog recovery area volume for a path. It may be specified, in this case, when the entryname identifies a path.

VOLUMES(*volser* [ƀ *volser* ...] **)**
specifies the object's volume(s).

volser
is a 1 to 6 alphanumeric or national character volume serial number.

When the *volser* contains a special character (that is, a comma [,], semicolon [;], blank [ƀ], period [.], single quote ['], ampersand [&], plus sign [+], equal sign [=], hyphen [-], parenthesis [(or)], slash [/], or asterisk [*]), enclose the *volser* in single-quotation marks (for example, VOLUMES('*DORIS')).

When the *volser* contains a special character and also contains a single-quotation mark, code the imbedded quotation mark as two single-quotation marks (for example, VOLUMES('*CA''RO')).

You can code *volser* in hexadecimal form, where two hexadecimal characters represent one EBCDIC character. For example, VOLUMES(X'E2E4C5') is the same as VOLUMES(SUE).

DEVICETYPE(*devtype* **)**
specifies the device type of a user catalog or nonVSAM data set being imported.

This parameter can be specified only when the entryname names a nonVSAM data set or user catalog.

CATALOG(*catname* [/ *password*] **)**
identifies the target catalog (that is, the catalog that is to contain the imported catalog entries).

If you don't include the CATALOG parameter, the JOBCAT or STEPCAT catalogs are used. If you haven't specified a JOBCAT or STEPCAT catalog, the VSAM master catalog is used. You must specify the CATALOG parameter when the target catalog is password protected. Otherwise, the system operator is prompted to supply the correct password.

catname
names the catalog. When you specify a user catalog, you must describe and allocate the catalog with a JOBCAT or STEPCAT DD statement.

password
is the target catalog's master password, if the catalog is password protected.

IMPORTRA Example

This example shows how EXPORTRA and IMPORTRA can be used to recover a VSAM catalog. The first part illustrates the use of EXPORTRA. The second part shows how IMPORTRA is used.

Recovering a VSAM Catalog: Example 1—Part 1 (EXPORTRA)

This example performs the EXPORTRA function against the VSAM master catalog, AMASTCAT, and against all of the volumes owned by it. All of the data sets listed in the catalog recovery areas on the catalog volume and on the other volumes owned by AMASTCAT are exported to a SAM data set on another disk volume. Using the FORCE option causes EXPORTRA to ignore time stamp mismatches between the volumes and the catalog.

```
//RECOVER  JOB    ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//CRAVOL1  DD   DISP=OLD,VOL=SER=VSER01,UNIT=2314,
//      AMP='AMORG'
//CRAVOL2  DD   DISP=OLD,VOL=SER=VSER04,UNIT=2314,
//      AMP='AMORG'
//PORT     DD   DSNAME=PORT,DISP=(NEW,KEEP),
//      SPACE=(CYL,(3,3)),VOL=SER=231401,UNIT=2314
//SYSIN    DD   *

     EXPORTRA -
          CRA( -
            (CRAVOL1 ALL) -
            (CRAVOL2 ALL) -
             ) -
          FORCE -
          OUTFILE(PORT) -
          MASTERPW(MCATMRPW)
/*
```

The job control statements are:

- CRAVOL1 DD, which identifies and allocates the first volume whose catalog recovery area contents is to be exported.
- CRAVOL2 DD, which identifies and allocates the second volume whose catalog recovery area contents is to be exported.

- PORT DD, which identifies the sequential file that is to receive the exported data sets and catalog recovery records.

The EXPORTRA command exports everything appearing in the catalog recovery areas of volumes VSER01 and VSER04. The EXPORTRA command's parameters are:

- CRA, which is required and identifies the catalog recovery areas and volumes from which the export is to take place. The *dnames* of the DD statements for these objects must be identical to the names specified for this parameter. The ALL subparameter specifies that everything is to be exported from each catalog recovery area.
- FORCE, which specifies that timestamp mismatches are to be ignored.
- OUTFILE, which is required and identifies the sequential (SAM) nonVSAM data set that is to receive the exported information. The *dname* of the DD statement for this object must be identical to the name specified with this parameter.
- MASTERPW, which specifies the master password of the master catalog. This parameter is required in order to export information from catalog recovery areas controlled by a recoverable catalog.

Recovering a VSAM Catalog: Example 1—Part 2 (IMPORTRA)

This example imports all of the data sets that were exported using EXPORTRA in the previous example. The receiving catalog is the VSAM master catalog, and the CATALOG parameter is used to supply its master password.

```
//RESTORE   JOB    ...
//STEP1     EXEC   PGM=IDCAMS
//SYSPRINT  DD     SYSOUT=A
//PORT      DD     DSNAME=PORT,DISP=OLD,UNIT=2314,
//      VOL=SER=231401
//VSAMIN    DD     DSNAME=DUMMY.NAME,DISP=OLD,UNIT=2314,
//      VOL=SER=(VSER01,VSER04),AMP='AMORG'
//SYSIN     DD     *
     IMPORTRA -
          INFILE(PORT) -
          OUTFILE(VSAMIN) -
          CATALOG(AMASTCAT/MCATMRPW)
/*
```

The job control statements are:

- PORT DD, which identifies and allocates the portable data set created previously by EXPORTRA.
- VSAMIN DD, which identifies and allocates the volumes of the objects to be imported. This permits them to be reloaded from the portable data set. The *dsname* appearing on the DD statement is required. It is a dummy name that must not appear either in the catalog or among the names on the portable data set.

The IMPORTRA command imports the data sets written on the portable data set by EXPORTRA previously. The IMPORTRA command's parameters are:

- INFILE, which is required and which identifies the portable data set.
- OUTFILE, which identifies each VSAM volume involved in the import. The DD statement identified by OUTFILE also identifies the dummy data set, DUMMY.NAME.
- CATALOG, which specifies the name of the master catalog. The master password of the target catalog is required in order to import catalog information.

LISTCAT

The format of the LISTCAT command is:

LISTCAT	**[CATALOG(***catname* **[/** *password* **])]** **[OUTFILE(***dname* **)]** **[ENTRIES(***entryname* **[/** *password* **]** **[ъ** *entryname* **[/** *password* **]...]) |** **LEVEL(** *level* **)]** **[CLUSTER][ъDATA][ъINDEX][ъUSERCATALOG]** **[ъSPACE][ъNONVSAM][ALTERNATEINDEX]** **[ъPATH][ъPAGESPACE][ъALIAS]** **[ъGENERATIONDATAGROUP]** **[ALL | NAME | VOLUME | ALLOCATION |** **HISTORY]** **[CREATION(***days* **)]** **[EXPIRATION(***days* **)]** **[NOTUSABLE]**

"Appendix F: Command Parameters Summary" contains a table for the LISTCAT command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

LISTCAT Parameters

LISTCAT
specifies that catalog entries are to be listed.

Note to TSO users: When LISTCAT is invoked from a TSO terminal and no operands are specified, the prefix of the TSO user becomes the highest level of entryname qualification and only those entries with a matching highest level of qualification are listed. It is as if you specified: **LISTCAT LEVEL(** *TSO-user-prefix* **)**

CATALOG(*catname* **[/** *password* **])**
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" for information about the order in which catalogs are searched.

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 are password-protected, 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.

If the catalog's volume is physically mounted, it is dynamically allocated. The volume must be mounted as permanently resident or reserved.

OUTFILE(*dname* **)**

specifies a data set, other than the SYSPRINT data set, to receive the output produced by LISTCAT (that is, the listed catalog entries). Output and completion messages produced by Access Method Services are sent, along with your job's JCL and input statements, to the SYSPRINT data set.

dname identifies a DD statement that describes 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."

ENTRIES(*entryname* [**/** *password*][ъ *entryname* [**/** *password*] ...]) |
LEVEL(*level* **)**

specifies the names of entries to be listed. The entire catalog is listed when all of the following are true:

- ENTRIES, LEVEL, NOTUSABLE, CREATION, and EXPIRATION are not coded.
- No entrytype parameter is coded.
- TSO is not being used to issue the LISTCAT command.

ENTRIES(*entryname* [**/** *password*]
[ъ *entryname* [**/** *password*] ...] **)**

specifies the name or generic name of each entry to be listed. When you want to list the entries that describe a user catalog, the catalog's volume must be physically mounted. You then specify the catalog's name as the entryname. 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 no other entrytype parameters.

Note to TSO users: When you specify an incomplete qualified entryname, TSO prompts you to complete it.

password

specifies a password when the entry to be listed is password-protected and a password is not specified with the CATALOG parameter. The password must be any of the entry's passwords. The entry's protection attributes are listed only when you specify the entry's (or its catalog's) master password. When you don't supply a password for a password-protected entry, the operator or TSO terminal user is prompted for the entry's password. You cannot supply a password for these types of entries: nonVSAM data set, generation data group, alias, user-catalog connector, and data space.

LEVEL(*level* **)**

specifies that all entries that match the level of qualification specified by (*level*) are to be listed irrespective of the number of qualifications after the level. If a generic level name is specified, one qualifier replaces the *.

Note: the * may not specify the last level of qualifications.

As an example, suppose the VSAM catalog contains the following names:

1. A.A.B
2. A.B.B
3. A.B.B.C
4. A.B.B.C.C
5. A.C.C
6. A.D
7. A.E

If ENTRIES(A.*) is specified, then entries 6 and 7 would be listed. If ENTRIES(A.*.B) is specified, then entries 1 and 2 would be listed. If LEVEL(A.*.B) is specified, then entries 1,2,3 and 4 would be listed. If LEVEL(A) is specified, then entries 1,2,3,4,5,6 and 7 would be listed.

Note to TSO users: TSO will prefix the *userid* to the specified data set name when the ENTRIES parameter is specified with an unqualified entry name. The *userid* is *not* prefixed when the LEVEL parameter is specified.

Entry-Types

You can specify that certain types of entries are to be listed. When you do, only those entries whose type is specified is listed (that is, when you specify CLUSTER but not DATA or INDEX, the cluster's entry is listed and its associated data and index entries are not listed.) When you identify an entry with its name (that is, when you specify ENTRIES) and also specify an entry type, the named entry is not listed unless it is of the specified type. You can specify as many entry types as desired. When you want to completely list a catalog, do not specify any entry type.

CLUSTER
specifies that cluster entries are to be listed. If CLUSTER is specified and DATA and INDEX are not also specified, entries for the cluster's data and index components are not listed.

DATA
specifies that entries for data components of clusters and alternate indexes. are to be listed. If a VSAM object's name is specified and DATA is coded, only the object's data-component entry is listed. When DATA is the only entrytype parameter coded, the catalog's data component isn't listed.

INDEX
specifies that entries for index components of key-sequenced clusters and alternate indexes are to be listed. If a VSAM object's name is specified and INDEX is coded, only the object's index-component entry is listed. When INDEX is the only entrytype parameter coded, the catalog's index component isn't listed.

USERCATALOG
specifies that catalog connectors are to be listed. The user-catalog connector entries are in the master catalog. (User-catalog connector entries can also be in a user catalog, but the OS/VS2 system doesn't recognize them when searching for a user catalog.)

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, SPACE can be coded only when no other entrytype parameter is coded.

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.

ALIAS
specifies that alias entries are to be listed.

GENERATIONDATAGROUP
specifies that entries for generation data groups are to be listed.

PAGESPACE
specifies that entries for page spaces are to be listed.

ALTERNATEINDEX
specifies that entries for alternate indexes are listed. If ALTERNATEINDEX is specified and DATA and INDEX are not also specified, entries for the alternate index's data and index components aren't listed.

PATH
specifies that entries for paths are listed.

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.

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; zero 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.

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 and it indicates that all entries are to be listed. Any value that exceeds the year 2000 will default to 99.999(yyddd). Zero 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. "Appendix B: Interpreting LISTCAT Output Listings" shows the listed information that results when you specify nothing (which defaults to NAME), ALL, VOLUME, ALLOCATION, and HISTORY.

ALL
specifies that all fields are to be listed.

NAME
specifies that the name and entry type of the entries are to be listed.

Note to TSO users: Only the name of each entry associated with the TSO user's prefix is listed when no other parameters are coded.

Some entrytypes are listed along with their associated entries. The entrytype and name of the associated entry follows the listed entry's name. For details, see "ASN: Associations Group" in "Appendix B: Interpreting LISTCAT Output Listings."

HISTORY
specifies that only the following information is to be listed for each entry: name, entry type, ownerid, creation date, expiration date, and for a recoverable catalog's entries, the catalog recovery area's volume, device type, and control interval number. It can be specified for CLUSTER, DATA, INDEX, ALTERNATEINDEX, PATH, GENERATIONDATAGROUP, PAGESPACE, and NONVSAM.

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 only listed for data and index component entries, data space (volume) entries, nonVSAM data set entries, and user catalog connector entries.

Note to TSO users: Only the name and volume serial numbers associated with the TSO user's prefix are listed when no other parameters are coded.

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.

NOTUSABLE
specifies that only those data and index entries which have the "unusable" indicator on are to be listed. A data or index component is marked "unusable" when a system failure occurs that results in damage to the entry's cataloged information.

When the cataloged information is reset, the damaged entry and its backup copy (in the catalog recovery area) might not match when the space allocation information is compared. VSAM marks the catalog entry "unusable" until the space allocation information is corrected. See "Restoring Catalog Entries After System Failure" for more details.

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(D40.EXAMPLE.KSDS1) -
          CLUSTER -
          ALL
/*
```

The LISTCAT command lists the cluster's catalog entry. It is assumed that the high level of the qualified cluster name is the same as the alias of the catalog D27UCAT2; this naming convention directs the catalog search to the appropriate catalog. 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.

Alter a Catalog Entry, Then List the Modified Entry: Example 2

In this example, the freespace attributes for the data component (KSDATA) 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, which 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.

LISTCRA

The format of the LISTCRA command is:

LISTCRA	**INFILE(***dname* [ъ *dname* ...]) **MASTERPW(***password* **)** **[OUTFILE(***dname* **)]** **[COMPARE | NOCOMPARE]** **[DUMP | NAME | SEQUENTIALDUMP]** **[CATALOG(***catname* [*/ password*] ъ *dname* **)]**

"Appendix F: Command Parameters Summary" contains a summary table for the LISTCRA command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

LISTCRA Parameters

INFILE(*dname* [ъ *dname* ...] **)**
identifies the DD statement(s) that describes the catalog recovery area's volume. You can list or compare more than one catalog recovery area. However, if COMPARE is specified, all volumes whose catalog recovery areas are listed must be owned by the same VSAM catalog.

MASTERPW(*password* **)**
specifies the master catalog's master password. You must supply the master catalog's master password with the MASTERPW parameter when the master catalog is password-protected.

OUTFILE(*dname* **)**
identifies the DD statement that describes an alternate output data set. See "Output Data Sets" for more details about alternate output data sets. When OUTFILE is not specified, the listing is printed on the output device described with the SYSPRINT DD statement.

COMPARE | NOCOMPARE
specifies whether the list is to be limited to those entries in the catalog recovery area which do not match their corresponding catalog entries.

When you specify COMPARE, only those entries which do not match are listed. All of the catalog's entries are compared, except the catalog's self-describing records. You must identify the catalog (that contains the to-be-compared catalog entries) with the CATALOG parameter.

When you specify NOCOMPARE (or allow it to default), all of the catalog recovery area's records are listed.

The COMPARE parameter cannot be specified if you specify SEQUENTIALDUMP.

Note: If you specify the COMPARE option, page space entries will always result in a mismatch. This is a normal condition and does not require any corrective action. The mismatch is due to the OPEN indicator not being set in the page space record in the CRA, whereas it *is* set in the catalog.

See "Appendix D: Interpreting LISTCRA Output Listings" for examples of each type of output listing.

DUMP | NAME | SEQUENTIALDUMP
specifies the amount of cataloged information to be listed.

DUMP
specifies that each listed entry is printed in its entirety in both hexadecimal and character form, sorted alphanumerically and grouped.

NAME
specifies that each listed entry includes only the entry's name, its volume serial numbers, and the name and entry type of each associated entry. All listed entries are sorted alphanumerically and grouped.

SEQUENTIALDUMP
specifies that each listed entry is to be printed in its entirety in both hexadecimal and character form. Records are listed in the sequence each appears in the catalog recovery area.

The DUMP, NAME, SEQUENTIALDUMP, COMPARE, and NOCOMPARE options can be specified to produce five different kinds of output listing. See "Appendix D: Interpreting LISTCRA Output Listings" for examples of each type of listing.

CATALOG(*catname* [/ *password*] ъ *dname* **)**
identifies the catalog that owns the volume(s) identified with the INFILE parameter. The CATALOG parameter is required when you specify the COMPARE option. If the catalog being compared is not the VSAM system catalog, a JOBCAT or STEPCAT DD statement is required.

password
When the catalog is password-protected, you must supply the catalog's master password. The password is required even though the master catalog's password may have been supplied in the MASTERPW parameter.

dname
When CATALOG is specified, you must describe and allocate the catalog to be compared with a DD statement. *dname* identifies the catalog's DD statement, and is required when you specify the CATALOG parameter. You cannot specify JOBCAT or STEPCAT as the *dname.*

If the catalog information is entered incorrectly, the NOCOMPARE default is taken, and all of the CRA records are listed.

LISTCRA Example

Listing a Catalog Recovery Area: Example 1

This example lists the catalog recovery areas for those volumes owned by the VSAM master catalog, AMASTCAT, in dump format, and compares those catalog recovery areas with the actual catalog records themselves.

```
//LISTAREA  JOB    ...
//STEP1     EXEC   PGM=IDCAMS
//SYSPRINT  DD     SYSOUT=A
//CRAVOL1   DD     DISP=OLD,VOL=SER=SG2001,UNIT=2314
//CRAVOL2   DD     DISP=OLD,VOL=SER=VSER04,UNIT=2314
//CATVOL    DD     DSNAME=AMASTCAT,DISP=OLD
//SYSIN     DD     *
     LISTCRA -
          INFILE( -
            CRAVOL1 -
            CRAVOL2 ) -
          MASTERPW(MCATMRPW) -
          COMPARE -
          DUMP -
          CATALOG(AMASTCAT/MCATMRPW CATVOL)
/*
```

The job control statements are:

- CRAVOL1 DD, which identifies and allocates the volume that contains the first catalog recovery area to be listed.
- CRAVOL2 DD, which identifies and allocates the volume that contains the second catalog recovery area to be listed.
- CATVOL DD, which identifies and allocates the catalog to be compared against, and makes the catalog available to LISTCRA as a data set.

The LISTCRA command causes the catalog recovery areas on volumes SG2001 and VSER04 to be listed and their contents compared with the VSAM system catalog AMASTCAT. The LISTCRA command's paraemters are:

- INFILE, which is required and specifies the catalog recovery areas to be listed by identifying the DD statement that describes each CRA's volume.
- MASTERPW, which specifies the master password of the master catalog. This parameter is required in order to open the catalog recovery areas for LISTCRA processing.
- COMPARE, which specifies that the catalog records are to be compared with their copies in the catalog recovery areas.
- DUMP, which specifies that the results of the record comparison (that is, the LISTCRA output listing) is to be printed in dump format.
- CATALOG, which is required (because the COMPARE option was specified) and identifies the catalog to be compared against by identifying the DD statement that describes and allocates the catalog as a data set.

PRINT

The format of the PRINT command is:

PRINT	{**INFILE(***dname* [*/ password*]) | **INDATASET(***entryname* [*/ password*])} [**OUTFILE(***dname*)] [**FROMKEY(***key*) | **FROMADDRESS(***address*) | **FROMNUMBER(***number*) | **SKIP(***count*)] [**TOKEY(***key*) | **TOADDRESS(***address*) | **TONUMBER(***number*) | **COUNT(***count*)] [**HEX** | **CHARACTER** | **DUMP**]

"Appendix F: Command Parameters Summary" contains a table for the PRINT command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

PRINT Parameters

INFILE(*dname* [*/ password*]) |
INDATASET(*entryname* [*/ password*])
identifies the data set or component to be printed.

INFILE(*dname* [*/ password*])
specifies the name of the DD statement that identifies the data set or component. You can list a base cluster in alternate-key sequence by specifying a path name as the data set name in the DD statement.

INDATASET(*entryname* [*/ password*])
specifies the name of the entry to be printed. If INDATASET is specified, the entryname is dynamically allocated.

password
If a VSAM data set or component is password protected, a password must be supplied. The password to be supplied is the master 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 password of the cluster if you are listing a component of a password-protected cluster. Passwords are applicable only to VSAM data sets and their components.

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 the "Introduction" under "Output Data Sets."

FROMKEY(*key* **) | FROMADDRESS(** *address* **) |**
FROMNUMBER(*number* **) | 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(*key* **)**
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 an alternate index, a key-sequenced VSAM data set, or an indexed-sequential (ISAM) nonVSAM data set is being printed. When you specify FROMKEY, you cannot also specify TOADDRESS to identify the last record to be printed.

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.

FROMADDRESS(*address* **)**
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 physical sequential order instead of in logical sequential order. FROMADDRESS can be specified only for VSAM key-sequenced or entry-sequenced data sets or components. FROMADDRESS cannot be specified when the data set is accessed through a path.

FROMNUMBER(*number* **)**
specifies the relative-record number of the first record you want printed.

SKIP(*count* **)**
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). SKIP should not be specified when you are accessing the data set through a path; the results are unpredictable.

address, number, and *count* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n'); the expression cannot be longer than one fullword (eight decimal or hexadecimal numbers, or 32 binary numbers).

TOKEY(*key* **) | TOADDRESS(** *address* **) |**
TONUMBER(*number* **) | 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.

The ending delimiter must be consistent with the starting delimiter. For example, if FROMADDRESS is specified for the starting location, neither

TOKEY nor TONUMBER may be specified for the location. Similarly, if FROMNUMBER is specified for the starting location, neither TOKEY nor TOADDRESS may be specified for the ending location.

TOKEY(*key* **)**
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 an alternate index, a key-sequenced VSAM data set, or an indexed-sequential (ISAM) nonVSAM data set is being printed.

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(*address* **)**
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 physical sequential order instead of in logical sequential order. TOADDRESS can be specified only for a VSAM data set or its data component. TOADDRESS cannot be specified when the data set is accessed through a path.

TONUMBER(*number* **)**
specifies the relative-record number of the last record you want printed.

COUNT(*count* **)**
specifies the number of logical records to be listed. COUNT should not be specified when you are accessing the data set through a path; the results are unpredictable.

address, number, and *count* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n'); the expression cannot be longer than one fullword (eight decimal or hexadecimal numbers, or 32 binary numbers).

HEX | CHARACTER | DUMP
specifies the format of the listing.

HEX
specifies that each byte in the logical record is to be printed as two hexadecimal digits. Key fields are listed in hexadecimal format (see Figure 10).

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 (see Figure 11).

```
IDCAMS  SYSTEM SERVICES                                    TIME:  19:26:13     03/08/75  PAGE  4

LISTING OF DATA SET -EXAMPLE.LISTC.DATA

RBA OF RECORD - 0
C1C2C3C4F0F0F0F0F0F0F0F0F0F0F1C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8
C9D1D2D3D4D5D6D7D8D9E2D3D4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5
F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3
E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8
C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5
F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F-F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3
E4E5E6E7E8E9F0F1F2F3F4F5f6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E300000000
```

Figure 10. Hex Format

```
IDCAMS  SYSTEM SERVICES                          TIME:  19:26:13     03/08/75  PAGE  6

LISTING OF DATA SET -EXAMPLE.LISTC.DATA

RBA OF RECORD - 0
ABCD000000000001ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ012345
6789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGH
IJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWZYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRST
UVWXYZ0123456789ABCDEFGHIJKLMNOPQRST....
```

Figure 11. Character Format

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 (see Figure 12).

PRINT Examples

Print a Key-Sequenced Cluster's Data Records: Example 1

In this example, the data records of a key-sequenced cluster, D40.EXAMPLE.KSDS1, are printed in dump format. That is, each character of the record is printed in its hexadecimal and alphanumeric forms.

```
//PRINT1   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     PRINT -
          INDATASET(D40.EXAMPLE.KSDS1)
/*
```

```
IDCAMS  SYSTEM SERVICES                          TIME:  19:25:14     03/08/75  PAGE  2

LISTING OF DATA SET -EXAMPLE.LISTC

KEY OF RECORD - F0F0F0F0F0F0F0F0F0F0F1
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....                *
```

Figure 12. Dump Format

The job control statement is:

- 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, D40.EXAMPLE.KSDS1. Its parameter is:

- INDATASET, which names the data set to be printed. Since the data set is cataloged in D27UCAT1 and the high-level qualifier of the data set is the name of the alias of D27UCAT1, Access Method Services can dynamically allocate the cluster without the need of JOBCAT or STEPCAT DD statements.

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, D50.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=D50.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 ) -
          CHARACTER
/*
```

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, D50.EXAMPLE.ESDS1, that the records are copied into.

  Note: If the AMP=(BUFND=n) parameter was specified, performance would improve when the data set's records are accessed. BUFND 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, D50.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 will search for the catalog entry describing the nonVSAM data set in either D27UCAT1 or D27UCAT2 user catalog, or in the master catalog.
- OUTFILE, which points to the VSDSET2 DD statement. The VSDSET2 DD statement identifies the output data set, D50.EXAMPLE.ESDS1.

VSAM assumes that D50.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 the SKIP parameter was not specified, Access Method Services assumes that the first fifteen records are to be copied. FROMNUMBER, FROMADDRESS, and FROMKEY cannot be specified for a nonVSAM data set. 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, D50.EXAMPLE.ESDS1. Its parameters are:

- INFILE, which points to the VSDSET2 DD statement. The VSDSET2 DD statement identifies the cluster, D50.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. If you want two or more Access Method Services commands to execute only when the IF statement is satisfied, enclose the commands in a DO...END command sequence.

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 non VSAM data set EXAMPLE.NONVSAM. VSAM will search for the catalog entry in either D27UCAT1 or D27UCAT2 user catalog or in the master catalog.
- COUNT, which specifies that fifteen records are to be printed. Because SKIP was not specified, Access Method Services assumes that the first fifteen records are to be printed. FROMNUMBER, FROMADDRESS, or FROMKEY cannot be specified for a nonVSAM sequential data set.
- CHARACTER, which specifies that each record is to be printed as a group of alphanumeric characters. An example of the printed record is shown in Figure 15.

```
RBA OF RECORD - 0
C2D6E8C5D940404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040404040
40404040404040404040404040404040404040404040
```

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

REPRO

The format of the REPRO command is:

REPRO	{**INFILE(***dname* [*/ password*] [b**ENVIRONMENT(DUMMY)**]) | **INDATASET(***entryname* [*/ password*] [b**ENVIRONMENT(DUMMY)**])} {**OUTFILE(***dname* [*/ password*]) | **OUTDATASET(***entryname* [*/ password*])} [**REPLACE** | **NOREPLACE**] [**REUSE** | **NOREUSE**] [**FROMKEY(***key*) | **FROMADDRESS(***address*) | **FROMNUMBER(***number*) | **SKIP(***count*)] [**TOKEY(***key*) | **TOADDRESS(***address*) | **TONUMBER(***number*) | **COUNT(***count*)]

"Appendix F: Command Parameters Summary" contains a summary table for the REPRO command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

REPRO Parameters

INFILE(*dname* [*/ password*][b**ENVIRONMENT(DUMMY)**]) |
INDATASET(*entryname* [*/ password*] [b**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.

INDATASET(*entryname* [*/ password*])
specifies the name of the entry to be copied. If INDATASET is specified, the entryname is dynamically allocated.

password
is the read or higher level password of the data to be copied. If the data is password protected, the read password must be supplied. If a catalog is to be copied, the master password is required. You can copy a base cluster in alternate-key sequence by specifying a path name as the data set name in the DD statement.

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.

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. For VSAM data sets, the data set name can be that of a path.

OUTDATASET(*entryname* [*/ password*])
specifies the name of the output data set. If OUTDATASET is specified, the entryname is dynamically allocated.

password
specifies the update or higher level password for a password-protected output data set or path.

REPLACE | NOREPLACE
specifies whether a record in the source cluster (INFILE) is to replace a record in the target cluster (OUTFILE) when the source cluster is copied into the target cluster. When a key-sequenced cluster or relative-record cluster (that is, the source cluster) is copied, its records might have keys or relative-record numbers identical to the keys or relative-record numbers of data records in the target cluster.

REPLACE cannot be used if the output data set is identified as a path through an alternate index, or if the output data set is a base cluster whose upgrade set includes an alternate index defined with the unique-key attribute.

Since the catalog Reload function automatically performs a Replace function, REPLACE | NOREPLACE is ignored if the output data set is a VSAM catalog.

When a key-sequenced cluster, other than a catalog, is copied and you specify:

- REPLACE, each source record whose key matches a target record's key replaces the target record. Otherwise, the source record is inserted into its appropriate place in the target cluster.
- NOREPLACE, target records aren't replaced by source records. Each source record whose key matches a target record's key is not inserted into the target data set and a "duplicate record" message is issued.

When a relative-record cluster is copied and you specify:

- REPLACE, each source record whose relative-record number identifies a data record (rather than an empty slot) in the target cluster replaces the target data record. Otherwise, the source data record is inserted into the empty slot its relative-record number identifies.
- NOREPLACE, target records aren't replaced by source records. Each source record whose relative-record number identifies a target data record instead of an empty slot is not inserted into the target data set and a "duplicate record" message is issued.

REUSE | NOREUSE
specifies whether the output data set is to be opened as a reusable data set. When REUSE is specified, the output data set, specified with OUTFILE, is opened as a reusable data set regardless of whether or not it was defined as REUSE (see the DEFINE CLUSTER command description). If the data set was defined as REUSE, its high-used relative byte address (RBA) is reset to zero (that is, the data set is effectively empty) and the operation proceeds. If the data set is NOREUSE, the REPRO command terminates with an error message, unless the data set is empty.

If the NOREUSE attribute is specified (either explicitly or by default) and OUTFILE identifies a nonempty data set, records are written at the end of an entry-sequenced data set. Processing of key-sequenced and relative-record data sets is controlled by the REPLACE/NOREPLACE option used.

FROMKEY(*key* **) | FROMADDRESS(** *address* **) | FROMNUMBER(** *number* **) | SKIP(** *count* **)**
specifies the location in the input data set from which copying is to start. You can use only one of the four 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.

FROMKEY(*key* **)**
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 specified only when an alternate index, a key-sequenced VSAM data set, or an indexed-sequential (ISAM) nonVSAM data set is being copied. If FROMKEY is specified for the starting location, TOADDRESS cannot identify 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.

FROMADDRESS(*address* **)**
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 physical sequential order instead of in logical sequential order. FROMADDRESS can be coded only for VSAM data sets. FROMADDRESS cannot be specified when the data set is being accessed through a path.

FROMNUMBER(*number* **)**
specifies, for a relative-record cluster, the relative-record number of the first record you want copied.

SKIP(*count* **)**
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). SKIP should not be specified when you access the data set through a path; the results are unpredictable.

address, number, and *count* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form; the expression cannot be longer than one fullword (eight decimal or hexadecimal numbers, or 32 binary numbers).

TOKEY(*key* **) | TOADDRESS(** *address* **) | TONUMBER(** *number* **) | COUNT(** *count* **)**
specifies the location in the data set being copied at which copying is to stop. You can use only one of the four 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(*key* **)**
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 can be specified only when an alternate index, a key-sequenced VSAM data set, or an indexed-sequential (ISAM) nonVSAM data set is being copied. 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(*address* **)**
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 physical sequential order instead of in 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. TOADDRESS cannot be specified when the data set is accessed through a path.

TONUMBER(*number* **)**
specifies the relative-record number of the last record you want copied. TONUMBER can be specified only when you copy a relative-record cluster.

COUNT(*count* **)**
specifies the number of logical records you want copied. COUNT should not be specified when you access the data set through a path; the results are unpredictable.

address, number, and *count* can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form; the expression cannot be longer than one fullword (eight decimal or hexadecimal numbers, or 32 binary numbers).

REPRO Examples

Copy Records Into a VSAM Cluster: Example 1

In this example, records from an indexed-sequential data set, ISAMDSET, are copied into key-sequenced VSAM cluster, D40.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)
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     REPRO -
          INFILE(INDSET1) -
          OUTDATASET(D40.EXAMPLE.KSDS1)
/*
```

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.
- 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, D40.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.
- OUTDATASET, which identifies the key-sequenced VSAM cluster into which the input records are to be copied. This data set is dynamically allocated by Access Method Services.

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 a key-sequenced VSAM cluster, D40.MYDATA. Next, records are copied from the key-sequenced cluster, D40.MYDATA, into an entry-sequenced cluster, ENTRY.

```
//REPRO2   JOB  ...
//JOBCAT   DD   DSNAME=MYCAT,DISP=SHR
//STEP1    EXEC PGM=IDCAMS
//INPUT    DD   DSNAME=SEQ.D27V,DISP=SHR,DCB=(BUFNO=6)
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     REPRO -
          INFILE(INPUT) -
          OUTDATASET(D40.MYDATA)
/*
//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, D40.MYDATA. STEP1's job control statements are:

- JOBCAT DD, which makes the catalog MYCAT available for this job.
- 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.
- 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, D40.MYDATA. Its parameters are:

- INFILE, which points to the INPUT DD statement. The INPUT DD statement identifies the source, or input, data set.
- OUTDATASET, which identifies the key-sequenced cluster into which the input records are to be copied. The data set is dynamically allocated by Access Method Services.

STEP2: Access Method Services copies some of the records of the key-sequenced cluster D40.MYDATA into another entry-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 key-sequenced cluster, D40.MYDATA, that contains the input records.
- OUTPUT DD, which identifies the entry-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, D40.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, key-sequenced cluster.
- OUTFILE, which points to the OUTPUT DD statement. The OUTPUT DD statement identifies the entry-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.

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) -
               UDATEPW(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 is:

- 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 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 20 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) specify 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 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.

Unload a VSAM User Catalog: Example 4

This example shows how a VSAM user catalog can be unloaded to tape using the catalog unload/reload feature of the REPRO command.

```
//UNLOAD   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT2,DISP=OLD
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//CATIN    DD     DSNAME=D27UCAT2,DISP=OLD
//CATOUT   DD     DSNAME=PORTABLE.TAPE,LABEL=(1,SL),
//      DISP=NEW,UNIT=2400-3,VOL=SER=TAPE01,
//      DCB=(DEN=3,RECFM=VB,LRECL=516,BLKSIZE=5164)
//SYSIN    DD     *
     REPRO -
          INFILE(CATIN/UCATMRPW) -
          OUTFILE(CATOUT)
/*
```

The job control statements are:

- JOBCAT DD, which is required, and which identifies and allocates the user catalog as the job catalog.
- CATIN DD, which describes and allocates the user catalog, D27UCAT2, as a VSAM data set to be opened and used by the REPRO command as the *source* data set for the unload operation.

- CATOUT DD, which describes and allocates a magnetic tape file to contain the copy of the catalog. The DCB parameters must be specified as shown. The BLKSIZE parameter can be any multiple of LRECL, plus 4, where the minimum LRECL=505.

The REPRO command causes the VSAM user catalog D27UCAT2 to be unloaded (that is, copied) to a magnetic tape file. The REPRO command's parameters are:

- INFILE, which is required, and which identifies the user catalog as an input, or *source*, VSAM data set. The catalog's master password is required to open it as a data set.
- OUTFILE, which is required, and which describes the magnetic tape file, or *target* data set that is to contain the catalog's copy.

Reload an Unloaded VSAM User Catalog: Example 5

This example shows how a VSAM user catalog can be reloaded from the backup copy created in the previous example using the catalog unload/reload feature of the REPRO command.

```
//RELOAD   JOB    ...
//JOBCAT   DD     DSNAME=D27UCAT2,DISP=OLD
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//CATIN    DD     DSNAME=PORTABLE.TAPE,LABEL=(1,SL),
//     DISP=OLD,UNIT=2400-3,VOL=SER=TAPE01,DCB=DEN=3
//CATOUT   DD     DSNAME=D27UCAT2,DISP=OLD
//SYSIN    DD     *
     REPRO -
          INFILE(CATIN) -
          OUTFILE(CATOUT/UCATMRPW)
/*
```

The job control statements are:

- JOBCAT DD, which is required and which describes and allocates the user catalog to be reloaded. The JOBCAT statement specifies that D27UCAT2 is the job catalog.
- CATIN DD, which describes and allocates the magnetic tape file that contains the unloaded copy of the catalog. Since a standard label tape is used, DCB parameters of record format, record size, and blocksize need not be specified. For a non-labelled tape, the same parameters as in the previous example must be specified.
- CATOUT DD, which describes the user catalog as a VSAM data set to be opened and used by REPRO as the target data set for the reload operation.

The REPRO command causes a VSAM user catalog to be reloaded from the backup copy created in the previous example. The REPRO command's parameters are:

- INFILE, which is required, and which identifies the magnetic tape file that contains the unloaded, or backup, copy of the user catalog.
- OUTFILE, which is required and which identifies the user catalog as a VSAM data set to be opened for record processing. The master password allows opening the catalog as a data set.

RESETCAT

The format of the RESETCAT command is:

RESETCAT	**CATALOG(***catname* [*/password*][*dname*]) {**CRAVOLUMES((***volser1* [*devtype*])[(*volser2* [*devtype*])]...) | **CRAFILES((***dname1* {**ALL** | **NONE**}) (*dname2* {**ALL** | **NONE**})...)} [**WORKFILE(***dname* [*/password*])] [**WORKCAT(***catname* [*/password*])] [**MASTERPW(***password*)] [**IGNORE** | **NOIGNORE**)]

"Appendix F: Command Parameters Summary" contains a table for the RESETCAT command that shows each parameter, its abbreviation, its defaults (if any), and an example of its use.

RESETCAT Parameters

CATALOG(*catname* [*/password*][*dname*])
identifies the catalog to be reset. The catalog specified by *catname* must have the RECOVERABLE attribute. Exclusive control of the catalog is required throughout the duration of the command; also, no VSAM data sets cataloged in the catalog may be open. The master catalog may not be reset while it is in use as a master catalog.

password
specifies the catalog's master password. If the catalog is password-protected, you must specify the master password of the catalog.

dname
specifies the name of the DD statement for the catalog being reset. If *dname* is not specified, the catalog will be dynamically allocated. It must also be specified in a JOBCAT or STEPCAT.

CRAVOLUMES((*volser* [*devtype*])...)
specifies the volume serial numbers and corresponding device types of the volumes used in resetting the catalog. Each volume contains a catalog recovery area (CRA) which has a copy of the catalog records that describe the objects on the volume. The volume is allocated dynamically.

devtype
specifies the device type for the associated volser. If you do not specify *devtype,* the volume must be mounted and on a unit marked permanently resident or reserved. If you specify a generic name for *devtype,* the volume need not be mounted.

If a volume contains a multivolume data set whose catalog entry is being reset, other volumes of that multivolume data set may be needed during the reset operation; if needed and not specified for reset, these volumes will be allocated dynamically.

CRAFILES((*dname1* {ALL | NONE})...)
specifies a list of DD statements which identify the volumes and devices to be used to reset the catalog. Each volume contains a CRA which has a copy of the catalog records describing the VSAM objects on that volume. ALL specifies that the catalog records in the CRAs on the volume specified by *dname1* are to be reset. If a volume contains a multivolume data set whose catalog entry is being reset, other volumes of that multivolume data set may be needed during the reset operation; you should specify these via additional *dname1* parameters. Use the NONE subparameter, if you do not want the entries in these additional volumes to be used to reset the catalog. If a volume is needed and you do not specify the DD statements provided by CRAFILES, it will be allocated dynamically.

WORKFILE(*dname* [/*password*])
specifies the name of a DD statement that identifies a VSAM data set name and a list of volume serial numbers of volumes containing VSAM data spaces and units to be used by RESETCAT for a work file. This data set will be defined by RESETCAT and used as temporary storage while processing the command; it will be deleted at the end of the command. The data set must not be already defined. Space requirements for the data set are noted in "WORKFILE Space Requirements." If you do not specify WORKFILE, IDCUT1 is used as a default *dname.*

password
If you desire, you may specify *password,* a password to be used by RESETCAT to password-protect the workfile. Since the work file will contain a copy of the catalog records, it may contain security-sensitive information. The password you specify will become the work file's master password. You may specify the password only if *dname* is explicitly specified.

WORKCAT(*catname* [/*password*])
specifies the name of the catalog in which RESETCAT defines the WORKFILE. This catalog must not be the same as the catalog being reset. If the catalog is password-protected, the update or higher level password is required. The catalog must be specified in a JOBCAT or STEPCAT DD statement. If you do not specify the WORKCAT parameter, the catalog will be chosen using existing rules for "Order of Catalog Use: DEFINE" in the chapter "Creating and Cataloging Objects." Further constraints must be followed if you do not specify the WORKCAT parameter, that is, it must be the first DD statement in the JOBCAT or STEPCAT DD statement concatenation. However, if the work file catalog is the master catalog, then you must specify the WORKCAT parameter and it must be the last DD statement in the JOBCAT or STEPCAT DD statement concatenation. The resultant catalog must not be the same one being reset.

MASTERPW(*password*)
specifies the master catalog's master password. The master catalog's master password is required when the master catalog is password protected.

IGNORE | NOIGNORE
specifies whether RESETCAT should continue processing and force the reset when certain errors are encountered. These errors may be:

- I/O error in the catalog
- I/O error in the CRA
- The volume record or its extensions in the CRA are in error.

When you specify IGNORE, the RESETCAT command will try to recover as much information as possible and reset the catalog's description of those data sets on the specified volumes. For example, if IGNORE is specified and an I/O error occurs while processing a record from a CRA, that record will be ignored. In the process, data sets that could not be recovered on a particular volume would be marked unusable on that volume, and the space that was assigned to them would be freed. NOIGNORE specifies that these errors result in immediate termination.

RESETCAT Examples

Resetting a Catalog: Example 1

This example illustrates the use of RESETCAT to reset a catalog (USERCAT1), one of whose owned volumes (VSER01) has been restored. The other volume owned by USERCAT1 is VSER02.

```
//RECOVER  JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//STEPCAT  DD   DSN=USERCAT1,DISP=OLD
//         DD   DSN=AMASTCAT,DISP=OLD
//IDCUT1   DD   DSN=WORKFILE,DISP=OLD,
//               VOL=SER=231401,UNIT=2314,AMP='AMORG'
//CRAVOL1  DD   DISP=OLD,VOL=SER=VSER01,UNIT=2314
//CRAVOL2  DD   DISP=OLD,VOL=SER=VSER02,UNIT=2314
//DDCAT    DD   DSN=USERCAT1,DISP=OLD
//SYSIN    DD   *
    RESETCAT -
      CATALOG(USERCAT1/MASTER DDCAT) -
      CRAFILES((CRAVOL1 ALL)(CRAVOL2 NONE)) -
      WORKCAT(AMASTCAT/MCATMRPW) -
      MASTERPW(MCATMRPW) -
      IGNORE
/*
```

The job control statements are:

- STEPCAT DD, which makes the user catalog, USERCAT1, and the master catalog, AMASTCAT, available for the step. These catalogs must be available for the catalog to be reset and for the work file to be defined, respectively. Note that when the work file is to be defined in the master catalog, the master catalog must be last in the concatenation and the catalog name is supplied via the WORKCAT parameter.
- IDCUT1 DD, which identifies the data set to be used for the work file.
- CRAVOL1 DD, which identifies and allocates a volume whose catalog recovery area is to be used to reset the catalog.
- CRAVOL2 DD, which identifies and allocates a volume whose catalog recovery area may change as a result of resetting the catalog recovery area on volume VSER01. This volume is not reset in the catalog.
- DDCAT DD, which allocates the catalog to be reset.

The RESETCAT command causes catalog USERCAT1 to be reset from the catalog recovery area on VSER01. The RESETCAT command's parameters are:

- CATALOG, which identifies the catalog to be reset. The master password of the catalog is required to update the catalog. The DD statement for the catalog is also identified.
- CRAFILES, which specifies a list of DD statements which identify the volumes to be used to reset the catalog. ALL specifies that the catalog records in the CRA on the volume specified by the CRAVOL1 DD statement are to be reset (in the catalog). NONE allows the CRAVOL2 DD statement to be specified for a CRA volume which is not to be used for reset, but which may be needed as a result of the reset (for example, a multivolume data set resides on a reset volume and on a non-reset volume).
- WORKCAT, which specifies the name of the catalog in which to define the work file. WORKCAT must be specified if the work file is to be defined in the master catalog.
- IGNORE, which specifies that RESETCAT should force the reset of the catalog despite certain errors which may be encountered during the RESETCAT processing.

Resetting a Catalog: Example 2

This example illustrates the use of RESETCAT to reset a catalog (USERCAT1) after the volume (VSER00) on which the catalog resides has been restored. The volumes owned by USERCAT1, VSER01, and VSER02 have been determined to be out of synchronization with the catalog by running a LISTCRA with the COMPARE option for all volumes owned by USERCAT1.

```
//RECOVER  JOB  ...
//JOBCAT   DD   DSN=USERCAT2,DISP=OLD
//         DD   DSN=USERCAT1,DISP=OLD
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//WORKF    DD   DSN=WORKFILE,DISP=OLD,UNIT=(2314,2),
//              VOL=SER=(231401,231402),AMP='AMORG'
//SYSIN    DD   *
    RESETCAT -
       CATALOG(USERCAT1) -
       CRAVOLS((VSER01 2314)(VSER02 2314))
       WORKFILE(WORKF/WRKPSW) -
       IGNORE
```

The job control statements are:

- JOBCAT DD, which makes the user catalogs USERCAT2 and USERCAT1 available for the job. These catalogs must be available for the workfile to be defined and for the catalog to be reset, respectively.
- WORKF DD, which identifies the data set to be used for the work file.

  Note: All other required volumes will be dynamically allocated by RESETCAT.

The RESETCAT command causes catalog USERCAT1 to be reset from the catalog recovery areas on VSER01 and VSER02. The RESETCAT parameters are:

- CATALOG, which identifies the catalog to be reset.
- CRAVOLS, which specifies a list of volumes and their device types to be used to reset the catalog.
- WORKFILE, which specifies the DD statement for the work file and a password to be used as the work file's master password.
- NOIGNORE, which specifies that RESETCAT should not force the reset of the catalog if certain errors are encountered.

Resetting a Catalog: Example 3

This three-part example illustrates the use of RESETCAT to reset a catalog (USERCAT1 on volume VSER00) which must be reallocated on a different volume (VSER03). Such a situation may occur when an irreparable physical failure has befallen the catalog. In this example, the catalog recovery area for the damaged catalog volume is assumed to be accessible and valid and thus included for reset in the newly defined catalog. Note that the newly defined catalog bears the same name as the damaged catalog. The other volumes owned by USERCAT1 are VSER01 and VSER02. The catalog is first disconnected, then redefined.

```
//DISC     JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//STEPCAT  DD   DSN=USERCAT1,DISP=OLD
//SYSIN    DD   *
    EXPORT USERCAT1 DISCONNECT
/*

//STEP2    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//UCATDD   DD   VOL=SER=VSER03,UNIT=3330,DISP=OLD
//CRAVOL1  DD   VOL-SER=VSER01,UNIT=2314,DISP=OLD
//CRAVOL2  DD   VOL=SER=VSER02,UNIT=2314,DISP=OLD
//SYSIN    DD   *
    DEFINE UCAT -
      (NAME (USERCAT1) FILE(UCATDD) VOL(VSER03) -
      RECOVERABLE CYL(20 1)) -
      DATA(CYL(10 1)) INDEX(CYL(1))
/*

//STEP3    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//STEPCAT  DD   DSN=USERCAT1,DISP=OLD
//STEPCAT  DD   DSN=AMASTCAT,DISP=OLD
//         DD   DSN=AMASTCAT,DISP=OLD
//IDCUT1   DD   DSN=WORKFILE,DISP=OLD,UNIT=2314,
                VOL=SER=231401,AMP='AMORG'
//SYSIN    DD   *
    RESETCAT -
        CATALOG(USERCAT1) -
        WORKCAT(AMASTCAT) -
        CRAVOLS((VSER00 3330)(VSER01 2314)(VSER02 2314))
/*
```

STEP1:

The user catalog, USERCAT1, is disconnected from the system by using the EXPORT command.

STEP2:

The user catalog, USERCAT1, is redefined on the volume VSER03.

STEP3:

The user catalog USERCAT1 is reset from the previous catalog's owned volumes.

The job control statements are:

- STEPCAT DD, which makes the user catalog, USERCAT1, and the master catalog, AMASTCAT, available for the step. These catalogs must be available for the catalog to be reset, and for the work file to be defined, respectively.
- IDCUT1 DD, which identifies the data set to be used for the work file.

The RESETCAT command causes the redefined catalog USERCAT1 to be reset from the catalog recovery areas on volumes VSER00, VSER01, and VSER02. The RESETCAT command's parameters are:

- CATALOG, which identifies the catalog to be reset.
- WORKCAT, which specifies the name of the catalog in which to define the workfile. WORKCAT must be specified if the workfile is to be defined in the master catalog.
- CRAVOLS, which specifies a list of volumes and their device types to be used to reset the catalog.

VERIFY

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, alternate indexes, (or their components) and catalogs can be verified.

To use the VERIFY command to verify a catalog, Access Method Services must be authorized. See "Authorized Program Facility (APF)" in *OS/VS2 System Programming Library: Supervisor* for information about program authorization.

The format of the VERIFY command is:

VERIFY	**FILE(***dname* [*/ password*]) | **DATASET(***entryname* [*/ password*])

VERIFY
specifies that the end-of-data-set information is to be verified. VERIFY can be abbreviated VFY.

FILE(*dname* [*/ password*] **)**
dname specifies the name of a DD statement that identifies the cluster or component to be verified.

password
is the control or master password of a password-protected cluster or component or the master password of a password-protected catalog.

DATASET(*entryname* [*/ password*] **)**
specifies the name and password of the object to be verified. If DATASET is coded, the object is dynamically allocated.

password
is the control or master password of a password-protected object, or is the master password of the object's (password-protected) catalog.

VERIFY Example

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:

```
//VERIFY   JOB  ...
//JOBCAT   DD   DSNAME=D27UCAT1,DISP=SHR
//FIXEOD   EXEC PGM=IDCAMS
//SYSIN    DD   *
           LISTCAT   ENTRIES(FAROUT) -
                     ALL
           VERIFY    DATASET(FAROUT)
           LISTCAT   ENTRIES(FAROUT) -
                     ALL
/*
//
```

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.

See "Appendix B: Interpreting LISTCAT Output Listings" for details on the order in which catalog records are to be listed and the meanings of the listed records.

[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.

CONTROLLING COMMAND EXECUTION

This chapter describes:

- 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 indicates that the requested function could not be performed. This condition code is set as a result of a logical error. A logical error condition exists when inconsistent parameters are specified, when a too small or too large value is specified for key length, recordsize, or bufferspace, or when required parameters are missing. More information on logical errors that occur during VSAM record processing is in the *OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide.*
- **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 (a SYSPRINT dd statement was missing, for example); 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**} ƀ *comparand* ƀ *number* **THEN**[ƀ *command* | **DO** *command set* **END**] [**ELSE**[ƀ *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. It is possible to terminate all processing simply by setting MAXCC or LASTCC to 16.

The format of the SET command is:

SET	{**MAXCC** | **LASTCC**} = *number*

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.

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 statement (in the JCL).

The format of the PARM command is:

PARM	[**TEST**({[**TRACE**] [**AREAS**(*areaid* [b *areaid*...])] [**FULL**((*dumpid* [b *count1* [b *count2*]]) [b (*dumpid*...)...])] | [**OFF**]})] [**GRAPHICS**(**CHAIN**(*chain*) | **TABLE**(*mname*))] [**MARGINS**(*leftmargin* b *rightmargin*)]

where:

TEST(
{ [**TRACE**]
[**AREAS** (*areaid* [ƀ *areaid* ...])]
[**FULL**((*dumpid* [ƀ *count1* [ƀ *count2*]])
[ƀ(*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* [ƀ *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 the *OS/VS2 Access Method Services Logic* for more information.

FULL((*dumpid* [ƀ *count1* [ƀ *count2*]]) [ƀ (*dumpid* ...)])
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 the *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.

If the FULL keyword is used, then an AMSDUMP DD statement must be provided. For example:

```
//AMSDUMP DD SYSOUT=A
```

OFF
specifies that testing is to stop. OFF cannot be specified with TRACE, AREAS, or FULL.

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 the /* and */ 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, an entry from 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
```

APPENDIX A: EXAMPLES OF JOBS USING ACCESS METHOD SERVICES COMMANDS

This set of examples is meant to show a wide range of functions provided by Access Method Services. It is assumed that the VSAM system catalog that exists is recoverable and security protected at the update, control and master password levels.

Example 1: Define a VSAM User Catalog

In this example, a user catalog is defined. The catalog is password protected. The user catalog in this example is security protected and is not defined with the recoverable attribute. The catalog's data space is defined to be 15 cylinders. The space explicitly specified for the data and index component of the catalog is taken from the catalog's data space. Since the total data space for the data and index component is less than the total catalog data space, the remaining catalog data space is available for suballocation. The volume VSER02 cannot be referenced by other VSAM catalogs since it now belongs to the catalog which it contains.

```
//DEFCAT   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     DEFINE USERCATALOG -
            (NAME(D27UCAT1) -
             MASTERPW(UCATMRPW) -
             UPDATEPW(UCATUPPW) -
             FOR(365) -
             CYLINDERS(15 5) -
             VOLUMES(VSER02) ) -
           DATA( -
             CYLINDERS(3 1) ) -
           INDEX( -
             CYLINDERS(1 1) ) -
           CATALOG(AMASTCAT/MCATUPPW)
/*
```

Explanation of Commands

1. The USERCATALOG parameter is required and NAME specifies the user catalog being defined.
2. The MASTERPW parameter specifies the master password for this catalog.
3. The UPDATEPW parameter specifies the update password of this catalog.
4. The FOR parameter specifies the retention period for this file--in this case, one year.
5. The CYLINDERS parameter specifies the amount of space to be allocated to the catalog's data space. A space parameter is required.
6. The VOLUMES parameter is required and specifies the volume containing this catalog. Access Method Services will dynamically allocate the catalog's volume, the volume should be mounted permanently RESIDENT or RESERVED to insure successful dynamic allocation.

7. The DATA parameter specifies the amount of space to be allocated to the catalog's data component.
8. The INDEX parameter specifies the amount of space to be allocated to the catalog's index component.
9. The CATALOG parameter is required since the master catalog is password protected. It specifies the name of the master catalog and its update password which is required to define into a protected catalog.

Example 2: Define a VSAM User Catalog and a VSAM Data Space

This example defines a user catalog by using the previously-defined user catalog, D27UCAT1, as a model. The second part of this example defines a VSAM data space on volume VSER04. Both volumes in this example become owned by the respective catalogs, D27UCAT2 and AMASTCAT; they cannot be referenced by any other VSAM catalogs.

The third part of this example defines alias names for the user catalogs, D27UCAT1 and D27UCAT2.

This example depends on the successful completion of the previous example for the user catalog, D27UCAT1.

```
//DEFCAT2  JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     DEFINE USERCATALOG -
          (NAME(D27UCAT2) -
          MODEL(D27UCAT1/UCATMRPW D27UCAT1) -
          VOLUME(VSER03) -
          CYLINDER(15 5) ) -
          CATALOG(AMASTCAT/MCATUPPW)
     DEFINE SPACE -
          (VOLUME(VSER04) -
          CYLINDERS(10 1) ) -
          CATALOG(AMASTCAT/MCATUPPW)
     DEFINE ALIAS -
          (NAME(D40) -
           RELATE(D27UCAT1))
     DEFINE ALIAS -
          (NAME(D50) -
           RELATE(D27UCAT2))
/*
```

Explanation of Commands

The first DEFINE command defines a user catalog on volume VSER03. The catalog's attributes are modeled from a previously-defined user catalog, D27UCAT1. The VOLUME and CYLINDERS parameters are required parameters, even though a model catalog is used.

1. The USERCATALOG parameter is required and NAME specifies the name of the user catalog, D27UCAT2.
2. The MODEL parameter names the object to be used as a model and its master password, D27UCAT1/UCATMRPW. The next field names the catalog that contains the model object's catalog entries. In this case,

D27UCAT1's self-describing entries describe the object model, D27UCAT1.

3. The VOLUME parameter names the volume that is to contain the user catalog. Access Method Services will dynamically allocate the catalog's volume; the volume should be mounted permanently RESIDENT or RESERVED to insure successful dynamic allocation.
4. The CYLINDERS parameter specifies the primary and secondary allocation amounts (of cylinders) to be allocated to the catalog's data space. A space parameter specifying the total amount of space is required. The space is allocated to data and index in the same way as that of the model.
5. The CATALOG parameter is required since the master catalog is password protected. It names the master catalog, AMASTCAT, and specifies its update password. When a user catalog is defined, a user catalog connector entry is built and written into the VSAM master catalog. The user catalog connector entry enables the user catalog's volume to be located through the master catalog.

The second DEFINE command defines a VSAM data space on volume VSER04 and establishes the volume's ownership by the master catalog, AMASTCAT. All subsequent VSAM objects created on volume VSER04 are required to be cataloged in the volume's catalog, AMASTCAT.

1. The SPACE parameter is required.
2. The VOLUME parameter names the volume that is to contain the data space. Access Method Services will dynamically allocate the volume; the volume should be mounted permanently RESIDENT or RESERVED to insure successful dynamic allocation.
3. The CYLINDERS parameter specifies the primary and secondary allocation amounts of cylinders to be allocated to the data space.
4. The CATALOG parameter identifies the master catalog, AMASTCAT, and supplies its update password.

The defines of the alias names permit references to the respective user catalogs by the alias names. Alias names for VSAM catalogs also provide a way by which data set names can be used to direct the search for catalog information about data sets. Some of the following examples will show how data set naming conventions are used to simplify JCL specification of JOBCAT and STEPCAT DD statements.

1. The ALIAS parameter is required and NAME specifies the alias name of the user catalog.
2. The RELATE parameter specifies the catalog for which the alias is defined.

Example 3: Define VSAM Data Sets

This example defines four VSAM data sets: two key-sequenced data sets into the user catalog, D27UCAT1, a relative-record data set into the system catalog, and a reusable entry-sequenced data set into the user catalog, D27UCAT2.

This example depends on the successful completion of Example 1 for the user catalog.

```
//DEFVSAM  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//SYSIN    DD     *
      DEFINE -
        CLUSTER ( -
          NAME(D40.MYDATA) -
          VOLUMES(VSER02) -
          RECORDS(1000  500)  ) -
        DATA( -
          NAME(D40.KSDATA) -
          KEYS(15 0) -
          RECORDSIZE(250 250) -
          FREESPACE(20 10) -
          BUFFERSPACE(25000)  ) -
        INDEX( -
          NAME(D40.KSINDEX) -
          IMBED ) -
        CATALOG(D27UCAT1/UCATUPPW)

     DEFINE CLUSTER -
            (NAME(D40.EXAMPLE.KSDS1) -
             READPW(KSD1PSWD) -
             MODEL(D40.MYDATA)) -
          DATA( -
            (NAME(D40.EXAMPLE.KSDS.DATA) -
             UNIQUE -
             CYLINDERS(2 1) ) -
          INDEX( -
            (NAME(D40.EXAMPLE.KSDS.INDEX) -
             UNIQUE -
             CYLINDERS(1 1) ) -
          CATALOG(D27UCAT1/UCATUPPW)

     IF LASTCC = 0 -
        THEN -
        LISTCAT ENTRIES( -
                D40.EXAMPLE.KSDS1/KSD1PSWD) -
                ALL

     DEFINE CLUSTER -
            (NAME(EXAMPLE.RRDS1) -
             VOLUMES(VSER04) -
             TRACKS(10 5) -
             RECORDSIZE(100 100) -
             NUMBERED  ) -
          CATALOG(AMASTCAT/MCATUPPW)

     IF LASTCC = 0 -
        THEN -
        LISTCAT ENTRIES( -
                  EXAMPLE.RRDS1) -
                CLUSTER -
                ALL
```

```
DEFINE CLUSTER -
            (NAME(D50.EXAMPLE.ESDS1) -
             MASTERPW(ESD1MRPW) -
             UPDATEPW(ESD1UPPW) -
             VOLUMES(VSER03) -
             SPANNED -
             REUSE -
             NONINDEXED -
             CYLINDERS(2 1) -
             RECORDSIZE(2500 3000) ) -
          CATALOG(D27UCAT2/UCATMRPW)

     IF LASTCC = 0 -
        THEN -
        LISTCAT ENTRIES( -
                D50.EXAMPLE.ESDS1) -
                ALLOCATION
/*
```

Explanation of Commands

The first DEFINE command defines a key-sequenced data set on volume VSER02. The high-level name of the data set is the alias name of the catalog into which it is being defined.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The VOLUMES parameter is required and specifies the volume containing the data set.
3. The RECORDS parameter specifies the space to be allocated to the cluster. This is a required parameter.
4. The DATA parameter is required when attributes are to be explicitly specified for the data component of the cluster. The NAME parameter specifies the name of the data component.
5. The KEYS parameter specifies the key length and offset.
6. The RECORDSIZE parameter specifies the average and maximum record sizes.
7. The BUFFERSPACE parameter is specified for improved performance.
8. The INDEX parameter is required when attributes are to be explicitly specified for the index component of the cluster. The NAME parameter specifies the name of the index component.
9. The IMBED parameter specifies that the index sequence set is to be placed with the data component.
10. The CATALOG parameter is required since the catalog is password protected.

The second DEFINE command defines a unique key-sequenced data set on volume VSER04. The high-level name of the data set is the alias name of the catalog into which it is being defined. This example shows how data set attributes can be specified by modeling and direct specification.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The READPW parameter specifies the read password of this cluster. Since no master password is defined, this password will be propagated up

to the master level. Thus this cluster is security protected even though its model was not protected.

3. The MODEL parameter specifes the name of the data set to be used as the model.
4. The DATA parameter is required if attributes are to be specified for the data component. The NAME parameter specifies the name of the data component. If a name is not specified, a name is generated.
5. The UNIQUE parameter specifies that this portion of the data set is the only one that occupies the data space allocated to it.
6. The CYLINDERS parameter specifies the amount of space to be allocated to the cluster's data component.
7. The INDEX parameter is required if attributes are to be specified for the index component. The NAME parameter specifies the name of the index component. If a name is not specified, a name is generated.
8. The UNIQUE parameter specifies that this portion of the data set is the only one that occupies the data space allocated to it.
9. The CYLINDERS parameter specifies the amount of space to be allocated to the cluster's index component.
10. The CATALOG parameter is required since the user catalog is password protected. It specifies the name of the user catalog and its update password which is required to define into a protected catalog.

If the define of the unique key-sequenced data set was successful, then the following LISTCAT command is executed. The high-level name of the data set will direct the LISTCAT to the appropriate user catalog.

1. The ENTRIES and ALL parameters cause the entire catalog description of the data set just defined to be listed.

The third DEFINE command defines a suballocated relative-record data set into the VSAM data space on volume VSER04 which is owned by the system catalog.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The VOLUMES parameter is required and specifies the volume containing this data set.
3. The TRACKS parameter specifies the amount of space allocated to this data set. A space parameter is required.
4. The RECORDSIZE parameter specifies the average and maximum record sizes which, in the case of a relative-record data set, must be equal.
5. The NUMBERED parameter is required to specify that this is a relative-record data set.
6. The CATALOG parameter is required since the master catalog is password protected. It specifies the name of the master catalog and its update password which is required to define into a protected catalog.

If the define of the suballocated relative-record data set was successful, then the following LISTCAT command is executed.

1. The ENTRIES, CLUSTER and ALL parameters cause the entry just defined to be listed, limited, however, to the cluster entry (that is, the data component's entry is not listed).

The fourth DEFINE command defines a suballocated entry-sequenced data set on volume VSER03.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined. The high-level name of the data set is the alias name of the catalog into which it is being defined.
2. The MASTERPW parameter specifies the master password of this cluster.
3. The UPDATEPW parameter specifies the update password of this cluster.
4. The VOLUMES parameter is required and specifies the volume containing this data set.
5. The SPANNED parameter specifies that records may span control interval boundaries.
6. The REUSE parameter specifies that the data set can be reused, that is, reloaded without being deleted and redefined.
7. The NONINDEXED parameter is required to override the default (INDEXED).
8. The CYLINDERS parameter specifies the amount of space to be allocated to this data set. A space parameter is required.
9. The RECORDSIZE parameter specifies the average and maximum record sizes.
10. The CATALOG parameter specifies the name of the catalog into which the cluster is to be defined. The catalog's update or higher-level password is required.

If the define of the suballocated entry-sequenced data set was successful, then the following LISTCAT command is executed. The high-level name of the data set will direct the LISTCAT to the appropriate catalog.

1. The ENTRIES and ALLOCATION parameters cause the data set entry just defined to be listed, limited, however, to only volume and allocation information.

Example 4: Define NonVSAM and VSAM Data Sets

This example defines a nonVSAM data set into a user catalog, a VSAM key-sequenced data set into the system catalog, and a VSAM entry-sequenced data set into a user catalog.

```
//DEFVSM2  JOB   ...
//JOBCAT   DD    DSN=D27UCAT2,DISP=SHR
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
     DEFINE NONVSAM -
            (NAME(EXAMPLE.NONVSAM1) -
             VOLUMES(231401) -
             DEVICETYPES(2314) ) -
          CATALOG(D27UCAT1/UCATUPPW)
     IF LASTCC = 0 -
        THEN -
        LISTCAT NONVSAM -
                ALL -
                CATALOG(D27UCAT1)
     DEFINE CLUSTER -
            (NAME(EXAMPLE.KSDS2) ) -
          DATA( -
             MASTERPW(DAT2MRPW) -
             UPDATEPW(DAT2UPPW) -
             READPW(DAT2RDPW) -
             RECORDS(500 100) -
             EXCEPTIONEXIT(DATEXIT) -
             ERASE -
             FREESPACE(20 10) -
             KEYS(6 4) -
             RECORDSIZE(80 100) -
             VOLUMES(VSER04) ) -
          INDEX( -
             MASTERPW(IND2MRPW) -
             UPDATEPW(IND2UPPW) -
             READPW(IND2RDPW) -
             RECORDS(300 300) -
             IMBED -
             VOLUMES(VSER04)  ) -
          CATALOG(AMASTCAT/MCATUPPW)
     IF LASTCC = 0 -
        THEN -
        LISTCAT DATA -
                ALL -
                ENTRIES( -
                  EXAMPLE.KSDS2/DAT2MRPW) -
                CATALOG(AMASTCAT)
```

```
DEFINE CLUSTER -
            (NAME(EXAMPLE.ESDS2) -
             VOLUMES(VSER03) -
             SPANNED -
             CYLINDERS(2 1) -
             NONINDEXED -
             REUSE -
             MASTERPW(ESD2MRPW) -
             CONTROLPW(ESD2CTPW) -
             UPDATEPW(ESD2UPPW) -
             READPW(ESD2RDPW) -
           CATALOG(D27UCAT2/UCATMRPW)
     IF LASTCC = 0 -
        THEN -
          DO
          LISTCAT ENTRIES( -
                    EXAMPLE.ESDS2/ESD2MRPW) -
                  ALL
          LISTCAT NAME -
                  CATALOG(AMASTCAT/MCATMRPW)
          END
/*
```

Explanation of Job Control Language Statements

1. The JOBCAT DD statement describes the user catalog D27UCAT2 as a job catalog. All references will be to this job catalog unless otherwise directed.

Explanation of Commands

The first DEFINE command defines an existing nonVSAM data set into user catalog D27UCAT1.

1. The NONVSAM parameter is required and NAME specifies the nonVSAM object being defined.
2. The VOLUMES parameter is required and specifies the volume containing the data set.
3. The DEVICETYPES parameter is required and specifies the device type of the volume.
4. The CATALOG parameter specifying the name of the user catalog is required because:
   - a job catalog also appears in the job control language so this parameter explicitly directs the define to the user catalog.
   - the catalog is protected and its update password is required for the define.

If the definition of the nonVSAM entry was successful, then the following LISTCAT command is executed.

1. The NONVSAM and ALL parameters cause all the nonVSAM entries cataloged in D27UCAT1 to be listed.
2. The CATALOG parameter directs the LISTCAT to a specific user catalog.

The second DEFINE command defines a key-sequenced data set into a VSAM data space owned by the master catalog. Note that attributes are specified at the data and index level rather than the cluster level. Access Method Services will dynamically allocate the volume containing the catalog recovery area for the cluster, since the system catalog is recoverable. This requires that the volume containing the index component of the cluster be mounted permanently RESIDENT or RESERVED to ensure successful dynamic allocation.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The DATA component is not explicitly named and will have a name generated for it.
3. The MASTERPW, UPDATEPW, and READPW parameters specify the master, update, and read passwords, respectively, of this data component.
4. The RECORDS parameter specifies the amount of space to be allocated to the data component. A space allocation parameter is required.
5. The EXCEPTIONEXIT parameter specifies the name of the routine to be given control if an exception occurs while processing the data component.
6. The ERASE parameter specifies that the data component is to be overwritten with binary zeros when it is deleted.
7. The FREESPACE parameter specifies the percentage of space within control intervals and control areas, respectively, that is to remain free.
8. The KEYS parameter specifies the key length and offset.
9. The RECORDSIZE parameter specifies the average and maximum record sizes.
10. The VOLUMES parameter is required and specifies the volume containing this data component.
11. The INDEX component is not explicitly named and will have a name generated for it.
12. The MASTERPW, UPDATEPW, and READPW parameters specify the master, update, and read passwords, respectively, of this index component.
13. The RECORDS parameter specifies the amount of space to be allocated to the index component. A space allocation parameter is required.
14. The IMBED parameter specifies that the index sequence set is to be placed with the data component.
15. The VOLUMES parameter is required and specifies the volume containing this index component.
16. The CATALOG parameter is required since the master catalog is password protected. It specifies the name of the master catalog and its update password which is required to define into a protected catalog. This parameter is also required because a DD statement for the job catalog (JOBCAT) appears in the job control language and this define must, therefore, be explicitly directed to the master catalog.

If the definition of the key-sequenced data set was successful, then the following LISTCAT command is executed.

1. The DATA, ALL, and ENTRIES parameters cause all of the information contained in the data component entry to be listed.
2. The CATALOG parameter directs the LISTCAT to the master catalog.

The third DEFINE command defines an entry-sequenced data set.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The VOLUMES parameter specifies the volume (VSER03) which is to contain the data set being defined.
3. The SPANNED parameter specifies that records may span control interval boundaries.
4. The CYLINDERS parameter sppecifies the amount of space to be allocated to this data set. A space parameter is required.
5. The NONINDEXED parameter is required to override the default (INDEXED).
6. The REUSE parameter specifies that the data set can be reused, that is, reloaded without being deleted and redefined.
7. The MASTERPW, CONTROLPW, UPDATEPW, and READPW parameters specify passwords different from the passwords specified for the data set being modeled.
8. The CATALOG parameter is required since the user catalog is password protected.

If the define of the entry-sequenced data set was successful, then the following LISTCAT commands are executed.

1. The ENTRIES and ALL parameters of the first LISTCAT command cause all the cataloged information in the entry just defined to be listed.
2. The NAME parameter of the second LISTCAT command causes only the names of the objects cataloged in the master catalog to be listed.
3. The CATALOG parameter specifies the master password of the master catalog which allows access to all objects in the catalog and directs the LISTCAT to the master catalog.

Example 5: Copying and Printing

This example shows various techniques which can be used to load and print data sets using the REPRO and PRINT commands. This example depends on the successful completion of Examples 3 and 4 for the existence of VSAM data sets to load records into. This example also requires that various nonVSAM data sets exist, so that data records can be loaded into the VSAM data sets.

```
//COPYPRNT  JOB    ...
//STEP1     EXEC   PGM=IDCAMS
//SYSPRINT  DD     SYSOUT=A
//* NONVSAM INDEXED-SEQUENTIAL DATA SET
//INDSET1   DD     DSNAME=D40.IF100,DISP=OLD,DCB=DSORG=IS,
//    VOL=SER=231401,UNIT=2314
//* NONVSAM SEQUENTIAL DATA SET (VARIABLE-LENGTH
//*     RECORDS)
//INDSET2   DD     DSNAME=D40.SVB200,DISP=OLD,
//    VOL=SER=231401,UNIT=2314
//* NONVSAM SEQUENTIAL DATA SET (FIXED-LENGTH
//*     RECORDS)
//INDSET3   DD     DSNAME=D40.SF100,DISP=OLD,
//    VOL=SER=231401,UNIT=2314
//* NONVSAM  SEQUENTIAL DATA SET
//INDSET4   DD     DSNAME=EXAMPLE.NONVSAM1,DISP=OLD,
//    VOL=SER=231401,UNIT=2314
//* VSAM DATA SETS
//SYSIN     DD     *
   /* LOAD A VSAM KEY-SEQUENCED DATA SET */
   /* FROM AN ISAM DATA SET */
   REPRO   INFILE(INDSET1) -
           OUTDATASET(D40.EXAMPLE.KSDS1/KSD1PSWD)
   IF MAXCC = 0 -
      THEN -
      PRINT  INDATASET(D40.EXAMPLE.KSDS1/KSD1PSWD) -
             FROMKEY(X'40F0F0F0F0F6')
   /* LOAD A VSAM ENTRY-SEQUENCED DATA SET FROM AN */
   /* EXISTING VARIABLE UNBLOCKED SAM DATA SET */
   REPRO     INFILE(INDSET2) -
             OUTDATASET(D50.EXAMPLE.ESDS1/ESD1UPPW) -
             COUNT(030)
   IF LASTCC = 0 -
      THEN -
      PRINT  INDATASET(D50.EXAMPLE.ESDS1) -
             FROMADDRESS(170) -
             HEX
   /* LOAD A VSAM RELATIVE-RECORD DATA SET FROM AN */
   /* EXISTING SAM DATA SET                  */
   REPRO     INFILE(INDSET3) -
             OUTDATASET(EXAMPLE.RRDS1) -
             SKIP(10)
```

```
   IF LASTCC = 0 -
      THEN -
      PRINT  INDATASET( EXAMPLE.RRDS1 ) -
             TONUMBER( 25 )
   /* PRINT THE CONTENTS OF THE SAM DATA SET */
       PRINT INFILE( INDSET3 ) -
             COUNT( 20 ) -
             CHARACTER
   /* LOAD A VSAM KEY-SEQUENCED DATA SET */
   /* FROM A NONVSAM DATA SET */
   REPRO     INFILE( INDSET4 ) -
             OUTDATASET( EXAMPLE.KSDS2 )
   IF LASTCC = 0 -
      THEN -
      PRINT  INDATASET( EXAMPLE.KSDS2 ) -
             FROMKEY( AAAAJA ) -
             TOKEY( AAAAJ9 )
/*
```

Explanation of Job Control Language Statements

1. The INDSET1 DD statement describes the ISAM data set which is to be copied into the VSAM data set 'EXAMPLE.KSDS1'.
2. The INDSET2 DD statement describes the variable-length SAM data set which is to be copied into the VSAM data set 'EXAMPLE.ESDS1'.
3. The INDSET3 DD statement describes the SAM data set which is to be copied into the VSAM data set 'EXAMPLE.RRDS1'.
4. The INDSET4 DD statement describes the SAM data set which is to be copied into the VSAM data set 'EXAMPLE.KSDS2'.

Explanation of Commands

The first REPRO command causes a VSAM key-sequenced data set to be loaded from an ISAM data set.

1. The INFILE parameter is required and identifies the data set containing the source data. The *dname* of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter is required and identifies the name of the data set to be loaded. The data set is dynamically allocated by Access Method Services. The update or higher password of the VSAM data set is required.

If the REPRO operation was successfully executed, then the contents of the VSAM key-sequenced data set just loaded are printed. The format of the listing is DUMP, since the default is taken.

1. The INDATASET parameter is required and identifies the name of the data set to be printed. The data set is dynamically allocated by Access Method Services. The update or higher-level password of the VSAM data set is required.
2. The FROMKEY parameter specifies that printing is to begin with the record whose key (high order three bytes) is greater than or equal to '00006' (that is, the character equivalent of X'40F0F0F0F0F6').

The second REPRO command causes a VSAM entry-sequenced data set to be loaded from an existing variable unblocked SAM data set.

1. The INFILE parameter is required and identifies the data set containing the source data. The *dname* of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter is required and identifies the name of the data set to be loaded. The data set is dynamically allocated by Access Method Services. The update or higher password of the VSAM data set is required.
3. The COUNT parameter specifies that the first 30 records of the SAM data set are to be loaded.

If the REPRO operation was successfully executed, then the contents of the VSAM entry-sequenced data set just loaded are printed in hexadecimal format.

1. The INDATASET parameter is required and identifies the name of the data set to be printed. The data set is dynamically allocated by Access Method Services. Since the data set is not read protected, no password is required.
2. The FROMADDRESS parameter specifies that the first record printed is that record whose relative byte address is exactly equal to 170.
3. The HEX parameter specifies that the listing is to be in hexadecimal format.

The third REPRO command causes a VSAM relative-record data set to be loaded from an existing fixed SAM data set. The relative-record data set can receive only fixed length records that equal its defined record length.

1. The INFILE parameter is required and identifies the data set containing the source data. The *dname* of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter is required and identifies the name of the data set to be loaded. The data set is dynamically allocated by Access Method Services.
3. The SKIP parameter specifies that the first 10 records of the SAM data set are to be bypassed.

If the REPRO operation was successfully executed, then the contents of the VSAM relative-record data set just loaded are printed.

1. The INDATASET parameter is required and identifies the name of the data set to be printed. The data set is dynamically allocated by Access Method Services. Since the data set is not read protected, no password is required.
2. The TONUMBER parameter limits the output to those relative records with relative-record number less than or equal to 25.

The PRINT command causes the contents of the SAM data set to be printed.

1. The INFILE parameter is required and identifies the data set to be printed. The *dname* of the DD statement for this data set must be identical to this name.
2. The COUNT parameter specifies that only 20 records are to be printed.

3. The CHARACTER parameter specifies that the listing is to be in character format.

The last REPRO command causes a VSAM key-sequenced data set to be loaded from a nonVSAM data set.

1. The INFILE parameter is required and identifies the data set containing the source data. The *dname* of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter is required and identifies the name of the data set to be loaded. The data set is dynamically allocated by Access Method Services. Note that a password is not required since the cluster component is not password protected although its data and index components are.

If the REPRO operation was successfully executed, then the contents of the VSAM key-sequenced data set just loaded are printed. The FROMKEY and TOKEY parameters are used to limit the output to a specific range of keys.

1. The INDATASET parameter is required and identifies the name of the data set to be printed. The data set is dynamically allocated by Access Method Services. No password is required because the cluster component is not password protected.
2. The FROMKEY and TOKEY parameters specify the keys at which printing is to begin and end, respectively.

Example 6: Record Replacement

This example shows techniques for modifying the contents of VSAM data sets using the REPRO command.

This example depends on the successful completion of Example 5 for the existence of nonempty VSAM data sets.

```
//REPLACE  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//INDSET4  DD     DSNAME=EXAMPLE.NONVSAM2,DISP=OLD,
//      VOL=SER=231401,UNIT-2314
//SYSIN    DD     *
   REPRO    INFILE( INDSET4 ) -
            OUTDATASET( EXAMPLE.KSDS2 ) -
            REPLACE

   IF LASTCC = 0 -
      THEN -

      PRINT    INDATASET( EXAMPLE.KSDS2 ) -
               FROMKEY( AAAAJA ) -
               TOKEY( AAAAJ9 )

   REPRO    INDATASET( EXAMPLE.RRDS1 ) -
            OUTDATASET( D50.EXAMPLE.ESDS1/ESD1UPPW ) -
            REUSE

   IF LASTCC = 0 -
      THEN -

      PRINT    INDATASET( D50.EXAMPLE.ESDS1 )
/*
```

Explanation of Job Control Language Statements

1. The INDSET4 DD statement describes the nonVSAM data set to be copied into the VSAM data set EXAMPLE.KSDS2.

The first REPRO command causes records in the VSAM key-sequenced data set to be replaced with input from a nonVSAM data set.

1. The INFILE parameter is required and identifies the data set containing the source data. The *dname* of the DD statement for this data set must be identical to this name.
2. The OUTDATASET parameter is required and identifies the target data set. Access Method Services will dynamically allocate the data set. Since the cluster is not password protected, no password is required.
3. The REPLACE parameter causes a record in the output data set having the same key as a record in the input data set to be replaced. Records in the input data set whose key is not already contained in the output data set will be inserted in the output data set.

If the REPRO operation was successfully executed, then the contents of the VSAM key-sequenced data set just changed are printed.

1. The INDATASET parameter is required and identifies the data set to be printed. Access Method Services will dynamically allocate the data set. Since the cluster is not password protected, no password is required.
2. The FROMKEY and TOKEY parameters specify the keys at which printing is to begin and end, respectively.

The second REPRO command causes the VSAM entry-sequenced data set to be loaded from the VSAM relative-record data set.

1. The INDATASET parameter is required and identifies the source data set. Access Method Services will dynamically allocate the data set Since the cluster is unprotected, no password is required.
2. The OUTDATASET parameter is required and identifies the target data set. Access Method Services will dynamically allocate the data set. The update or higher-level password is required to load the data set.
3. The REUSE parameter specifies that any records already in the entry-sequenced data set output are to be overwritten since the entry-sequenced data set was defined with the REUSE attribute.

If the REPRO operation was successfully executed, then the entire contents of the reloaded VSAM entry-sequenced data set are printed.

- The INDATASET parameter is required and identifies the data set to be printed. Access Method Services will dynamically allocate the data set. Since no read password exists for this data set, no password is required.

Example 7: Alter the Cataloged Attributes of VSAM Data Sets

This example shows how the cataloged attributes of two VSAM data sets are modified. Each ALTER command is followed by a LISTCAT command, which will execute only if its previous ALTER command completed successfully. The LISTCAT command prints the updated catalog entry.

This example depends on the successful completion of Example 3, which defined the two VSAM data sets whose attributes are being altered.

```
//ALTER    JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
     ALTER -
          D40.EXAMPLE.KSDS.DATA -
          FREESPACE(10 10)
     IF LASTCC = 0 -
        THEN -
        LISTCAT -
            ENTRIES(D40.EXAMPLE.KSDS.DATA/KSD1PSWD) -
            ALL
     ALTER -
          D50.EXAMPLE.ESDS1/ESD1MRPW -
          MASTERPW(ESD1PWMR) -
          CONTROLPW(ESD1PWCT) -
          UPDATEPW(ESD1PWUP) -
          READPW(ESD1PWRD) -
          AUTHORIZATION(ESD1AUTH)
     IF LASTCC = 0 -
        THEN -
        LISTCAT -
            ENTRIES(D50.EXAMPLE.ESDS1/ESD1PWMR) -
            CLUSTER -
            ALL
/*
```

Explanation of Commands

The first ALTER command shows how a data set's space management attributes are "tuned" for optimum performance.

The data component, D40.EXAMPLE.KSDS.DATA, of a key-sequenced VSAM data set,D40.EXAMPLE.KSDS1, was defined with 40 percent freespace in both control intervals and control areas. Now that data records have been loaded into the data set, its freespace attributes no longer appear to require 40 percent freespace. It is now desirable to have ten percent freespace in both control intervals and control areas.

- D40.EXAMPLE.KSDS.DATA names the entry whose attributes are to be altered with this command. No password is required since the data object is not password protected.
- The FREESPACE parameter respecifies percentages of freespace that apply to the data component.

The ALTER command is followed by a modal command that examines the condition code set when the ALTER command completes. If the ALTER command completes successfully, the LISTCAT command that immediately follows it prints the changed catalog entry.

- The ENTRIES and ALL parameters explicitly name the entry to be listed and specify that the entire entry is to be listed. Since all information about the data component is to be listed, some information in the associated cluster must be accessed; this requires the cluster's read password.

The second ALTER command shows how a data set's passwords can be modified. You could use the same technique to provide passwords for an existing VSAM object that doesn't have passwords.

- D50.EXAMPLE.ESDS1 names the entry whose attributes are to be altered with this command. The entry's master password, ESD1MRPW, is supplied to allow the command to alter the entry's passwords.
- The MASTERPW, CONTROLPW, UPDATEPW, and READPW parameters respecify passwords for the data set.
- The AUTHORIZATION parameter specifies the name of a user security verification routine that is to be called for additional security checking when the data set is opened.

If the ALTER command was successful, the LISTCAT command lists all of the cluster entry.

Example 8: Creating an Alternate Index and Its Path

This example defines an alternate index over a previously loaded VSAM key-sequenced base cluster, defines a path over the alternate index to provide a means for processing the base through the alternate index, and builds the alternate index. The alternate index, path, and base cluster must all be defined in the same catalog, in this case, the master catalog.

This example depends on the successful completion of Examples 4, 5, and 6 for the existence of the nonempty VSAM base cluster EXAMPLE.KSDS2.

```
//MAKEAIX  JOB    ...
//STEP1    EXEC   PGM=IDCAMS
//SYSPRINT DD     SYSOUT=A
//IDCUT1   DD     DSNAME=SORT.WORK.ONE,DISP=OLD,
//      AMP='AMORG',VOL=SER=VSER01,UNIT=2314
//IDCUT2   DD     DSNAME=SORT.WORK.TWO,DISP=OLD,
//      AMP='AMORG',VOL=SER=VSER01,UNIT=2314
//SYSIN    DD     *

     DEFINE ALTERNATEINDEX -
            (NAME(EXAMPLE.AIX) -
             RELATE(EXAMPLE.KSDS2) -
             MASTERPW(AIXMRPW) -
             UPDATEPW(AIXUPPW) -
             KEYS(3 0) -
             RECORDSIZE(40 50) -
             VOLUMES(VSER04) -
             CYLINDERS(3 1) -
             FILE(VOL4) -
             NONUNIQUEKEY -
             UPGRADE ) -
          CATALOG(AMASTCAT/MCATUPPW)

     DEFINE PATH -
            (NAME(EXAMPLE.PATH) -
             PATHENTRY(EXAMPLE.AIX/AIXMRPW) -
             READPW(PATHRDPW) -
             FILE(VOL4) ) -
          CATALOG(AMASTCAT/MCATUPPW)

     BLDINDEX INDATASET(EXAMPLE.KSDS2) -
              OUTDATASET(EXAMPLE.AIX/AIXUPPW) -
              CATALOG(AMASTCAT/MCATMRPW)

     PRINT INDATASET(EXAMPLE.PATH/PATHRDPW)
/*
```

Explanation of Job Control Language Statements

- The IDCUT1 and IDCUT2 DD statements describe the *dsnames* and a volume containing VSAM data space to be made available to BLDINDEX for defining and using two sort work data sets in the event an external sort is performed. This data space will not be used by BLDINDEX if enough virtual storage is available to perform an internal sort.

Explanation of Commands

The first DEFINE command creates a VSAM alternate index over the base cluster EXAMPLE.KSDS2.

1. The NAME parameter is required and names the object being defined.
2. The RELATE parameter is required and specifies the name of the base cluster over which the alternate index is defined.

3. The MASTERPW and UPDATEPW parameters specify the master and update passwords, respectively, for the alternate index.
4. The KEYS parameter specifies the length of the alternate key and its offset in the base cluster record.
5. The RECORDSIZE parameter specifies the length of the alternate index record. It must be large enough to contain the prime keys for all occurrences of any one alternate key since the alternate index is being defined with the NONUNIQUEKEY attribute.
6. The VOLUMES parameter is required and specifies the volume containing the alternate index EXAMPLE.AIX.
7. The CYLINDERS parameter specifies the amount of space to be allocated to the alternate index. A space parameter is required.
8. The FILE parameter for the base cluster's volume is required as the alternate index is being defined in a recoverable catalog.
9. The NONUNIQUEKEY parameter specifies that the base cluster can contain multiple occurrences of any one alternate key.
10. The UPGRADE parameter specifies that the alternate index is to reflect all changes made to the base cluster records, for example, additions or deletions of base cluster records.
11. The CATALOG parameter is required since the master catalog is password protected. It specifies the name of the master catalog and its update password which is required to define into a protected catalog.

The second DEFINE command defines a path over the alternate index. After the alternate index has been built, opening with the path name will cause processing of the base cluster via the alternate index.

1. The NAME parameter is required and names the object being defined.
2. The PATHENTRY parameter is required and specifies the name of the alternate index over which the path is defined and its master password.
3. The READPW parameter specifies a read password for the path; it will be propagated to master password level.
4. The FILE parameter is required as the path is being defined in a recoverable catalog.
5. The CATALOG parameter is required since the master catalog is password protected. It specifies the name of the master catalog and its update password which is required to define into a protected catalog.

The BLDINDEX command builds an alternate index. The assumption is made that enough virtual storage will be available to perform an internal sort. However, note that DD statements with the default *dnames* of IDCUT1 and IDCUT2 have been provided for two external sort work data sets in the event that the assumption is incorrect and an external sort must be performed.

1. The INDATASET parameter is required and identifies the base cluster. Access Method Services will dynamically allocate the base cluster. The base cluster is not password protected.
2. The OUTDATASET parameter is required and identifies the alternate index. Access Method Services will dyanmically allocate the alternate index. The update or higher-level password of the alternate index is required.

3. The CATALOG parameter specifies the name of the master catalog. If it is necessary for BLDINDEX to use external sort work data sets, they will be defined in and deleted from the master catalog. The master password will permit these actions.

The PRINT command causes the base cluster to be printed by means of the alternate key using the path defined to create this relationship.

- The INDATASET parameter is required and identifies the path object. Access Method Services will dynamically allocate the path. The read password of the path is required.

Example 9: Exporting a Base Cluster and Its Alternate Index

This example shows various methods of exporting data sets to provide backup and portability.

The example depends on the successful completion of previous examples for the existance of the various objects to be exported.

```
//EXPORT   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//RECEIVE  DD   DSNAME=PORTABLE.TAPE1,LABEL(1,SL),
//  UNIT=2400-3,DISP=(NEW,PASS),VOL=SER=TAPE01,
//  DCB=(DEN=3,BLKSIZE=6000)
//SYSIN    DD   *
     EXPORT -
          D40.EXAMPLE.KSDS1/KSD1PSWD -
          PURGE -
          OUTFILE(RECEIVE)
//STEP2    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//RECEIVE  DD   DSNAME=PORTABLE.TAPE2,LABEL=(2,SL),
//  UNIT=2400-3,DISP=(NEW,PASS),VOL=SER=TAPE01,
//  DCB=DEN=3
//SYSIN    DD   *
     EXPORT -
          D50.EXAMPLE.ESDS1/ESD1PWMR -
          OUTFILE(RECEIVE) -
          TEMPORARY -
          INHIBITSOURCE -
          INHIBITTARGET
//STEP3    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//RECEIVE  DD   DSNAME=PORTABLE.TAPE3,LABEL=(3,SL),
//  DISP=(NEW,PASS),VOL=SER=TAPE01,UNIT=2400-3,DCB=DEN=3
//SYSIN    DD    *
     EXPORT -
          EXAMPLE.AIX/MCATMRPW -
          OUTFILE(RECEIVE)
//STEP4    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//RECEIVE  DD   DSNAME=PORTABLE.TAPE4,LABEL=(4,SL),
//     DISP=(NEW,PASS),VOL=SER=TAPE01,UNIT=2400-3,
//     DCB=DEN=3
//SYSIN    DD    *
     EXPORT -
          EXAMPLE.KSDS2/MCATMRPW -
          OUTFILE(RECEIVE)
/*
```

Explanation of Job Control Language Statements

1. The RECEIVE DD statements describe the portable data sets. The record format (VBS) and logical record length are set by EXPORT. The blocksize is set by EXPORT as 2048 except where overridden as in step 1.

Explanation of Commands

The first EXPORT command causes a key-sequenced VSAM data set to be exported from a user catalog. When it has been exported to a magnetic tape file, the key-sequenced data set is deleted from the user catalog.

Note: When an object is exported, the record format of its records on the portable file are "VBS," and the EXPORT process determines the appropriate record size. However, the RECEIVE DD statement specifies a blocksize (BLKSIZE=6000) to override the blocksize used by the EXPORT process (2048 bytes) and to improve performance.

1. D40.EXAMPLE.KSDS1 names the key-sequenced VSAM data set being exported. Its password, KSD1PSWD, is also supplied (when the data set was defined, its read password propagated upward and all passwords for the data set are KSD1PSWD). Access Method Services will dynamically allocate the cluster.
2. The PURGE parameter is required, because the data set was defined with a retention period of 365 days. The data set cannot be exported permanently (that is, deleted from the catalog after its copy is made in the portable file) unless the PURGE parameter is specified to override its cataloged retention period.
3. The OUTFILE parameter is required and names the DD statement that describes and allocates the first file on a magnetic tape reel.

The second EXPORT command causes an entry-sequenced VSAM data set to be exported from a user catalog. When it has been exported to a magnetic tape file, the data set's entry in the user catalog is marked "temporary export" and "inhibit update," which prevents the data set from being modified in any way. The user's program can only read the data set's records. In addition, when the data set's copy on magnetic tape is imported into another system catalog, the data set's entry in the new, or "target," catalog is marked "inhibit update."

1. D50.EXAMPLE.ESDS1 names the entry-sequenced data set being exported. Its master password, ESD1PWMR, is also supplied. (Example 7 showed how the data set's passwords were changed.) Access Method Services will dynamically allocate the cluster.
2. The OUTFILE parameter is required and names the DD statement that describes and allocates the second file on a magnetic tape reel.
3. The TEMPORARY parameter specifies that the data set is not to be deleted from the catalog when it is exported.
4. The INHIBITSOURCE parameter specifies that the data set that remains in the "source" catalog and system is not to be updated or modified.
5. The INHIBITTARGET parameter specifies that the data set's exported copy is not to be updated or modified when it has been imported into the "target" catalog and system.

The third and fourth EXPORT commands cause the alternate index and base cluster to be exported from the master catalog. Any paths defined over either object will be exported with their PATHENTRY object. Since the export is permanent, both the base cluster and alternate index will be deleted from the catalog. The alternate index must be exported first, because the delete of the base cluster will cause deletion of all objects defined over it.

The third EXPORT command causes an alternate index to be exported from the master catalog.

1. The name of the alternate index being exported is required. A master password is required for the deletion and to allow VSAM locates against both the alternate index and path to obtain the catalog information (including passwords) to be exported. The master password of the catalog covers all requirements. Access Method Services will dynamically allocate the alternate index; this will also provide access to the catalog recovery area.
2. The OUTFILE parameter is required and names the DD statement that describes and allocates the third file on a magnetic tape reel.

The fourth EXPORT command causes a base cluster to be exported from the master catalog.

1. The name of the base cluster being exported is required. Since the cluster level is not protected, no password would be required for the deletion. However, a password is required for the VSAM locates against the data and index components to obtain the catalog information (including passwords) to be exported. Since only one password can be supplied, it must be that of the master catalog. Access Method Services will dynamically allocate the cluster; this will also provide access to the catalog recovery area.
2. The OUTFILE parameter is required and names the DD statement that describes and allocates the fourth file on a magnetic tape reel.

Example 10: Importing a Base Cluster and Its Alternate Index

This example shows various methods of importing data sets.

This example depends on the successful completion of Example 9, which created a portable magnetic tape that contains a copy of each VSAM data set to be imported.

```
//IMPORT   JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SOURCE   DD    DSNAME=PORTABLE.TAPE4,LABEL=(4,SL),
//  DISP=(OLD,PASS),VOL=SER=TAPE01,UNIT=2400-3,DCB=DEN=3
//SYSIN    DD    *

     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(EXAMPLE.KSDS2) -
          CATALOG(AMASTCAT/MCATUPPW)

//STEP2    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SOURCE   DD    DSNAME=PORTABLE.TAPE3,LABEL=(3,SL),
//  DISP=(OLD,PASS),VOL=SER=TAPE01,UNIT=2400-3,DCB=DEN=3
//SYSIN    DD    *

     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(EXAMPLE.AIX) -
          CATALOG(AMASTCAT/MCATUPPW)
//STEP3    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SOURCE   DD   DSNAME=PORTABLE.TAPE2,LABEL=(2,SL),
//  UNIT=2400-3,DISP=(OLD,PASS),VOL=SER=TAPE01,
//  DCB=(DEN=3,LRECL=3004)
//SYSIN    DD   *

     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(D50.EXAMPLE.ESDS1) -
          CATALOG(D27UCAT2/UCATMRPW)

//STEP4    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SOURCE   DD   DSNAME=PORTABLE.TAPE1,LABEL=(1,SL),
//  UNIT=2400-3,DISP=(OLD,PASS),VOL=SER=TAPE01,
//  DCB=(DEN=3,LRECL=479,BLKSIZE=6000)
//SYSIN    DD   *

     IMPORT -
          INFILE(SOURCE) -
          OUTDATASET(D50.EXAMNEW.KSDS1) -
          CATALOG(D27UCAT2/UCATUPPW) -
          OBJECTS( -
             (D40.EXAMPLE.KSDS1 -
              VOLUMES(VSER03) -
              NEWNAME(D50.EXAMNEW.KSDS1) ) -
             (D40.EXAMPLE.KSDS.DATA -
              NEWNAME(D50.EXAMNEW.KSDS.DATA) ) -
             (D40.EXAMPLE.KSDS.INDEX -
              NEWNAME(D50.EXAMNEW.KSDS.INDEX) ) )
/*
```

Explanation of Job Control Language Statements

- The SOURCE DD statements describe the portable data sets. For standard label tapes, the DCB parameters of LRECL and BLKSIZE need not be specified. However, the example shows parameters which would be required for a non-labeled tape. Unless overridden by DCB

parameters, the EXPORT command sets the blocksize to 2048 and the record length to blocksize minus 4.

Explanation of Commands

The first and second IMPORT commands will import the base cluster and alternate index exported in Example 9 into the master catalog. The importation causes each component to be newly defined. Since the alternate index cannot be defined until the base cluster has been defined, the base cluster must be imported first. The importation will cause any paths over the objects which were exported to be redefined.

The first IMPORT command causes the base cluster to be imported into the master catalog.

1. The INFILE parameter is required and names the DD statement that describes and allocates the fourth file on the magnetic tape reel containing the portable data set.
2. The OUTDATASET parameter is required and identifies the data set being imported. Access Method Services dynamically allocates the data set after it has been defined; this will also provide access to the catalog recovery area.
3. The CATALOG parameter is required since the master catalog is password protected. It specifies the name of the master catalog and its update password which is required to define into a protected catalog.

The second IMPORT command causes an alternate index to be imported into the master catalog.

1. The INFILE parameter is required and names the DD statement that describes and allocates the third file on the magnetic tape reel containing the portable data set.
2. The OUTDATASET parameter is required and identifies the data set being imported. Access Method Services, dynamically allocates the alternate index after it has been defined; this will also provide access to the catalog recovery area.
3. The CATALOG parameter is required since the master catalog is password protected. It specifies the name of the master catalog and its update password which is required to define into a protected catalog.

The third IMPORT command causes the previously-exported entry-sequenced VSAM data set to be imported. The data set is imported into the catalog from which it was exported. The portable data set (that is, the copy being imported) replaces the copy that exists in user catalog D27UCAT2. The IMPORT process searches user catalog D27UCAT2 for the entry with an entryname of D50.EXAMPLE.ESDS1. It deletes that entry, then redefines a cluster entry using the catalog information obtained from the portable data set. Because the data set was exported with the TEMPORARY attribute (see the previous example), the IMPORT command doesn't need to supply volume information.

Note: The SOURCE DD statement specifies a recordsize (LRECL=3004) because the largest record in the portable data set is 3000 bytes (that is, LRECL = (largest-record size) + 4). Otherwise, the default recordsize, blocksize minus 4, would be erroneously used by the EXPORT command. In the two previous steps, the default recordsize was used.

1. The INFILE parameter is required and names the DD statement that describes and allocates the third file on a magnetic tape reel. This is the portable data set to be imported.
2. The OUTDATASET parameter is required and identifies the data set being imported. Access Method Services dynamically allocates the data set after it has been defined.
3. The CATALOG parameter is required since the catalog is protected. It names the catalog that is to contain the imported data set. The catalog's master password is supplied, and allows the IMPORT process to delete the existing entry in the catalog and redefine a new entry for the entry-sequenced VSAM data set.

The fourth IMPORT command causes the previously-exported key-sequenced VSAM data set, D40.EXAMPLE.KSDS1, to be imported from its copy (that is, the first portable data set on the magnetic tape reel) to a different user catalog than it was exported from, D27UCAT2. The IMPORT command renames the data set and each of its components, as specified with the NEWNAME parameters. The high-level qualifier (D50) of the new components name is the alias name of the user catalog, D27UCAT2.

Note: The SOURCE DD statement describes the portable data set created in STEP1 of the previous example. The blocksize parameter is included (even though it needn't be, because the tape has a standard label and the information is contained in the data set header label) to illustrate the fact that the information specified when the data set is imported is required to be the same as was specified when the data set was exported. The LRECL parameter is not required, because the maximum record size is 475 bytes and the default (blocksize minus 4) is adequate. However, by specifying a recordsize, the default is overridden and virtual storage is more efficiently used. When recordsize is specified, it is the largest-record size + 4.

1. The INFILE parameter is required and names the DD statement that describes and allocates the portable data set containing the to-be-imported VSAM data set.
2. The OUTDATASET parameter is required and identifies the renamed data set. Access Method Services dynamically allocates the data set after it has been defined.
3. The CATALOG parameter is required since the catalog is protected. It names the catalog that is to contain the imported data set's entry. The catalog's update password is supplied and allows the data set to be imported into the catalog.
4. The OBJECTS parameter identifies the volume that is to contain the imported data set and specifies new names for each of the data set's components. The OBJECTS parameter identifies each entry of the imported data set with its original entryname, then specifies information that is to replace the information found in the portable data set's imported catalog entries.

Example 11: Deleting a VSAM Catalog and Its Cataloged Objects

This example deletes all entries defined in the previous examples and deletes the user catalogs which were defined. The result of this job is the removal of all data sets and catalogs defined in this set of examples.

```
//DELETE   JOB   ...
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
   /* DELETE THE NONVSAM DATA SET EXAMPLE.NONVSAM1 */
   DELETE  EXAMPLE.NONVSAM1 -
           NOSCRATCH -
           NONVSAM -
           CATALOG(D27UCAT1)
   /* DELETE VSAM DATA SETS CATALOGED IN */
   /* USER CATALOGS D27UCAT1 AND D27UCAT2 */
   DELETE  (D50.EXAMPLE.ESDS1/ESD1PWMR -
           D50.EXAMNEW.KSDS1/KSD1PSWD -
           D40.MYDATA) -
           PURGE -
           CLUSTER
   /* DELETE THE ENTRY-SEQUENCED VSAM */
   /*   DATA SET EXAMPLE.ESDS2        */
   DELETE  EXAMPLE.ESDS2/ESD2MRPW -
           PURGE -
           CATALOG(D27UCAT2)
   /* DELETE THE KEY-SEQUENCED VSAM */
   /*   CLUSTER EXAMPLE.KSDS2       */
   DELETE  EXAMPLE.KSDS2 -
           CATALOG(AMASTCAT/MCATMRPW)
   /* DELETE THE RELATIVE-RECORD VSAM */
   /*   CLUSTER EXAMPLE.RRDS1
   DELETE  EXAMPLE.RRDS1 -
           CLUSTER
   /* DELETE VSAM DATA SPACES FROM VOLUME */
   /*   VSER04 AND, IF THE VOLUME DOESN'T */
   /*   CONTAIN VSAM OBJECTS, DELETE THE  */
   /*   VOLUME'S ENTRY IN AMASTCAT        */
   DELETE  VSER04 -
           SPACE -
           CATALOG(AMASTCAT/MCATUPPW)
   /* DELETE THE VSAM USER CATALOG D27UCAT1 */
   DELETE  D27UCAT1/UCATMRPW -
           USERCATALOG -
           PURGE
   /* DELETE THE VSAM USER CATALOG D27UCAT2 */
   DELETE  D27UCAT2/UCATMRPW -
           USERCATALOG -
           PURGE
/*
```

The first DELETE command deletes the nonVSAM data set named EXAMPLE.NONVSAM1.

1. The name of the nonVSAM data set is required.
2. The NOSCRATCH parameter specifies that the VTOC entry of the object being deleted is not to be removed.
3. The NONVSAM parameter insures that the entry being deleted is a nonVSAM data set.
4. The CATALOG parameter is required in this example to identify the catalog that describes the data set to be deleted.

The second DELETE command deletes VSAM data sets cataloged in two user catalog s, D27UCAT1 and D27UCAT2.

1. The names of the VSAM data sets are specified; in addition the master passwords of two protected data sets are required. Since the high-level qualifier of the three data set names is the alias of the catalog that describes the data sets, the CATALOG parameter is not required to direct the catalog search.
2. The PURGE parameter causes the entries to be deleted without regard for the retention period.
3. The CLUSTER parameter insures that the catalog object being deleted is a VSAM data set.

The third DELETE command deletes the entry-sequenced VSAM data set EXAMPLE.ESDS2 from the user catalog, D27UCAT2.

1. The name of the entry-sequenced data set, EXAMPLE.ESD2, and its master password, ESD2MRPW, are required to identify the object to be deleted.
2. The PURGE parameter causes the entry to be deleted without regard to its retention period.
3. The CATALOG parameter is required in this example to identify the catalog that describes the data set to be deleted.

The fourth DELETE command deletes the key-sequenced VSAM data set EXAMPLE.KSDS2 from the master catalog. Because this data set is a base cluster, its related alternate index, EXAMPLE.AIX, and its path, EXAMPLE.PATH, are also automatically deleted from the master catalog.

1. The name of the key-sequenced VSAM data set, EXAMPLE.KSDS2, is required to identify the object to be deleted.
2. The CATALOG parameter is required, because the delete process will result in more than one protected object being deleted. It names the catalog that contains the catalog entry of each object to be deleted and supplies the catalog's master password, which allows each object in the catalog to be deleted.

The fifth DELETE command deletes the relative-record VSAM data set EXAMPLE.RRDS1 from the master catalog.

1. The name of the relative-record VSAM data set, EXAMPLE.RRDS1, is required to identify the object to be deleted. Because the data set was created without passwords, no password is required to delete it.

2. The CLUSTER parameter specifies that a cluster is being deleted.

The sixth DELETE command deletes each empty VSAM data space from volume VSER04. Because the volume no longer contains any VSAM objects, the DELETE command also removes the volume's entry from the master catalog and removes the master catalog's ownership of the volume. Access Method Services will dynamically allocate the volume, since the VTOC must be accessed. The volume should be mounted permanently RESIDENT or RESERVED to assure successful dynamic allocation.

Note: If VSAM data sets remain on a volume, the DELETE SPACE command only frees unused VSAM space on the volume.

1. The name (that is, volume serial number) of the volume, VSER04, is required to identify the object to be deleted.
2. The SPACE parameter is required to specify that all data spaces on the volume are to be deleted and, if the volume doesn't contain VSAM objects, that the volume's entry is to be deleted from the catalog.
3. The CATALOG parameter is required and names the catalog that owns the volume. The catalog's update password, MCATUPPW, allows the DELETE command to delete a volume entry or VSAM data space.

The seventh DELETE command deletes the VSAM user catalog D27UCAT1.

1. The name of the catalog and its master password are required to identify the object to be deleted.
2. The USERCATALOG parameter is required to specify that a user catalog is being deleted.
3. The PURGE parameter causes the entry to be deleted without regard to its retention period.

The eighth DELETE command deletes the VSAM user catalog D27UCAT2.

1. The name of the user catalog and its master password are required.
2. The USERCATALOG parameter is required to specify that the object being deleted is a user catalog.
3. The PURGE parameter causes the entry to be deleted without regard for the retention period.

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 or a generic entryname list was 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
HIS	History Group
PRT	Protection Group
STA	Statistics Group
VLS	Volumes Group
VOL	Volume Entry, Special Fields for

Alias Entry Keywords

ASSOCIATIONS (ASN)
entryname (HIS)
HISTORY (HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)

Alternate-Index Entry Keywords

ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
ATTRIBUTES (ATT)
CLUSTER (ASN)
CODE (PRT)
CONTROLPW (PRT)
DATA (ASN)
entryname (HIS)
HISTORY (HIS)
  CREATION (HIS)
  EXPIRATION (HIS)
  OWNER-IDENT(HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)
INDEX (ASN)
MASTERPW (PRT)
NOUPGRADE (ATT)
PATH (ASN)
PROTECTION (PRT)
RACF (PRT)
READPW (PRT)
UPDATEPW (PRT)
UPGRADE (ATT)
USAR (PRT)
USVR (PRT)

Cluster Entry Keywords

AIX (ASN)
ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
CODE (PRT)
CONTROLPW (PRT)
DATA (ASN)
entryname (HIS)
HISTORY (HIS)
  CREATION (HIS)
  EXPIRATION (HIS)
  OWNER-IDENT (HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)
INDEX (ASN)
MASTERPW (PRT)
PATH (ASN)
PROTECTION (PRT)
RACF (PRT)
READPW (PRT)
UPDATEPW (PRT)
USAR (PRT)
USVR (PRT)

Data Entry Keywords

ALLOCATION (ALC)
AIX (ASN)
ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
ATTRIBUTES (ATT)
AVGLRECL (ATT)
AXRKP (ATT)
BIND (ATT)
BUFSPACE (ATT)
BYTES/TRACK (VLS)
CI/CA (ATT)
CISIZE (ATT)
CLUSTER (ASN)
CODE (PRT)
CONTROLPW (PRT)
CYLFAULT (ATT)
DEVTYPE (VLS)
DSTGWAIT (ATT)
entryname (HIS)
ERASE (ATT)
EXCPEXIT (ATT)
EXTENT-NUMBER (VLS)
EXTENT-TYPE (VLS)
EXTENTS (VLS)
  HIGH-CCHH (VLS)
  HIGH-RBA (VLS)
  LOW-CCHH (VLS)
  LOW-RBA (VLS)
  TRACKS (VLS)
FREESPACE-%CI (STA)
FREESPACE-%CA (STA)
FREESPC-BYTES (STA)
HIGH-KEY (VLS)
HI-KEY-RBA (VLS)
HISTORY (HIS)
  CREATION (HIS)
  EXPIRATION (HIS)
  OWNER-IDENT (HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)
IMBED (ATT)
INDEX (ASN)
INH-UPDATE (ATT)
INDEXED (ATT)
KEYLEN (ATT)
LOW-KEY (VLS)
MASTERPW (PRT)
MAXLRECL (ATT)
MAXRECS (ATT)
NOERASE (ATT)
NOIMBED (ATT)
NONINDEXED (ATT)
NOREPLICAT (ATT)
NOREUSE (ATT)
NONSPANNED (ATT)
NOSWAP (ATT)
NOTRKOVFL (ATT)
EXCPS (STA)
EXTENTS (STA)
NONUNIQKEY (ATT)
NOTUSABLE (ATT)
NOWRITECHK (ATT)
NUMBERED (ATT)
ORDERED (ATT)
PGSPC (ASN)

PHYRECS/TRK (VLS)
PHYREC/SIZE (VLS)
PROTECTION (PRT)
RACF (PRT)
HI-ALLOC-RBA (ALC)
HI-USED-RBA (ALC)
HI-ALLOC-RBA (VLS)
HI-USED-RBA (VLS)
READPW (PRT)
RECOVERY (ATT)
REC-DELETED (STA)
REC-INSERTED (STA)
REC-RETRIEVED (STA)
REC-TOTAL (STA)
REC-UPDATED (STA)
RECORDS/CI (ATT)
REPLICATE (ATT)
RKP (ATT)
REUSE (ATT)
RECVABLE (ATT)
SHROPTNS (ATT)
SPACE-PRI (ALC)
SPACE-SEC (ALC)
SPACE-TYPE (ALC)
SPEED (ATT)
SPLITS-CA (STA)
SPLITS-CI (STA)
SPANNED (ATT)
STATISTICS (STA)
SUBALLOC (ATT)
SWAP (ATT)
SYSTEM-TIMESTAMP (STA)
TEMP-EXP (ATT)
TRACKS/CA (VLS)
TRKOVFL (ATT)
UNORDERED (ATT)
UNIQUE (ATT)
UNIQUEKEY (ATT)
UPDATEPW (PRT)
USAR (PRT)
USVR (PRT)
VOLFLAG (VLS)
VOLSER (VLS)
VOLUMES (VLS)
WRITECHECK (ATT)

Index Entry Keywords

ALLOCATION (ALC)
AIX (ASN)
ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
ATTRIBUTES (ATT)
AVGLRECL (ATT)
BIND (ATT)
BUFSPACE (ATT)
BYTES/TRACK (VLS)
CI/CA (ATT)
CISIZE (ATT)
CLUSTER (ASN)
CODE (PRT)
CONTROLPW (PRT)
CYLFAULT (ATT)
DEVTYPE (VLS)
DSTGWAIT (ATT)
entryname (HIS)
ERASE (ATT)
EXCPEXIT (ATT)
EXTENT-NUMBER (VLS)
EXTENT-TYPE (VLS)
EXTENTS (VLS)
  HIGH-CCHH (VLS)
  HIGH-RBA (VLS)
  LOW-CCHH (VLS)
  LOW-RBA (VLS)
  TRACKS (VLS)
FREESPACE-%CI (STA)
FREESPACE-%CA (STA)
FREESPC-BYTES (STA)
HIGH-KEY (VLS)
HISTORY (HIS)
  CREATION (HIS)
  EXPIRATION (HIS)
  OWNER-IDENT (HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)
IMBED (ATT)
INDEX (STA)
  ENTRIES/SECT (STA)
  HI-LEVEL-RBA (STA)
  LEVELS (STA)
  SEQ-SET-RBA (STA)
INH-UPDATE (ATT)
KEYLEN (ATT)
LOW-KEY (VLS)
MASTERPW (PRT)
MAXLRECL (ATT)
NOERASE (ATT)
NOIMBED (ATT)
NOREPLICAT (ATT)
NOREUSE (ATT)
EXCPS (STA)
EXTENTS (STA)
NOTUSABLE (ATT)
NOWRITECHK (ATT)
ORDERED (ATT)
PHYRECS/TRK (VLS)
PHYREC-SIZE (VLS)
PROTECTION (PRT)
RACF (PRT)
HI-ALLOC-RBA (ALC)
HI-USED-RBA (ALC)
HI-ALLOC-RBA (VLS)
HI-USED-RBA (VLS)
RECOVERY (ATT)
READPW (PRT)
REC-DELETED (STA)
REC-INSERTED (STA)
REC-RETRIEVED (STA)
REC-TOTAL (STA)
REC-UPDATED (STA)
REPLICATE (ATT)
RKP (ATT)
REUSE (ATT)
SHROPTNS (ATT)
SPACE-PRI (ALC)
SPACE-SEC (ALC)
SPACE-TYPE (ALC)
SPEED (ATT)
SPLITS-CA (STA)
SPLITS-CI (STA)

STATISTICS (STA)
SUBALLOC (ATT)
SYSTEM-TIMESTAMP (STA)
TEMP-EXP (ATT)
TRACKS/CA (VLS)
UNORDERED (ATT)
UNIQUE (ATT)
UPDATEPW (PRT)
USAR (PRT)
USVR (PRT)
VOLFLAG (VLS)
VOLSER (VLS)
VOLUME (VLS)
WRITECHECK (ATT)

Generation Data Group Base Entry Keywords

ASSOCIATIONS (ASN)
ATTRIBUTES (GDG)
  EMPTY (GDG)
  LIMIT (GDG)
  NOEMPTY (GDG)
  NOSCRATCH (GDG)
  SCRATCH (GDG)
entryname (HIS)
HISTORY (HIS)
  CREATION (HIS)
  EXPIRATION (HIS)
  OWNER-IDENT (HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)
NONVSAM (ASN)

NonVSAM Entry Keywords

ALIAS (ASN)
ASSOCIATIONS (ASN)
DEVTYPE(VLS)
entryname (HIS)
FSEQN (NVS)
HISTORY (HIS)
  CREATION (HIS)
  EXPIRATION (HIS)
  OWNER-IDENT (HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)
VOLSER(VLS)

Page Space Entry Keywords

ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
CODE (PRT)
CONTROLPW (PRT)
DATA (ASN)
entryname (HIS)
HISTORY (HIS)
  CREATION (HIS)
  EXPIRATION (HIS)
  OWNER-IDENT (HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)
INDEX (ASN)
MASTERPW (PRT)
PROTECTION (PRT)
| RACF (PRT)
READPW (PRT)
UPDATEPW (PRT)
USAR (PRT)
USVR (PRT)

Path Entry Keywords

AIX (ASN)
ASSOCIATIONS (ASN)
ATTEMPTS (PRT)
ATTRIBUTES (ATT)
CLUSTER (ASN)
CODE (PRT)
CONTROLPW (PRT)
DATA (ASN)
entryname (HIS)
HISTORY (HIS)
  CREATION (HIS)
  EXPIRATION (HIS)
  OWNER-IDENT (HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)
INDEX (ASN)
MASTERPW (PRT)
NOUPDATE (ATT)
PROTECTION (PRT)
| RACF (PRT)
READPW (PRT)
UPDATE (ATT)
UPDATEPW (PRT)
USAR (PRT)
USVR (PRT)

User Catalog Entry Keywords

ALIAS (ASN)
ASSOCIATIONS (ASN)
DEVTYPE(VLS)
entryname (HIS)
HISTORY (HIS)
  RCVY-CI (HIS)
  RCVY-DEVT (HIS)
  RCVY-VOL (HIS)
  RELEASE (HIS)
VOLFLAG (VLS)
VOLSER (VLS)

Volume Entry Keywords

ATTRIBUTES (DSP)
  AUTOMATIC (DSP)
  EXPLICIT (DSP)
  MASTERCAT (DSP)
  SUBALLOC (DSP)
  UNIQUE (DSP)
  USERCAT (DSP)
BYTES/TRK (VOL)
CHARACTERISTICS (VOL)
  CYLS/VOL (VOL)

DATASET-DIRECTORY (DSP)
- ATTRIBUTES (DSP)
- DSN (DSP)
- EXTENTS (DSP)

DATASETS (DSP)

DATASETS-ON-VOL (VOL)

DATASPACE (DSP)

DATASPCS-ON-VOL (VOL)

DEVTYPE (VLS)

EXTENT-DESCRIPTOR (DSP)
- BEG-CCHH (DSP)
- SPACE-MAP (DSP)
- TRACKS-TOTAL (DSP)
- TRACKS-USED (DSP)

EXTENTS (DSP)

FORMAT-1-DSCB (DSP)
- CCHHR
- TIMESTAMP (DSP)

MAX-PHYREC-SZ (VOL)

MAX-EXT/ALLOC (VOL)

HISTORY (HIS)
- RCVY-CI (HIS)
- RCVY-DEVT (HIS)
- RCVY-VOL (HIS)
- RELEASE (HIS)

SEC ALLOC (DSP)

TRKS/CYL (VOL)

TYPE (DSP)

volume serial number (HIS)

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
HIS	History 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.

HI-ALLOC-RBA — The highest RBA (plus 1) available within allocated space to store data.

HI-USED-RBA — The highest RBA (plus 1) within allocated space that actually contains data.

SPACE-PRI — 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.

SPACE-SEC — 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.

SPACE-TYPE — Indicates the unit of space allocation:

CANDIDATE—The volume is a candidate for future allocation.

CYLINDER—Cylinders

RECORD—Records

TRACK—Tracks

ASN: Associations Group

The fields in this group identify entries associated with the entry in which they appear:

- An alias entry points to:
  - Its associated nonVSAM data set entry.
  - A user catalog entry. (All alias entries for a nonVSAM data set entry are chained together, as are alias entries for a user catalog entry.)
- An alternate-index entry points to:
  - Its associated data and index entries.
  - Its base cluster's cluster entry.
  - Each associated path entry.
- An alternate-index's data entry points to:
  - Its associated alternate-index entry.
- An alternate-index's index entry points to:
  - Its associated alternate-index entry.
- A cluster entry points to:
  - Its associated data entry.
  - Each associated path entry.
  - For a key-sequenced cluster, its associated index entry.
  - For a cluster with alternate indexes, each associated alternate index entry.
- A cluster's data entry points to:
  - Its associated cluster entry.
- A cluster's index entry points to:
  - Its associated cluster entry.
- A generation data group base entry points to:
  - Its associated nonVSAM data set entries.
- A nonVSAM data set entry points to:
  - Its associated alias entry.
  - Its associated generation data group (for a G0000V00 nonVSAM).
- A page space entry points to:
  - Its associated data entry. (The page space is cataloged as an entry-sequenced cluster with a cluster entry and an associated data entry.)

- A path entry (that establishes the connection between a base cluster and an alternate index) points to:
  - Its associated alternate index entry, and the alternate index's associated data and index entries.
  - The data entry of its associated base cluster.
  - For a key-sequenced base cluster, the index entry of its associated base cluster.
- Path entry (that is an alias for a cluster entry) points to:
  - Its associated base cluster entry.
  - The data entry of its associated base cluster.
  - For a key-sequenced cluster, the index entry of its associated base cluster.
- A user catalog entry points to:
  - Its associated alias entry.

AIX — Identifies an alternate-index entry.

ALIAS — Identifies an alias entry.

CLUSTER — Identifies a cluster entry.

DATA — Identifies a data entry.

GDG — Identifies a genration data group (GDG) base entry.

INDEX — Identifies an index entry.

NONVSAM — Identifies a nonVSAM data set entry.

PGSPC — Identifies a page space entry.

PATH — Identifies a path 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.

AXRKP — Alternate Index Relative Key Position. The offset, from the beginning of the base cluster's data record, at which the alternate-key field begins.

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.

BUFSPACE — The minimum buffer space in virtual storage to be provided by a processing program.

CI/CA — The number of control intervals per control area.

CISIZE — The size of a control interval.

CYLFAULT — 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.

DSTGWAIT — 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.

EXCPEXIT — The name of the object's exception exit routine.

IMBED — The sequence-set index record is stored along with its associated data control area.

INH-UPDATE — 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.

INDEXED — 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.

MAXRECS — Identifies the highest relative-record number, for a relative-record data set.

NOERASE — Records are not to be erased (set to binary 0s) when deleted.

NOIMBED — The sequence-set index record is not stored along with its associated data control area.

NONINDEXED — The data component has no index—it is entry-sequenced.

NONSPANNED — Data records cannot span control intervals.

NONUNIQKEY — More than one data record in the base cluster can contain the same alternate-key value.

NOREPLICAT — Index records are not replicated.

NOREUSE — The data set cannot be reused.

NOSWAP — The page space is a conventional page space and cannot be used as a high speed swap data set.

NOTRKOVFL — The physical blocks of a pagespace data set cannot span a track boundary.

NOUPDATE — When the path is opened for processing, its associated base cluster and index are opened but the base cluster's upgrade set is not opened.

NOUPGRADE — The alternate index is not upgraded unless it is opened and being used to access the base cluster's data records.

NOTUSABLE — The entry is not usable because the catalog could not be correctly recovered.

NOWRITECHK — Write operations are not checked for correctness.

NUMBERED — The cluster is a relative-record data set.

ORDERED — Volumes are used for space allocation in the order they were specified when the cluster was defined.

RECOVERY — 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.

RECORDS/CI — Specifies the number of records, or slots, in each control interval of a relative-record data set.

RECVABLE — This attribute is set in the data entry of a recoverable catalog, and each of the catalog's volumes contains a catalog recovery area.

REPLICATE — Index records are replicated (that is, each is duplicated around a track of the index's direct-access device).

REUSE — The data set can be reused (that is, its contents are temporary and its high-used RBA is reset to 0 each time it is opened).

RKP — The relative key position—the displacement from the beginning of a data record to its key field.

SHROPTNS — *(n,m)* The numbers n and m identify the types of sharing permitted. See SHAREOPTIONS in the DEFINE CLUSTER section for more details.

SPANNED — Data records can be longer than control-interval length, and can cross, or span, control-interval boundaries.

SPEED — CLOSE is not issued until the data set has been loaded.

SUBALLOC — More than one VSAM cluster can share the data space. A VSAM catalog might also occupy the data space.

SWAP — The page space is a high speed swap data set used by Auxiliary Storage Management during a swap operation to store and retrieve the set of LSQA pages owned by an address space.

TEMP-EXP — The data component was temporarily exported.

TRKOVFL — The physical blocks of a pagespace data set can span a track boundary.

UNIQUE — Only one VSAM cluster or catalog can occupy the data space—the cluster or catalog is unique.

UNIQUEKEY — The alternate-key value identifies one, and only one, data record in the base cluster.

UNORDERED — Volumes specified when the cluster was defined can be used for space allocation in any order.

UPDATE — When the path is opened, the upgrade set's alternate indexes (associated with the path's base cluster) are also opened and are updated when the base cluster's contents change.

UPGRADE — When the alternate index's base cluster is opened, the alternate index is also opened and will be updated to reflect any changes to the base cluster's contents.

WRITECHECK — 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.

AUTOMATIC — 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.

EXPLICIT — The data space was created explicitly by a DEFINE

MASTERCATALOG or USERCATALOG command, or a DEFINE ALTERNATEINDEX or DEFINE CLUSTER command with the UNIQUE attribute specified.

MASTERCAT — The data space contains the master catalog.

SUBALLOC — The data space might contain several VSAM clusters.

USERCAT — The data space contains a user catalog.

UNIQUE — 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.

CANDIDATE—The volume is a candidate for storing the data set

(NULL)—The data set is stored (in whole or in part) in the data space

DSN — The data set name of the object that can be stored or is stored on the volume.

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 volume 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-TOTAL — Allocated to the data space altogether.

TRACKS-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.

CCHHR — 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.

SEC-ALLOC — 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:

CYLINDER—Cylinders

TRACK—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 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.

NOSCRATCH
Generation data sets are not to be scratched (see SCRATCH below) when uncataloged.

SCRATCH
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.

HIS: History Group

The fields in this group identify the object's owner, identify its catalog recovery volume and release level, 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.

RCVY-CI — If the entry is in a recoverable catalog, this field contains the control interval number in the catalog recovery area where a duplicate of the entry can be found.

RCVY-DEVT — If the entry is in a recoverable catalog, this field identifies the recovery volume's device type.

RCVY-VOL — If the entry is in a recoverable catalog, this field contains the recovery volume's serial number.

RELEASE — The release of VSAM under which the entry was created (not the same as the release number of OS/VS2):

1 = OS/VS2 Releases 3 and releases previous to Release 3

2 = OS/VS2 Release 3.6 and later releases

PRT: Protection Group

The fields in this group describe how the alternate index, cluster, data component, index component, or path defined by the entry is password protected or RACF protected. NULL or SUPPRESSED might be listed under password protection and YES or NO might be listed under RACF protection.

NULL indicates that the object defined by the entry has no passwords.

SUPP indicates that the master password of neither the catalog nor the entry was specified, so authority to list protection information is not granted.

RACF — Indicates whether or not the entry is protected via the Resource Access Control Facility.

YES — Entry is RACF protected.

NO — Entry is not RACF protected.

ATTEMPTS — Gives the number of times the console operator is allowed to attempt to enter a correct password.

CODE — Gives the code used to tell the console operator what alternate index, catalog, cluster, path, 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.

CONTROLPW — 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.

USAR — 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-%CI — Percentage of space to be left free in a control interval for subsequent processing.

FREESPACE-%CA — Percentage of control intervals to be left free in a control area for subsequent processing.

FREESPC-BYTES — 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/SECT — The number of entries in each section of entries in an index record.

HI-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.

SEQ-SET-RBA — The RBA (relative byte address) of the first sequence-set record. The sequence set might be separated from the index set by some quantity of RBA space.

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.

REC-DELETED — The number of records that have been deleted from the data or index component.

REC-INSERTED — The number of records added to the data or index component that have not been added to the end of the data set. It includes the number of records originally loaded.

REC-RETRIEVED — The number of records that have been retrieved from the data or index component, whether for update or not for update.

REC-TOTAL — The total number of records actually in the data or index component.

REC-UPDATED — The number of records that have been retrieved for update and rewritten.

SPLITS-CA — Control-area splits. Half the data records in a control area were written into a new control area and then were deleted from the old control area.

SPLITS-CI — Control-interval splits. Half the data records in a control interval were written into a new control interval and then

interval were written into a new control interval and then were deleted from the old control interval.

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 or relative-record 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 index 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/TRACK — The number of bytes that VSAM can write on a track.

DEVTYPE — The type of device to which the volume belongs.

EXTENT-NUMBER — The number of extents allocated for the data or index component on the volume.

EXTENT-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[1] — 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.

HI-KEY-RBA[1] — 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[1] — 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/TRK[1] — The number of physical records (of the size indicated under PHYRECS-SIZE) that VSAM can write on a track on the volume.

PHYREC-SIZE — The number of bytes that VSAM uses for a physical record in the data or index component.

HI-ALLOC-RBA — The highest RBA (plus 1) available within allocated space to store data component, its key-range, the index component, or the sequence set records of a key range.

HI-USED-RBA — The highest RBA (plus 1) within allocated space that actually contains data component, its key-range, the index component, or the sequence set records of a key range.

TRACKS/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.

OVERFLOW — 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 characteristics of the space that VSAM uses on the volume.

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.

BYTES/TRK — The number of bytes that VSAM can use on each track on the volume.

CYLS/VOL — The number of cylinders that VSAM can use on the volume.

DATASETS-ON-VOL — The number of VSAM clusters that reside, in whole or in part, on the volume.

DATASPCS-ON-VOL — The number of VSAM data spaces on the volume.

MAX-PHYREC-SZ — The size of the largest physical record that VSAM can write on the volume.

MAX-EXT/ALLOC — The maximum number of extents that can be

---

[1] Multiple keyranges might reside on a single volume—the volumes group is repeated for each such keyrange field.

allocated on the volume for a single data set.

TRKS/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 Jobs

The job control language (JCL) statements that can be used to list a catalog's entries are:

```
//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 statement contains user and accounting information required for your installation.

The EXEC statement 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 auxiliary 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=629) 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 previuosly:

```
//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 Access Method Services 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 16):

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, the entryname and type of each entry is listed (see Figure 17). The same listing would result if the NAMES parameter were specified.

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.

```
                        LISTING FROM CATALOG -- MJKCAT

      THE NUMBER OF ENTRIES PROCESSED WAS:
                AIX ------------------1
                ALIAS ----------------1
                CLUSTER --------------2
                DATA -----------------3
                GDG ------------------1
                INDEX ----------------3
                NONVSAM --------------2
                PAGESPACE ------------0
                PATH -----------------1
                SPACE ----------------1
                USERCATALOG ----------0
                TOTAL ----------------15

      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 16. Messages that Follow the Entry Listing

```
  /* A:   LIST ENTRYNAMES OF THE MASTER CATALOG */
     LISTCAT -
          CATALOG (MJKCAT)

                              LISTING FROM CATALOG -- MJKCAT
AIX ------------ MJK.ALT.INDEX1
   DATA ------- T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320
   INDEX ------ T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380
   PATH ------- MJK.AIX1.PATH
CLUSTER ------- MJK.CLUSTER1
   DATA ------- T7D503F0.VSAMDSET.DFD75223.T87A1369.T7D503F0
   INDEX ------ T7D550E0.VSAMDSET.DFD75223.T87A1369.T7D550E0
GDG BASE ------ MJK.GDG1
   NONVSAM ---- MJK.GDG1.G0001V00
ALIAS ---------- MJK.GDG1.ALIAS
NONVSAM ------- MJK.NONVSAM1
CLUSTER ------- MJKCAT
   DATA ------- VSAM.CATALOG.BASE.DATA.RECORD
   INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
VOLUME -------- 333001

                              LISTING FROM CATALOG -- MJKCAT
         THE NUMBER OF ENTRIES PROCESSED WAS:
                   AIX --------------------1
                   ALIAS ------------------1
                   CLUSTER ----------------2
                   DATA -------------------3
                   GDG --------------------1
                   INDEX ------------------3
                   NONVSAM ----------------2
                   PAGESPACE --------------0
                   PATH -------------------1
                   SPACE ------------------1
                   USERCATALOG ------------0
                   TOTAL -----------------15
         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. 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 and device type of each volume that contains part or all of the cataloged object are listed (see Figure 18).

```
  /* B:   LIST VOLUMES FOR SELECTED ENTRIES  */
     LISTCAT -
          VOLUME -
          CATALOG (MJKCAT)

                              LISTING FROM CATALOG -- MJKCAT
AIX ----------- MJK.ALT.INDEX1
      HISTORY
        OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000010'
        RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
    DATA ------- T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320
      HISTORY
        OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000011'
        RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
      VOLUMES
        VOLSER------------333001      DEVTYPE------X'30502009'
    INDEX ------ T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380
      HISTORY
        OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000012'
        RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
      VOLUMES
        VOLSER------------333001      DEVTYPE------X'30502009'
    PATH ------- MJK.AIX1.PATH
      HISTORY
        OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000014'
        RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
CLUSTER ------- MJK.CLUSTER1
      HISTORY
        OWNER-IDENT-----OWNCLUST      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000E'
        RELEASE----------------2      EXPIRATION--------75.323     RCVY-DEVT----X'30502009'
    DATA ------- T7D503F0.VSAMDSET.DFD75223.T87A1369.T7D503F0
      HISTORY
        OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000D'
        RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
      VOLUMES
        VOLSER------------333001      DEVTYPE------X'30502009'
    INDEX ------ T7D550E0.VSAMDSET.DFD75223.T87A1369.T7D550E0
      HISTORY
        OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000F'
        RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
      VOLUMES
        VOLSER------------333001      DEVTYPE------X'30502009'
GDG BASE ------ MJK.GDG1
      HISTORY
        OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000015'
        RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
    NONVSAM ---- MJK.GDG1.G0001V00
      HISTORY
        OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000016'
        RELEASE----------------2      EXPIRATION--------00.000     RCVY-VOL-----X'30502009'
      VOLUMES
        VOLSER------------333001      DEVTYPE------X'30502009'
        VOLSER------------333002      DEVTYPE------X'30502009'
NONVSAM ------- MJK.NONVSAM1
      HISTORY
        OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000018'
        RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
      VOLUMES
        VOLSER------------333001      DEVTYPE------X'30502009'
```

Figure 18 (Part 1 of 2). An Example of LISTCAT VOLUME Output

```
CLUSTER ------- MJKCAT
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223
       RELEASE----------------2      EXPIRATION--------99.999
   DATA ------- VSAM.CATALOG.BASE.DATA.RECORD
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------00.000
       RELEASE----------------2      EXPIRATION--------00.000
     VOLUMES
       VOLSER------------333001      DEVTYPE------X'30502009'
   INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------00.000
       RELEASE----------------2      EXPIRATION--------00.000
     VOLUMES
       VOLSER------------333001      DEVTYPE------X'30502009'
VOLUME -------- 333001
     HISTORY
       RELEASE----------------2      RCVY-VOL----------333001     RCVY-DEVT----X'30502009'     RCVY-CI--------X'000009'
     VOLUMES
       VOLSER------------333001      DEVTYPE------X'30502009'
                          LISTING FROM CATALOG -- MJKCAT
         THE NUMBER OF ENTRIES PROCESSED WAS:
                   AIX --------------------1
                   ALIAS ------------------0
                   CLUSTER ----------------2
                   DATA -------------------3
                   GDG --------------------1
                   INDEX ------------------3
                   NONVSAM ----------------2
                   PAGESPACE --------------0
                   PATH -------------------1
                   SPACE ------------------1
                   USERCATALOG ------------0
                   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 (Part 2 of 2). An Example of LISTCAT VOLUME Output

LISTCAT SPACE ALL Output Listing

When the LISTCAT command is specified with the SPACE and ALL parameters, all the information for each volume entry in the catalog is listed (see Figure 16). You can use this type of listing to determine how space on each cataloged volume is allocated to VSAM data spaces. You may 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(MJKCAT)

                              LISTING FROM CATALOG -- MJKCAT

VOLUME -------- 333001
     HISTORY
       RELEASE----------------2     RCVY-VOL----------333001     RCVY-DEVT----X'30502009'     RCVY-CI--------X'000009'
     CHARACTERISTICS
       BYTES/TRK----------13165     DEVTYPE------X'30502009'     MAX-PHYREC-SZ------13030     DATASETS-ON-VOL--------5
       TRKS/CYL--------------19     VOLUME-TIMESTAMP:            MAX-EXT/ALLOC----------5     DATASPCS-ON-VOL--------1
       CYLS/VOL-------------411          X'87A1390412233000'
     DATASPACE
       DATASETS---------------5     FORMAT-1-DSCB:               ATTRIBUTES:
       EXTENTS----------------1     CCHHR------X'0000000103'     SUBALLOC
       SEC-ALLOC--------------2     TIMESTAMP                    EXPLICIT
       TYPE------------CYLINDER          X'87A1355C90BB2000'     USERCAT
       EXTENT-DESCRIPTOR:
       TRACKS-TOTAL---------152     BEG-CCHH-----X'00010000'     SPACE-MAP-----------4157
       TRACKS-USED-----------65
       DATASET-DIRECTORY:
         DSN----MJKCAT                                           ATTRIBUTES--------(NULL)     EXTENTS----------------3
         DSN----T7D503F0.VSAMDSET.DFD75223.T87A1369.T7D503F0     ATTRIBUTES--------(NULL)     EXTENTS----------------1
         DSN----T7D550E0.VSAMDSET.DFD75223.T87A1369.T7D550E0     ATTRIBUTES--------(NULL)     EXTENTS----------------1
         DSN----T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320     ATTRIBUTES--------(NULL)     EXTENTS----------------1
         DSN----T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380     ATTRIBUTES--------(NULL)     EXTENTS----------------1
                              LISTING FROM CATALOG -- MJKCAT
         THE NUMBER OF ENTRIES PROCESSED WAS:
                   AIX -------------------0
                   ALIAS -----------------0
                   CLUSTER ---------------0
                   DATA ------------------0
                   GDG -------------------0
                   INDEX -----------------0
                   NONVSAM ---------------0
                   PAGESPACE -------------0
                   PATH ------------------0
                   SPACE -----------------1
                   USERCATALOG -----------0
                   TOTAL -----------------1
         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 SPACE ALL Output

LISTCAT ALL Output Listing

When you specify the LISTCAT command and include the ALL parameter, all the information for each catalog entry is listed (see Figure 17). This example illustrates the LISTCAT output for each type of catalog entry. You can use this type of listing to obtain all cataloged information (except password and security information) about each entry that is listed. When you want to list an entry's passwords, you must provide the catalog's master password (which results in listing the passwords of each password-protected entry) or each entry's master password.

Note: When ENTRIES is specified, you specify only those entrynames that identify catalog entries which aren't 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 parameter is 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 -
          CATALOG (MJKCAT)
                          LISTING FROM CATALOG -- MJKCAT

AIX ----------- MJK.ALT.INDEX1
     HISTORY
       OWNER-IDENT-------(NULL)     CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000010'
       RELEASE----------------2     EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
     PROTECTION-PSWD-----(NULL)     RACF----------------(No)
     ASSOCIATIONS
       DATA-----T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320
       INDEX----T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380
       CLUSTER--MJK.CLUSTER1
       PATH-----MJK.AIX1.PATH
     ATTRIBUTES
       UPGRADE

   DATA ------- T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320
     HISTORY
       OWNER-IDENT-------(NULL)     CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000011'
       RELEASE----------------2     EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
     PROTECTION-PSWD-----(NULL)     RACF----------------(No)
     ASSOCIATIONS
       AIX------MJK.ALT.INDEX1
     ATTRIBUTES
       KEYLEN-----------------5     AVGLRECL------------4086     BUFSPACE------------8704     CISIZE--------------4096
       RKP--------------------5     MAXLRECL-----------32600     EXCPEXIT----------(NULL)     CI/CA-----------------18
       AXRKP-----------------45
       SHROPTNS(1,3)   RECOVERY     SUBALLOC          NOERASE    INDEXED        NOWRITECHK    NOIMBED        NOREPLICAT
       UNORDERED        NOREUSE     SPANNED        NONUNIQKEY
     STATISTICS
       REC-TOTAL--------------0     SPLITS-CI--------------0     EXCPS------------------0
       REC-DELETED------------0     SPLITS-CA--------------0     EXTENTS----------------1
       REC-INSERTED-----------0     FREESPACE-%CI----------0     SYSTEM-TIMESTAMP:
       REC-UPDATED------------0     FREESPACE-%CA----------0          X'0000000000000000'
       REC-RETRIEVED----------0     FREESPC-BYTES------73728
     ALLOCATION
       SPACE-TYPE---------TRACK     HI-ALLOC-RBA-------73728
       SPACE-PRI--------------6     HI-USED-RBA------------0
       SPACE-SEC--------------6
     VOLUME
       VOLSER-----------333001      PHYREC-SIZE---------4096     HI-ALLOC-RBA-------73728     EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'     PHYRECS/TRK------------3     HI-USED-RBA------------0     EXTENT-TYPE--------X'40'
       VOLFLAG------------PRIME     TRACKS/CA--------------6
       EXTENTS:
       LOW-CCHH-----X'00040001'     LOW-RBA----------------0     TRACKS-----------------6
       HIGH-CCHH----X'00040006'     HIGH-RBA----------73727
```

Figure 20 (Part 1 of 5). An Example of LISTCAT ALL Output

```
INDEX ------ T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380
  HISTORY
    OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000012'
    RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
  PROTECTION-PSWD-----(NULL)      RACF----------------(No)
  ASSOCIATIONS
    AIX------MJK.ALT.INDEX1
  ATTRIBUTES
    KEYLEN-----------------5      AVGLRECL--------------0      BUFSPACE---------------0     CISIZE---------------512
    RKP--------------------5      MAXLRECL-------------505     EXCPEXIT----------(NULL)     CI/CA-----------------20
    SHROPTNS(1,3)   RECOVERY      SUBALLOC          NOERASE    NOWRITECHK        NOIMBED    NOREPLICAT      UNORDERED
    NOREUSE
  STATISTICS
    REC-TOTAL--------------0      SPLITS-CI--------------0     EXCPS------------------0     INDEX:
    REC-DELETED------------0      SPLITS-CA--------------0     EXTENTS----------------1     LEVELS-----------------0
    REC-INSERTED-----------0      FREESPACE-%CI----------0     SYSTEM-TIMESTAMP:            ENTRIES/SECT-----------4
    REC-UPDATED------------0      FREESPACE-%CA----------0          X'0000000000000000'     SEQ-SET-RBA------------0
    REC-RETRIEVED----------0      FREESPC-BYTES------10240                                  HI-LEVEL-RBA-----------0
  ALLOCATION
    SPACE-TYPE---------TRACK      HI-ALLOC-RBA-------10240
    SPACE-PRI--------------1      HI-USED-RBA------------0
    SPACE-SEC--------------1
  VOLUME
    VOLSER------------333001      PHYREC-SIZE----------512     HI-ALLOC-RBA-------10240     EXTENT-NUMBER----------1
    DEVTYPE------X'30502009'      PHYRECS/TRK-----------20     HI-USED-RBA------------0     EXTENT-TYPE--------X'40'
    VOLFLAG------------PRIME      TRACKS/CA--------------1
    EXTENTS:
    LOW-CCHH-----X'00040007'      LOW-RBA----------------0     TRACKS-----------------1
    HIGH-CCHH----X'00040007'      HIGH-RBA-----------10239

PATH ------- MJK.AIX1.PATH
  HISTORY
    OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000014'
    RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
  PROTECTION-PSWD-----(NULL)      RACF----------------(No)
  ASSOCIATIONS
    AIX------MJK.ALT.INDEX1
    DATA-----T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320
    INDEX----T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380
    DATA-----T7D503F0.VSAMDSET.DFD75223.T87A1369.T7D503F0
    INDEX----T7D550E0.VSAMDSET.DFD75223.T87A1369.T7D550E0
  ATTRIBUTES
    UPDATE

CLUSTER ------- MJK.CLUSTER1
  HISTORY
    OWNER-IDENT-----OWNCLUST      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000E'
    RELEASE----------------2      EXPIRATION--------75.323     RCVY-DEVT----X'30502009'
  PROTECTION
    MASTERPW----------MASTCL      UPDATEPW----------UPDCL      CODE-------------CODECL      RACF----------------(No)
    CONTROLPW---------CNTLCL      READPW------------READCL     ATTEMPTS---------------3     USVR--------------(NULL)
    USAR-------------(NONE)
  ASSOCIATIONS
    DATA-----T7D503F0.VSAMDSET.DFD75223.T87A1369.T7D503F0
    INDEX----T7D550E0.VSAMDSET.DFD75223.T87A1369.T7D550E0
    AIX------MJK.ALT.INDEX1

DATA ------- T7D503F0.VSAMDSET.DFD75223.T87A1369.T7D503F0
  HISTORY
    OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000D'
    RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
  PROTECTION-PSWD-----(NULL)      RACF----------------(No)
  ASSOCIATIONS
    CLUSTER--MJK.CLUSTER1
  ATTRIBUTES
    KEYLEN-----------------5      AVGLRECL--------------50     BUFSPACE-----------20992     CISIZE-------------10240
    RKP--------------------5      MAXLRECL--------------50     EXCPEXIT--------EXITCLUS     CI/CA------------------2
    SHROPTNS(1,3)   RECOVERY      SUBALLOC          NOERASE    INDEXED       NOWRITECHK     NOIMBED        NOREPLICAT
    UNORDERED        NOREUSE      NONSPANNED      CYLFAULT     DSTGWAIT
  STATISTICS
    REC-TOTAL--------------0      SPLITS-CI--------------0     EXCPS------------------0
    REC-DELETED------------0      SPLITS-CA--------------0     EXTENTS----------------1
    REC-INSERTED-----------0      FREESPACE-%CI----------15    SYSTEM-TIMESTAMP:
    REC-UPDATED------------0      FREESPACE-%CA----------20         X'0000000000000000'
    REC-RETRIEVED----------0      FREESPC-BYTES------20480
  ALLOCATION
    SPACE-TYPE---------TRACK      HI-ALLOC-RBA-------20480
    SPACE-PRI--------------2      HI-USED-RBA------------0
    SPACE-SEC--------------2
  VOLUME
    VOLSER------------333001      PHYREC-SIZE---------2048     HI-ALLOC-RBA-------20480     EXTENT-NUMBER----------1
    DEVTYPE------X'30502009'      PHYRECS/TRK------------6     HI-USED-RBA------------0     EXTENT-TYPE------- X'40'
    VOLFLAG------------PRIME      TRACKS/CA--------------2
    EXTENTS:
    LOW-CCH------X'00030011'      LOW-RBA----------------0     TRACKS-----------------2
    HIGH-CCHH----X'00030012'      HIGH-RBA-----------20479
```

Figure 20 (Part 2 of 5). An Example of LISTCAT ALL Output

```
   INDEX ------ T7D550E0.VSAMDSET.DFD75223.T87A1369.T7D550E0
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000F'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
     PROTECTION-PSWD-----(NULL)      RACF----------------(No)
     ASSOCIATIONS
       CLUSTER--MJK.CLUSTER1
     ATTRIBUTES
       KEYLEN-----------------5      AVGLRECL---------------0     BUFSPACE---------------0     CISIZE--------------512
       RKP--------------------5      MAXLRECL-----------505       EXCPEXIT--------EXITCLUS     CI/CA----------------20
       SHROPTNS(1,3)   RECOVERY      SUBALLOC          NOERASE    NOWRITECHK        NOIMBED    NOREPLICAT      UNORDERED
       NOREUSE          CYLFAULT     DSTGWAIT
     STATISTICS
       REC-TOTAL--------------0      SPLITS-CI--------------0     EXCPS------------------0     INDEX:
       REC-DELETED------------0      SPLITS-CA--------------0     EXTENTS----------------1     LEVELS-----------------0
       REC-INSERTED-----------0      FREESPACE-%CI----------0     SYSTEM-TIMESTAMP:            ENTRIES/SECT-----------1
       REC-UPDATED------------0      FREESPACE-%CA----------0          X'0000000000000000'     SEQ-SET-RBA------------0
       REC-RETRIEVED----------0      FREESPC-BYTES------10240                                  HI-LEVEL-RBA-----------0
     ALLOCATION
       SPACE-TYPE---------TRACK      HI-ALLOC-RBA-------10240
       SPACE-PRI----------------1    HI-USED-RBA------------0
       SPACE-SEC----------------1
     VOLUME
       VOLSER------------333001      PHYREC-SIZE----------512     HI-ALLOC-RBA-------10240     EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK-----------20     HI-USED-RBA------------0     EXTENT-TYPE--------X'40'
       VOLFLAG------------PRIME      TRACKS/CA--------------1
       EXTENTS:
       LOW-CCHH-----X'00040000'      LOW-RBA----------------0     TRACKS-----------------1
       HIGH-CCHH----X'00040000'      HIGH-RBA-----------10239

GDG BASE ------ MJK.GDG1
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000015'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
     ATTRIBUTES
       LIMIT----------------250      NOSCRATCH         NOEMPTY
     ASSOCIATIONS
       NONVSAM--MJK.GDG1.G0001V00

   NONVSAM ---- MJK.GDG1.G0001V00
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000016'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
     VOLUMES
       VOLSER------------333001      DEVTYPE------X'30502009'     FSEQN------------------0
       VOLSER------------333002      DEVTYPE------X'30502009'     FSEQN------------------0
     ASSOCIATIONS
       GDG------MJK.GDG1
       ALIAS----MJK.GDG1.ALIAS

ALIAS --------- MJK.GDG1.ALIAS
     HISTORY
       RELEASE----------------2      RCVY-VOL----------333001     RCVY-DEVT----X'30502009'     RCVY-CI--------X'000017'
     ASSOCIATIONS
       NONVSAM--MJK.GDG1.G0001V00

NONVSAM ------- MJK.NONVSAM1
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000018'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
     VOLUMES
       VOLSER------------333001      DEVTYPE------X'30502009'     FSEQN------------------0
     ASSOCIATIONS--------(NULL)

CLUSTER ------- MJKCAT
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223
       RELEASE----------------2      EXPIRATION--------99.999
     PROTECTION-PSWD-----(NULL)      RACF----------------(No)
     ASSOCIATIONS
       DATA-----VSAM.CATALOG.BASE.DATA.RECORD
       INDEX----VSAM.CATALOG.BASE.INDEX.RECORD
```

Figure 20 (Part 3 of 5). An Example of LISTCAT ALL Output

```
DATA ------- VSAM.CATALOG.BASE.DATA.RECORD
  HISTORY
    OWNER-IDENT-------(NULL)      CREATION----------00.000
    RELEASE----------------2      EXPIRATION--------00.000
  PROTECTION-PSWD-----(NULL)      RACF----------------(No)
  ASSOCIATIONS
    CLUSTER--MJKCAT
  ATTRIBUTES
    KEYLEN-----------------44     AVGLRECL------------505     BUFSPACE------------3072     CISIZE---------------512
    RKP--------------------0      MAXLRECL------------505     EXCPEXIT----------(NULL)     CI/CA-----------------40
    SHROPTNS(3,3)    RECOVERY     SUBALLOC          NOERASE   INDEXED        NOWRITECHK    IMBED          NOREPLICAT
    UNORDERED         NOREUSE     NONSPANNED          BIND    RECVABLE
  STATISTICS
    REC-TOTAL--------------15     SPLITS-CI--------------0    EXCPS-----------------22
    REC-DELETED------------0      SPLITS-CA--------------0    EXTENTS----------------2
    REC-INSERTED-----------0      FREESPACE-%CI----------0    SYSTEM-TIMESTAMP:
    REC-UPDATED------------0      FREESPACE-%CA----------0         X'87A135BC20E95000'
    REC-RETRIEVED----------0      FREESPC-BYTES-----218112
  ALLOCATION
    SPACE-TYPE---------TRACK      HI-ALLOC-RBA------225280
    SPACE-PRI-------------33      HI-USED-RBA-------225280
    SPACE-SEC-------------18
  VOLUME
    VOLSER------------333001      PHYREC-SIZE----------512    HI-ALLOC-RBA------204800     EXTENT-NUMBER----------1
    DEVTYPE------X'30502009'      PHYRECS/TRK-----------20    HI-USED-RBA--------20480     EXTENT-TYPE--------X'00'
    VOLFLAG------------PRIME      TRACKS/CA--------------3
    LOW-KEY---------------00
    HIGH-KEY--------------3F
    HI-KEY-RBA----------6144
    EXTENTS:
    LOW-CCHH-----X'00010000'      LOW-RBA----------------0    TRACKS----------------30
    HIGH-CCHH----X'0002000A'      HIGH-RBA---------204799
  VOLUME
    VOLSER------------333001      PHYREC-SIZE----------512    HI-ALLOC-RBA------225280     EXTENT-NUMBER----------1
    DEVTYPE------X'30502009'      PHYRECS/TRK-----------20    HI-USED-RBA-------225280     EXTENT-TYPE--------X'00'
    VOLFLAG------------PRIME      TRACKS/CA--------------3
    LOW-KEY---------------40
    HIGH-KEY--------------FF
    HI-KEY-RBA--------204800
    EXTENTS:
    LOW-CCHH-----X'0002000E'      LOW-RBA----------204800     TRACKS-----------------3
    HIGH-CCHH----X'00020010'      HIGH-RBA---------225279
INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
  HISTORY
    OWNER-IDENT-------(NULL)      CREATION----------00.000
    RELEASE----------------2      EXPIRATION--------00.000
  PROTECTION-PSWD------(NULL)     RACF----------------(No)
  ASSOCIATIONS
    CLUSTER--MJKCAT
  ATTRIBUTES
    KEYLEN-----------------44     AVGLRECL--------------0     BUFSPACE--------------0      CISIZE---------------512
    RKP--------------------0      MAXLRECL------------505     EXCPEXIT----------(NULL)     CI/CA-----------------20
    SHROPTNS(3,3)    RECOVERY     SUBALLOC          NOERASE   NOWRITECHK          IMBED    NOREPLICAT      UNORDERED
    NOREUSE              BIND
  STATISTICS
    REC-TOTAL--------------3      SPLITS-CI--------------0    EXCPS-----------------13     INDEX:
    REC-DELETED------------0      SPLITS-CA--------------0    EXTENTS----------------3     LEVELS----------------2
    REC-INSERTED-----------0      FREESPACE-%CI----------0    SYSTEM-TIMESTAMP:            ENTRIES/SECT----------7
    REC-UPDATED------------0      FREESPACE-%CA----------0         X'87A135BC20E95000'     SEQ-SET-RBA--------30720
    REC-RETRIEVED----------0      FREESPC-BYTES------34816                                 HI-LEVEL-RBA----------0
  ALLOCATION
    SPACE-TYPE---------TRACK      HI-ALLOC-RBA-------36352
    SPACE-PRI--------------3      HI-USED-RBA--------36352
    SPACE-SEC--------------3
  VOLUME
    VOLSER------------333001      PHYREC-SIZE----------512    HI-ALLOC-RBA-------30720     EXTENT-NUMBER----------1
    DEVTYPE------X'30502009'      PHYRECS/TRK-----------20    HI-USED-RBA----------512     EXTENT-TYPE--------X'00'
    VOLFLAG------------PRIME      TRACKS/CA--------------1
    EXTENTS:
    LOW-CCHH-----X'0002000B'      LOW-RBA----------------0    TRACKS-----------------3
    HIGH-CCHH----X'0002000D'      HIGH-RBA----------30719
  VOLUME
    VOLSER------------333001      PHYREC-SIZE----------512    HI-ALLOC-RBA-------35840     EXTENT-NUMBER----------1
    DEVTYPE------X'30502009'      PHYRECS/TRK-----------20    HI-USED-RBA--------31232     EXTENT-TYPE--------X'80'
    VOLFLAG------------PRIME      TRACKS/CA--------------3
    LOW-KEY---------------00
    HIGH-KEY--------------3F
    EXTENTS:
    LOW-CCHH-----X'00010000'      LOW-RBA------------30270    TRACKS----------------30
    HIGH-CCHH----X'0002000A'      HIGH-RBA-----------35839
  VOLUME
    VOLSER------------333001      PHYREC-SIZE----------512    HI-ALLOC-RBA-------36352     EXTENT-NUMBER----------1
    DEVTYPE------X'30502009'      PHYRECS/TRK-----------20    HI-USED-RBA--------36352     EXTENT-TYPE--------X'80'
    VOLFLAG------------PRIME      TRACKS/CA--------------3
    LOW-KEY---------------40
    HIGH-KEY--------------FF
    EXTENTS:
    LOW-CCHH-----X'0002000E'      LOW-RBA------------35840    TRACKS-----------------3
    HIGH-CCHH----X'00020010'      HIGH-RBA-----------36351
```

Figure 20 (Part 4 of 5). An Example of LISTCAT ALL Output

```
VOLUME -------- 333001
     HISTORY
       RELEASE----------------2     RCVY-VOL----------333001     RCVY-DEVT----X'30502009'     RCVY-CI--------X'000009'
     CHARACTERISTICS
       BYTES/TRK----------13165     DEVTYPE------X'30502009'     MAX-PHYREC-SZ------13030     DATASETS-ON-VOL--------5
       TRKS/CYL-----------19        VOLUME-TIMESTAMP:            MAX-EXT/ALLOC----------5     DATASPCS-ON-VOL--------1
       CYLS/VOL-----------411           X'87A1390412233000'
     DATASPACE
       DATASETS---------------5     FORMAT-1-DSCB:               ATTRIBUTES:
       EXTENTS----------------1     CCHHR------X'0000000103'     SUBALLOC
       SEC-ALLOC--------------2     TIMESTAMP                    EXPLICIT
       TYPE------------CYLINDER          X'87A1355C90BB2000'     USERCAT
       EXTENT-DESCRIPTOR:
       TRACKS-TOTAL---------152     BEG-CCHH-----X'00010000'     SPACE-MAP-----------4157
       TRACKS-USED----------65
       DATASET-DIRECTORY:
         DSN----MJKCAT                                           ATTRIBUTES--------(NULL)     EXTENTS----------------3
         DSN----T7D503F0.VSAMDSET.DFD75223.T87A1369.T7D503F0     ATTRIBUTES--------(NULL)     EXTENTS----------------1
         DSN----T7D550E0.VSAMDSET.DFD75223.T87A1369.T7D550E0     ATTRIBUTES--------(NULL)     EXTENTS----------------1
         DSN----T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320     ATTRIBUTES--------(NULL)     EXTENTS----------------1
         DSN----T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380     ATTRIBUTES--------(NULL)     EXTENTS----------------1
                              LISTING FROM CATALOG -- MJKCAT
         THE NUMBER OF ENTRIES PROCESSED WAS:
                   AIX -------------------1
                   ALIAS -----------------1
                   CLUSTER ---------------2
                   DATA ------------------3
                   GDG -------------------1
                   INDEX -----------------3
                   NONVSAM ---------------2
                   PAGESPACE -------------0
                   PATH ------------------1
                   SPACE -----------------1
                   USERCATALOG -----------0
                   TOTAL -----------------15
         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 5 of 5). An Example of LISTCAT ALL 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 21). 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 DATA and INDEX.

```
  /* G:   LIST SPACE ALLOCATION INFORMATION  */
       LISTCAT -
           ALLOCATION -
           CATALOG(MJKCAT)

                              LISTING FROM CATALOG -- MJKCAT
AIX ----------- MJK.ALT.INDEX1
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223      RCVY-VOL----------333001      RCVY-CI--------X'000010'
       RELEASE----------------2      EXPIRATION--------00.000      RCVY-DEVT----X'30502009'

   DATA ------- T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223      RCVY-VOL----------333001      RCVY-CI--------X'000011'
       RELEASE----------------2      EXPIRATION--------00.000      RCVY-DEVT----X'30502009'
     ALLOCATION
       SPACE-TYPE---------TRACK      HI-ALLOC-RBA-------73728
       SPACE-PRI----------------6    HI-USED-RBA------------0
       SPACE-SEC----------------6
     VOLUME
       VOLSER------------333001      PHYREC-SIZE---------4096      HI-ALLOC-RBA-------73728      EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK------------3      HI-USED-RBA------------0      EXTENT-TYPE--------X'40'
       VOLFLAG------------PRIME      TRACKS/CA----------------6
       EXTENTS:
       LOW-CCHH-----X'00040001'      LOW-RBA----------------0      TRACKS-----------------6
       HIGH-CCHH----X'00040006'      HIGH-RBA-----------73727

   INDEX ------ T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223      RCVY-VOL----------333001      RCVY-CI--------X'000012'
       RELEASE----------------2      EXPIRATION--------00.000      RCVY-DEVT----X'30502009'
     ALLOCATION
       SPACE-TYPE---------TRACK      HI-ALLOC-RBA-------10240
       SPACE-PRI----------------1    HI-USED-RBA------------0
       SPACE-SEC----------------1
     VOLUME
       VOLSER------------333001      PHYREC-SIZE----------512      HI-ALLOC-RBA-------10240      EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK-----------20      HI-USED-RBA------------0      EXTENT-TYPE--------X'40'
       VOLFLAG------------PRIME      TRACKS/CA--------------1
       EXTENTS:
       LOW-CCHH-----X'00040007'      LOW-RBA----------------0      TRACKS-----------------1
       HIGH-CCHH----X'00040007'      HIGH-RBA-----------10239

   PATH ------- MJK.AIX1.PATH
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223      RCVY-VOL----------333001      RCVY-CI--------X'000014'
       RELEASE----------------2      EXPIRATION--------00.000      RCVY-DEVT----X'30502009'

CLUSTER ------- MJK.CLUSTER1
     HISTORY
       OWNER-IDENT-----OWNCLUST      CREATION----------75.223      RCVY-VOL----------333001      RCVY-CI--------X'00000E'
       RELEASE----------------2      EXPIRATION--------75.323      RCVY-DEVT----X'30502009'

   DATA ------- T7D503F0.VSAMDSET.DFD75223.T87A1369.T7D503F0
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223      RCVY-VOL----------333001      RCVY-CI--------X'00000D'
       RELEASE----------------2      EXPIRATION--------00.000      RCVY-DEVT----X'30502009'
     ALLOCATION
       SPACE-TYPE---------TRACK      HI-ALLOC-RBA-------20480
       SPACE-PRI----------------2    HI-USED-RBA------------0
       SPACE-SEC----------------2
     VOLUME
       VOLSER------------333001      PHYREC-SIZE---------2048      HI-ALLOC-RBA-------20480      EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK------------6      HI-USED-RBA------------0      EXTENT-TYPE--------X'40'
       VOLFLAG------------PRIME      TRACKS/CA----------------2
       EXTENTS:
       LOW-CCHH-----X'00030011'      LOW-RBA----------------0      TRACKS-----------------2
       HIGH-CCHH----X'00030012'      HIGH-RBA-----------20479
```

Figure 21 (Part 1 of 3). An Example of LISTCAT ALLOCATION Output

```
   INDEX ------ T7D550E0.VSAMDSET.DFD75223.T87A1369.T7D550E0
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000F'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
     ALLOCATION
       SPACE-TYPE---------TRACK      HI-ALLOC-RBA-------10240
       SPACE-PRI--------------1      HI-USED-RBA------------0
       SPACE-SEC--------------1
     VOLUME
       VOLSER------------333001      PHYREC-SIZE----------512     HI-ALLOC-RBA-------10240     EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK----------20      HI-USED-RBA------------0     EXTENT-TYPE--------X'40'
       VOLFLAG------------PRIME      TRACKS/CA--------------1
       EXTENTS:
       LOW-CCHH-----X'00040000'      LOW-RBA----------------0     TRACKS-----------------1
       HIGH-CCHH----X'00040000'      HIGH-RBA----------10239
CLUSTER ------- MJKCAT
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223
       RELEASE----------------2      EXPIRATION--------99.999
   DATA ------- VSAM.CATALOG.BASE.DATA.RECORD
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------00.000
       RELEASE----------------2      EXPIRATION--------00.000
     ALLOCATION
       SPACE-TYPE---------TRACK      HI-ALLOC-RBA------225280
       SPACE-PRI-------------33      HI-USED-RBA-------225280
       SPACE-SEC-------------18
     VOLUME
       VOLSER------------333001      PHYREC-SIZE----------512     HI-ALLOC-RBA------204800     EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK----------20      HI-USED-RBA--------20480     EXTENT-TYPE--------X'00'
       VOLFLAG------------PRIME      TRACKS/CA--------------3
       LOW-KEY----------------00
       HIGH-KEY---------------3F
       HI-KEY-RBA----------6144
       EXTENTS:
       LOW-CCHH-----X'00010000'      LOW-RBA----------------0     TRACKS----------------30
       HIGH-CCHH----X'0002000A'      HIGH-RBA----------204799
     VOLUME
       VOLSER------------333001      PHYREC-SIZE----------512     HI-ALLOC-RBA------225280     EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK----------20      HI-USED-RBA-------225280     EXTENT-TYPE--------X'00'
       VOLFLAG------------PRIME      TRACKS/CA--------------3
       LOW-KEY----------------40
       HIGH-KEY---------------FF
       HI-KEY-RBA--------204800
       EXTENTS:
       LOW-CCHH-----X'0002000E'      LOW-RBA----------204800      TRACKS-----------------3
       HIGH-CCHH----X'00020010'      HIGH-RBA----------225279
   INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------00.000
       RELEASE----------------2      EXPIRATION--------00.000
     ALLOCATION
       SPACE-TYPE---------TRACK      HI-ALLOC-RBA-------36352
       SPACE-PRI--------------3      HI-USED-RBA--------36352
       SPACE-SEC--------------3
     VOLUME
       VOLSEP------------333001      PHYREC-SIZE----------512     HI-ALLOC-RBA-------30720     EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK----------20      HI-USED-RBA----------512     EXTENT-TYPE--------X'00'
       VOLFLAG------------PRIME      TRACKS/CA--------------1
       EXTENTS:
       LOW-CCHH-----X'0002000B'      LOW-RBA----------------0     TRACKS-----------------3
       HIGH-CCHH----X'0002000D'      HIGH-RBA----------30719
     VOLUME
       VOLSER------------333001      PHYREC-SIZE----------512     HI-ALLOC-RBA-------35840     EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK----------20      HI-USED-RBA--------31232     EXTENT-TYPE--------X'80'
       VOLFLAG------------PRIME      TRACKS/CA--------------3
       LOW-KEY----------------00
       HIGH-KEY---------------3F
       EXTENTS:
       LOW-CCHH-----X'00010000'      LOW-RBA-----------30720      TRACKS----------------30
       HIGH-CCHH----X'0002000A'      HIGH-RBA----------35839
     VOLUME
       VOLSER------------333001      PHYREC-SIZE----------512     HI-ALLOC-RBA-------36352     EXTENT-NUMBER----------1
       DEVTYPE------X'30502009'      PHYRECS/TRK----------20      HI-USED-RBA--------36352     EXTENT-TYPE--------X'80'
       VOLFLAG------------PRIME      TRACKS/CA--------------3
       LOW-KEY----------------40
       HIGH-KEY---------------FF
       EXTENTS:
       LOW-CCHH-----X'0002000E'      LOW-RBA-----------35840      TRACKS-----------------3
       HIGH-CCHH----X'00020010'      HIGH-RBA----------36351
```

Figure 21 (Part 2 of 3). An Example of LISTCAT ALLOCATION Output

```
                        LISTING FROM CATALOG -- MJKCAT

        THE NUMBER OF ENTRIES PROCESSED WAS:
                  AIX ------------------1
                  ALIAS ----------------0
                  CLUSTER --------------2
                  DATA -----------------3
                  GDG ------------------0
                  INDEX ----------------3
                  NONVSAM --------------0
                  PAGESPACE ------------0
                  PATH -----------------1
                  SPACE ----------------0
                  USERCATALOG ----------0
                  TOTAL ----------------10

        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 3 of 3). 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 22). Only these types of entries have HISTORY information: ALTERNATEINDEX, CLUSTER, DATA, INDEX, NONVSAM, and PATH.

```
/* H: LIST HISTORY INFORMATION FOR SELECTED ENTRIES */
LISTCAT -
  HISTORY -
  CATALOG(MJKCAT)
                          LISTING FROM CATALOG -- MJKCAT
AIX ---------- MJK.ALT.INDEX1
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000010'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
   DATA ------- T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000011'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
   INDEX ------ T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000012'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
   PATH ------- MJK.AIX1.PATH
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000014'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
CLUSTER ------- MJK.CLUSTER1
     HISTORY
       OWNER-IDENT-----OWNCLUST      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000E'
       RELEASE----------------2      EXPIRATION--------75.323     RCVY-DEVT----X'30502009'
   DATA ------- T7D503F0.VSAMDSET.DFD75223.T87A1369.T7D503F0
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000D'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
   INDEX ------ T7D550E0.VSAMDSET.DFD75223.T87A1369.T7D550E0
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'00000F'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
GDG BASE ------ MJK.GDG1
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000015'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
   NONVSAM ---- MJK.GDG1.G0001V00
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000016'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
ALIAS --------- MJK.GDG1.ALIAS
     HISTORY
       RELEASE----------------2      RCVY-VOL----------333001     RCVY-DEVT----X'30502009'     RCVY-CI--------X'000017'
NONVSAM ------- MJK.NONVSAM1
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223     RCVY-VOL----------333001     RCVY-CI--------X'000018'
       RELEASE----------------2      EXPIRATION--------00.000     RCVY-DEVT----X'30502009'
CLUSTER ------- MJKCAT
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------75.223
       RELEASE----------------2      EXPIRATION--------99.999
   DATA ------- VSAM.CATALOG.BASE.DATA.RECORD
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------00.000
       RELEASE----------------2      EXPIRATION--------00.000
   INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
     HISTORY
       OWNER-IDENT-------(NULL)      CREATION----------00.000
       RELEASE----------------2      EXPIRATION--------00.000
VOLUME -------- 333001
     HISTORY
       RELEASE----------------2      RCVY-VOL----------333001     RCVY-DEVT----X'30502009'     RCVY-CI--------X'000009'
```

Figure 22 (Part 1 of 2). An Example of LISTCAT HISTORY Output

```
                         LISTING FROM CATALOG -- MJKCAT

        THE NUMBER OF ENTRIES PROCESSED WAS:
                  AIX ------------------1
                  ALIAS ----------------1
                  CLUSTER --------------2
                  DATA -----------------3
                  GDG ------------------1
                  INDEX ----------------3
                  NONVSAM --------------2
                  PAGESPACE ------------0
                  PATH -----------------1
                  SPACE ----------------1
                  USERCATALOG ----------0
                  TOTAL ----------------15

        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 (Part 2 of 2). 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 23, for example, LISTCAT was run August 11, 1975. Because all entries in the listed catalog, MJKCAT, were created on the same day, no entries are listed as a result of the LISTCAT CREATION(5) job. When that job is run on an older catalog, each entry whose creation date is previous to the number of days specified with the CREATION parameter is listed (that is, the CREATION number of days specifies that all objects in the catalog at least 5 days old are to be listed).

When you list all entries of a catalog, and you specify the CREATION parameter, each user catalog connector entry and each alias entry are also listed regardless of their creation date.

When the LISTCAT EXPIRATION(20) job is run, each entry whose expiration date occurs within 20 days of today's date is listed.

These types of entries can have a creation or expiration date: ALTERNATEINDEX, PATH, CLUSTER, DATA, INDEX, NONVSAM, GDG, and PAGESPACE.

```
/*  I: LIST EACH CATALOG ENTRY WHOSE CREATION DATE   */
/*     IS 5 DAYS AGO OR GREATER (THAT IS, THE OBJECT */
/*     IS AT LEAST 5 DAYS OLD).                      */
LISTCAT -
   CREATION(5) -
   CATALOG(MJKCAT)

                            LISTING FROM CATALOG -- MJKCAT
ALIAS --------- MJK.GDG1.ALIAS
                            LISTING FROM CATALOG -- MJKCAT
         THE NUMBER OF ENTRIES PROCESSED WAS:
                   AIX ----------0
                   ALIAS --------1
                   CLUSTER ------0
                   DATA ---------0
                   GDG ----------0
                   INDEX --------0
                   NONVSAM ------0
                   PAGESPACE ----0
                   PATH ---------0
                   SPACE --------0
                   USERCATALOG --0
                   TOTAL --------1
         THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
/*  J: LIST EACH CATALOG ENTRY WHOSE EXPIRATION DATE */
/*     IS WITHIN 20 DAYS
LISTCAT -
   EXPIRATION(20) -
   CATALOG(MJKCAT)
                            LISTING FROM CATALOG -- MJKCAT
AIX ----------- MJK.ALT.INDEX1
   DATA ------- T688F320.VSAMDSET.DFD75223.T87A13A7.T688F320
   INDEX ------ T6893380.VSAMDSET.DFD75223.T87A13A7.T6893380
   PATH ------- MJK.AIX1.PATH
GDG BASE ------ MJK.GDG1
   NONVSAM ---- MJK.GDG1.G0001V00
ALIAS --------- MJK.GDG1.ALIAS
NONVSAM ------- MJK.NONVSAM1
VOLUME -------- 333001
                            LISTING FROM CATALOG -- MJKCAT
         THE NUMBER OF ENTRIES PROCESSED WAS:
                   AIX ----------1
                   ALIAS --------1
                   CLUSTER ------0
                   DATA ---------1
                   GDG ----------1
                   INDEX --------1
                   NONVSAM ------2
                   PAGESPACE ----0
                   PATH ---------1
                   SPACE --------1
                   USERCATALOG --0
                   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 23. An Example of LISTCAT CREATION/EXPIRATION Output

Examples of LISTCAT in a TSO Environment

The following examples illustrate the output produced at a TSO terminal for a LISTCAT NAMES (default) and LISTCAT VOLUME. A TSO logon ID of IBMUSER is assumed.

For LISTCAT NAMES, the catalog name is printed followed by the names of all entries that have a high-level qualifier equal to the USER logon ID.

For LISTCAT VOLUME, all entrynames that have a high-level qualifier equal to the USER logon ID are printed followed by the volume serial numbers for those entries that contain volume information.

Note: Because volume serial numbers for a cluster or an alternate index are contained in the data and index components, the data and index must have been named on the initial DEFINE in order to list the volume serial numbers.

```
LOGON IBMUSER

   .
   .
   .
READY
LISTCAT

IN CATALOG: AMASTCAT
IBMUSER.AIX
IBMUSER.AIXDATA
IBMUSER.AIXIDX
IBMUSER.GDG
IBMUSER.GDG.G0001V00
IBMUSER.GDG.G0002V00
IBMUSER.GDG.G0003V00
IBMUSER.KSDS
IBMUSER.KSDSDATA
IBMUSER.KSDSIDX
IBMUSER.NVSAM1
IBMUSER.NVSAM2
IBMUSER.NVSAM3
IBMUSER.NVSAM4
IBMUSER.NVSAM5
READY

LISTCAT VOLUME

IBMUSER.AIX
IBMUSER.AIXDATA
  --VOLUMES--
    333001
IBMUSER.AIXIDX
  --VOLUMES--
    333001
IBMUSER.GDG
IBMUSER.GDG.G0001V00
  --VOLUMES--
    333001
    333002
    333003
IBMUSER.GDG.G0002V00
  --VOLUMES--
    333004
    333005
    333006
    333007
    333008
```

```
IBMUSER.GDG.G0003V00
  --VOLUMES--
    333009
    333010
IBMUSER.KSDS
IBMUSER.KSDSDATA
  --VOLUMES--
    333001
IBMUSER.KSDSIDX
  --VOLUMES--
    333001
IBMUSER.NVSAM1
  --VOLUMES--
    231401
    231402
IBMUSER.NVSAM2
  --VOLUMES--
    231403
    231404
    231405
IBMUSER.NVSAM3
  --VOLUMES--
    231406
IBMUSER.NVSAM4
  --VOLUMES--
    231407
IBMUSER.NVSAM5
  --VOLUMES--
    231408
    231409
    231410
    231411
    231412
READY
```

APPENDIX C: JCL DD PARAMETERS TO TAKE CARE WITH

Because OS/VS 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 24 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, OS/VS 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.
DSNAME	*dsname(areaname)*	The name is used; areaname is ignored.
	dsname(generation)	A generation data group entry must exist in the VSAM catalog for this to apply.
	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.

Figure 24 (Part 1 of 2). JCL DD Parameters

Parameter	Subparameter	Comment
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.
LABEL	**NOPWREAD**	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD specification in the LABEL parameter.
	PASSWORD	The password-protection bit is set for all VSAM data sets, regardless of the PASSWORD/NOPWREAD 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, OS/VS 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.
SUBALLOC		VSAM data spaces are initially allocated as a result of the Access Method Services DEFINE command. If SUBALLOC is specified, OS/VS 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 24 (Part 2 of 2). JCL DD Parameters

APPENDIX D: INTERPRETING LISTCRA OUTPUT LISTINGS

When you code the LISTCRA command, you can specify options that allow you to tailor the contents of the LISTCRA output. This appendix illustrates the various types of the LISTCRA output, the order in which entries are listed, and the meanings of the listed fields.

There are five types of LISTCRA listings. Four of the types are illustrated in this appendix. The fifth type, SEQUENTIALDUMP, is exactly the same as the DUMP NOCOMPARE type except that the entries aren't sorted into groups.

Each listed entry is identified by *type* (that is, cluster, nonVSAM, etc.) and by *name*. Entries are usually listed in alphabetic order within groups of entries, according to entryname (except for SEQUENTIALDUMP). However, if insufficient virtual storage is available for the sorting operation, the records are listed as they appear in the catalog recovery area.

On the listing, entries are sorted into three groups (except for a SEQUENTIALDUMP listing):

- VSAM entries (which are also sorted according to entryname). Cluster entries, alternate index entries, and their associated data, index, and path entries are within this group.
- Other entries that are sorted according to entryname. User catalog connector entries and nonVSAM entries are in this group.
- Unsorted entries: other entries that are not sorted according to entryname. Entries that are listed within this group depend on the type of LISTCRA listing requested.

The following list contains the abbreviation, type, and description for each kind of entry that can be listed as a result of the LISTCRA command:

Abbreviation	Type	Description
AIX	G	Alternate index entry
ALIA	X	Alias entry
CLUS	C	Cluster entry
DATA	D	Data component entry
FRSP	F	Freespace entry
GDGB	B	Generation data group entry
INDX	I	Index component entry
NONV	A	NonVSAM data set entry
OEXT	E	Extension record for an entry other than a volume entry
PATH	P	Path entry
UCAT	U	User catalog connector entry (in the master catalog only)
UPGD	Y	Upgrade set entry
VEXT	W	Extension record for a volume entry
VOL	V	Volume entry

Examples of LISTCRA Listings

The five types of LISTCRA listings are:

- NAME NOCOMPARE—This is the default option. Each entry name, its volumes, and the name and volumes of each related entry is listed. Within the "unsorted entries" group, all catalog records not yet printed are printed. See Figure 25.
- DUMP NOCOMPARE—When DUMP is specified (NOCOMPARE is a default), each record is listed in the dump format. In addition, the name and volumes of each indirectly related entry are also listed. Within the "unsorted entries" group, the CRA's self-describing records (control intervals 0 through 8) are printed, and any freespace records and records not yet printed are printed. See Figure 26.
- NAME COMPARE—When COMPARE is specified (NAME is a default), only the name of each miscompared catalog entry is listed. A miscompared catalog entry is one whose information is not identical to the information contained in the entry's copy in the catalog recovery area. A MISCOMPARE message is printed that identifies the most serious level of miscomparison. Within the "unsorted entries" group, miscompared records not yet printed out are printed. See Figure 27.
- DUMP COMPARE—When DUMP and COMPARE are specified, only the records (in dump format) of each miscompared catalog entry is listed. A miscompared catalog entry is one whose information is not identical to the information contained in the entry's copy in the catalog area. For each entry, the catalog recovery area copy is listed first, followed by the catalog entry, followed by a line that contains asterisks to identify each miscompared byte. Following each entry, a MISCOMPARE message is printed that identifies the most serious level of miscomparison. Within the "unsorted entries" group, miscompared records not yet printed out are printed in the dump format. See Figure 28.

  Note: As explained above, all MISCOMPARE messages result from a comparison between the catalog and the CRA if the comparison shows that the catalog and CRA records are not identical. No inference is intended as to which is correct. You must make this determination yourself by looking at other mismatches in the same listing and by examining related records in the catalog or CRA.

- SEQUENTIALDUMP—When SEQUENTIALDUMP is specified, each record in the catalog recovery area is printed. The format of the output is the same as the DUMP NOCOMPARE format, except that the entries aren't sorted into groups or alphabetic sequence.

  See the *OS/VS2 Independent Component: Virtual Storage Access Method (VSAM) Logic* for a complete description of catalog recovery area record formats, catalog record formats, and relationships between catalog records.

Job Control Language (JCL) for LISTCRA Jobs

The job control language (JCL) statements that can be used to list catalog recovery areas are:

```
//LISTCAT   JOB     ...
//STEP1     EXEC    PGM=IDCAMS
//STEPCAT   DD      DSN=YOURCAT,DISP=SHR
//CATDD     DD      DSN=YOURCAT,DISP=OLD
//CRADD1    DD      VOL=SER=333001,UNIT=3330,DISP=OLD
//CRADD2    DD      VOL=SER=333002,UNIT=3330,DISP=OLD
//OUTDD     DD      DSN=LISTCRA.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      *
     LISTCRA -
        COMPARE -
        INFILE(CRADD1,CRADD2) -
        CATALOG(YOURCAT/PASSWORDÞCATDD) -
        MASTERPW(SECRET) -
        OUTFILE(OUTDD) -
        ...
/*
```

The JOB statement contains user and accounting information required for your installation.

The EXEC statement identifies the program to be executed, IDCAMS (that is, the Access Method Services program).

- STEPCAT DD, which allocates your catalog. This is only required if the COMPARE option is specified. The catalog must be open as a catalog before it can be opened as a data set through the DD card that so identifies it for the comparisons.
- CATDD, which specifies the catalog to be opened as a data set and compared to the catalog recovery areas. This is only required if the COMPARE option is specified.
- CRADD1, which specifies a volume whose catalog recovery area (CRA) is to be listed.
- CRADD2, which specifies another volume whose CRA is to be listed.
- OUTDD, which specifies an alternate output file, so that the LISTCRA output can be written onto an auxiliary storage device. The LISTCRA command's OUTFILE parameter points to the OUTDD DD statement. Only the LISTCRA output is written to the alternate output device. JCL statements, system messages, and job statistics are written to the SYSPRINT output device.
  - DSN=LISTCRA.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 LISTCRA 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 LISTCRA command, DISP=(MOD,KEEP) or DISP=(MOD,PASS) can be used to concatenate all of the LISTCRA output in one sequential file.
- DCB=(RECFM=VBA,LRECL=125,BLKSIZE=629) specifies that the LISTCRA 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 LISTCRA 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 LISTCRA command parameters shown above are common to the LISTCRA NAME COMPARE example that follows. Other LISTCRA parameters may be coded and the output that results is illustrated.

COMPARE, which specifies that the CRA entries are to be compared with the catalog entries identified by the CATALOG parameter. Only those that miscompare will be listed.

INFILE, which specifies the two DD statements, CRADD1 and CRADD2, which identify the two volumes whose CRA's are to be compared with the catalog entries and the miscomparing entries listed.

CATALOG, which identifies the DD statement, CATDD, which identifies the catalog, YOURCAT, whose entries are to be compared with those in the CRAs. If the catalog is password protected, its master password, PASSWORD, is required.

MASTERPW, which specifies the master password of the master catalog, SECRET, to enable the CRAs to be OPENed.

OUTFILE, which points to the OUTDD DD statement. The OUTDD DD statement allocates an alternate output file for the LISTCRA output.

If you want to print the LISTCRA 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, LISTCRA.OUTPUT, that was allocated previously:

```
//PRINTOUT   JOB     ...
//STEP1      EXEC    PGM=IEBGENER
//SYSUT1     DD      DSN=LISTCRA.OUTPUT,UNIT=2400,
               VOL=SER=TAPE1,LABEL=(1,NL),DISP=(OLD,KEEP),
               DCB=RECFM=VBA,LRECL=125,BLKSIZE=629)
//SYSUT2     DD      SYSOUT=A
//SYSPRINT   DD      SYSOUT=A
//SYSIN      DD      DUMMY
/*
//
```

LISTCRA Output Listing

When you specify LISTCRA without any parameters, each entry name, its volume(s), and the name and volume(s) of each related entry is listed. The same listing would result if the NAME and NOCOMPARE parameters were specified.

You can use this type of listing to list the name of each catalog entry whose copy is in the catalog recovery area, and to determine the number of entries in each catalog recovery area. The total number of entries is an approximate size, in records, of the catalog recovery area.

LISTCRA DUMP Output Listing

When you specify LISTCRA with the DUMP parameter, each record in the catalog recovery area is listed in dump format. In addition, the name and volumes of each indirectly related entry is also listed. The same listing would result if the DUMP and NOCOMPARE parameters were specified.

When you specify LISTCRA SEQUENTIALDUMP, the same listing results—except that the records are listed in their entry-sequence within the catalog recovery area.

You can use this type of listing to list the complete contents of each catalog entry whose copy is in the catalog recovery area, and to determine the exact number of entries and records in each catalog recovery area.

LISTCRA COMPARE Output Listing

When you specify LISTCRA with the COMPARE parameter, each record in the catalog recovery area is compared to its original in the catalog. The name of each catalog entry that miscompares is listed. A miscompared catalog entry is one whose information is not identical to the information contained in the entry's copy in the catalog recovery area. The same listing would result if the NAME and COMPARE parameters were specified.

You can use this type of listing to determine the number of damaged entries in your catalog, and you can get an idea of the type of damage that occurred to each entry. You can issue the EXPORTRA and IMPORTRA commands to recover the catalog and make its damaged entries usable.

```
                    LISTING OF CATALOG RECOVERY AREA FOR VOLUME -- 333301 -- VSAM ENTRIES
VOL  - 333301
     CRAVOLRCD  -  10/07/74   23:35:00
     F4DSCBVSAM -  10/07/74   23:35:00
     F4DSCBDUMP -  10/07/74   23:35:00
CLUS - AA.LISTCRA.ESDS
  DATA - TF41C9A0.VSAMDSET.DFD74280.T861DAD4.TF41C9A0
    DATA VOL -
      333301
  AIX  - AA2.LR.ESDS
    DATA VOL -
      333301
    INDX VOL -
      333301
  AIX  - AA1.LR.ESDS
    DATA VOL -
      333301
    INDX VOL -
      333301
  PATH - AAU.LR.ESDS
  UPGD -
CLUS - AA.LISTCRA.KSDS
  DATA - T5C55DD0.VSAMDSET.DFD74280.T861DB0A.T5C55DD0
    DATA VOL -
      333301
  INDX - T5C56E70.VSAMDSET.DFD74280.T861DB0A.T5C56E70
    INDX VOL -
      333301
AIX  - AA1.LR.ESDS
  DATA - TCD41020.VSAMDSET.DFD74280.T861DB23.TCD41020
    DATA VOL -
      333301
  INDX - TCD41F00.VSAMDSET.DFD74280.T861DB23.TCD41F00
    INDX VOL -
      333301
  CLUS - AA.LISTCRA.ESDS
  PATH - AA1U.LR.ESDS
  PATH - AA1N.LR.ESDS
CLUS - LR.MCKEYRNG.KSDS
  DATA - TF081480.VSAMDSET.DFD74280.T861DB0E.TF081480
    DATA VOL - HIGH KEY
      333301 - C1C1C1C1C1C1C1C1C5F9
      333301 - C1C1C1C1C1C1C1C1E9E9
  INDX - TF082370.VSAMDSET.DFD74280.T861DB0E.TF082370
    INDX VOL -
      333301
      333301
                    LISTING OF CATALOG RECOVERY AREA FOR VOLUME -- 333301 -- OTHER ENTRIES
UCAT - USERCAT2
  DATA VOL -
    333304
NUMBER OF ENTRIES PROCESSED
 CLUS -  3
 DATA -  4
 AIX  -  1
 INDX -  2
 PATH -  3
 VOL  -  1
 UPGD -  1
SUM  - 15
```

Figure 25 (Part 1 of 2). An Example of LISTCRA NAME NOCOMPARE Output

LISTCRA NAME NOCOMPARE Output Notes:

1. Volume serial number
2. Identifies the group that the following entries are within (in this case, VSAM Entries)
3. The time stamps:

   CRAVOLRCD is the timestamp of the CRA volume record. It is updated only when the first catalog record and first CRA volume record are updated, after the CRA is opened when VSAM is running under OS/VS. (Note: If the volume is moved to a DOS/VS system and used there, the CRAVOLRCD timestamp is updated each time the catalog and CRA volume records are updated for a space allocation change.)

   F4DSCBVSAM is a timestamp in the format-4 DSCB, and is updated in the same manner as the CRAVOLRCD timestamp.

   F4DSCBDUMP is a timestamp in the format-4 DSCB, and is updated in the same manner as the CRAVOLRCD timestamp. It is also updated whenever the OS/VS utility program IEHDASDR dumps the volume.
4. VSAM entries are listed along with the entryname and volume(s) of each related entry.
5. Paths are shown as related only to the entry that the path provides access to. A path that serves as an alias for a VSAM data set is listed with the data set's entry. A path that shows the relationship of an alternate index to a base cluster is listed with the alternate index's entry only.
6. UPGD indicates that there are alternate indexes in the base cluster's upgrade set. If you want to identify the indexes in the upgrade set, use the DUMP NOCOMPARE option.
7. The high key value of each key range is shown.
8. SUM is the total number of catalog recovery area entries printed.

Figure 25 (Part 2 of 2). An Example of LISTCRA NAME NOCOMPARE Output

LISTCRA DUMP COMPARE Output Listing

When you specify LISTCRA with the DUMP and COMPARE parameters, each record in the catalog recovery area is compared to its original in the catalog. Each catalog entry's copy in the catalog recovery area that miscompares is listed, followed by its original catalog entry (which is damaged), followed by asterisks to indicate each miscompared byte.

You can use this type of listing to determine the exact damage that occurred to each entry. You might be able to correct some or all of the damage by using the ALTER command. You can issue the EXPORTRA and IMPORTRA commands to recover the catalog and make its damaged entries usable.

```
                                LISTING OF CATALOG RECOVERY AREA FOR VOLUME -- 333301 -- VSAM ENTRIES
VOL   - 333301

000009 00000000 01F3F3F3 F3F0F100 00093050  2009861D 8A4E0000 00000000 00000000  .....333301....&.....+..........
  0020 00000000 00000000 00000000 E501BD00  7FF3F3F3 F3F0F100 00000000 00000000  ..........V.... 333301.........
  0040 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00861DEE  ................................
  0060 1152A820 00305020 09000032 E6019B00  13336DBF BF380002 00000500 00000000  ......&.....W.....  ............
  0080 00000000 2600003E 00000000 000D8500  0100000E 85000200 000F8500 03000037  ................................
  00A0 86000100 00378800 01000037 88000200  00378800 03000037 88000400 00378800  ................................
  00C0 05000037 88000600 00480800 07000037  88000800 00378800 09000037 88000A00  ................................
  00E0 00378800 0B000037 88000C00 00378800  0D000037 88000E00 00378800 0F000037  ................................
  0100 88001000 00378800 11000037 88001200  00378800 1300003E 88001400 003E8800  ................................
  0120 1500003E 88001600 003E8800 1700003E  88001800 003E8800 1900003E 88001A00  ................................
  0140 003E8800 1B000000 08001C00 000C0800  1D000018 08001E00 00240800 1F000030  ................................
  0160 08002000 003C0800 21000C00 0051861D  B1D60000 42000C00 0052861D B1D60000  ................O...........O..
  0180 46000C00 0054861D B2050000 47000C00  0055861D B2050000 48000C00 0057861D  ................................
  01A0 B23F0000 4C000C00 0058861D B23F0000  50000C00 0060861D B79C0000 2C1DB173  ....)...........&....-..........


CLUS - KSDS01

00000E 00000017 01F3F3F3 F3F0F100 000E3050  2009861D 8A4E0000 00000000 00000000  .....333301....&.....+..........
  0020 00000000 00000000 00000000 C300A800  6CD2E2C4 E2F0F140 40404040 40404040  ............C...(KSDS01
  0040 40404040 40404040 40404040 40404040  40404040 40404040 40404040 40FFFFFF                              ...
  0060 FFFFFFFF FF74280F 00000F00 00000000  00070000 00C00000 00000201 00006024  .....(.........................
  0080 0200000C 02030000 12020400 00004401  0006C400 00180006 C9000019 0006D900  ..................D.....I.....R.
  00A0 001A0006 C700001B 00000000 00000000  00000000 00000000 00000000 00000000  ....G...........................

DATA - T188D0C0.VSAMDSET.DFD74280.T861D99E.T188D0C0

00000D 00000018 01F3F3F3 F3F0F100 000D3050  2009861D 8A4E861D 99E20000 00000000  .....333301....&.....+...S......
  0020 00000000 00000000 00000000 C4016200  8FE3F1F8 F8C4F0C3 F04BE5E2 C1D4C4E2  ............D....T188D0C0.VSAMDS
  0040 C5E34BC4 C6C4F7F4 F2F8F04B E3F8F6F1  C4F9F9C5 4BE3F1F8 F8C4F0C3 F0FFFFFF  ET.DFD74280.T861D99E.T188D0C0...
  0060 FFFFFFFF FF74280F 00000000 20000000  22000000 01000001 80000000 00000030  .....(.........................
  0080 0000000F F90000FF FFFFFFFF FFFFFF00  00000000 06000000 C0000000 00010100  ....9...........................
  00A0 00620201 0000A902 02000068 03010000  00440100 62608000 60000000 03001400  .....-..-.......-..-............
  00C0 00000300 00000000 00000010 0000000F  F9000000 00000000 00000000 00000000  ................9...............
  00E0 00000000 00000000 00000000 00000000  01000000 00000000 00000000 00000000  ................................
  0100 00000000 00000030 00000000 00000000  00000000 000006C3 00001703 27305020  .....(.........C.......&.
  0120 09F3F3F3 F3F0F100 00800100 00000000  00000000 00300000 00100000 03000140  .333301.........................
  0140 00010000 00000014 00010002 00000002  00000001 00000000 00002FFF 0006E800  ..............................Y.
  0160 00150000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  ................................

INDX - T188E190.VSAMDSET.DFD74280.T861D99E.T188E190

00000F 00000019 01F3F3F3 F3F0F100 000F3050  2009861D 8A4E861D 99E20000 00000000  .....333301....&.....+...S......
  0020 00000000 00000000 00000000 C9015700  8FE3F1F8 F8C5F1F9 F04BE5E2 C1D4C4E2  ............I....T188E190.VSAMDS
  0040 C5E34BC4 C6C4F7F4 F2F8F04B E3F8F6F1  C4F9F9C5 4BE3F1F8 F8C5F1F9 F0FFFFFF  ET.DFD74280.T861D99E.T188E190...
  0060 FFFFFFFF FF74280F 00000000 2000FFFF  FFFF0000 01000001 80000000 00000028  .....(.........................
  0080 00FFFFFF FF0000FF FFFFFFFF FFFFFF00  00000000 05000000 C0000000 00010100  ................................
  00A0 00620201 00006803 01000000 44010062  60000060 00010003 00140000 00140000  .....h.........D..b-..-.........
  00C0 00000000 00000200 000001F9 00000000  00000000 00000000 00000000 00000000  ..............9.................
  00E0 00000000 00000000 00000001 00000000  00000000 00000000 00000000 00000000  ................................
  0100 00002800 00000000 00000000 00000000  0006C300 00170327 30502009 F3F3F3F3  ..(.............C......&..3333
  0120 F0F10000 80010000 00000000 00000000  28000000 02000014 00140000 02000000  01..........(.....  ............
  0140 00001400 01000200 01000200 01000100  00000000 0027FF00 00000000 00000000  .....................'..........

PATH - PATH01

000010 0000001A 01F3F3F3 F3F0F100 00100000  0000861D 8A4E0000 00000000 00000000  .....333301.........+..........
  0020 00000000 00000000 00000000 D9009800  6CD7C1E3 C8F0F140 40404040 40404040  ............R...(PATH01
  0040 40404040 40404040 40404040 40404040  40404040 40404040 40404040 40FFFFFF                              ...
  0060 FFFFFFFF FF74280F 00000F80 00000000  00000000 00C00000 00000201 00000602  .....(.........................
  0080 0200000C 02030006 C3000017 0006C400  00180006 C9000019 00000000 00000000  ........C.....D.....I...........
```

Figure 26 (Part 1 of 2). An Example of LISTCRA DUMP NOCOMPARE Output

```
  AIX  - AIX01
    DATA VOL -
7 ->333301
    INDX VOL -
      333301
  UPGD -

000014 00000015 01F3F3F3 F3F0F100 00143050  2000861D 8A4E0000 00000000 00000000   .....333301....&.....+.........
  0020 00000000 00000000 00000000 E8004B00  31000000 00000200 0000C000 00000002   ............Y...................
  0040 01000AC4 00001DC9 00001D00 00000000  00000000 00000000 00000000 00000000   ..D..I.....................

  DATA - TF081480.VSAMDSET.DFD74280.T861DB0E.TF081480

000038 00000048 01F3F3F3 F3F0F100 00383050  2009861D 8A4E861D B0F00000 00000000   .....333301....&.....+...0......
  0020 00000000 00000000 00000000 C401C500  8FE3C6F0 F8F1F4F8 F04BE5E2 C1D4C4E2   ............D.E..TF081480.VSAMDS
  0040 C5E34BC4 C6C4F7F4 F2F8F04B E3F8F6F1  C4C2F0C5 4BE3C6F0 F8F1F4F8 F0FFFFFF   ET.DFD74280.T861DB0E.TF081480...
  0060 FFFFFFFF FF74280F 00000000 20000000  06000000 01000001 80000000 00000050   .....&.........................
  0080 00000000 500000FF FFFFFFFF FFFFFF00  00000000 06000000 C0000000 00010100   ....&...........................
  00A0 00620201 00006803 010000BD 03020000  00440100 62608400 60000000 0A000A00   .....-.-.......-.-.............
  00C0 00001400 00000000 00000002 00000000  50000000 00000000 00000000 00000000   ................&...............
  00E0 00000000 00000000 00000000 00000000  02000000 00000000 00000000 00000000   ................................
  0100 00000000 00000050 00000000 00000000  00000000 000006C3 00004703 27305020   .......&.............C......&.
  0120 09F3F3F3 F3F0F100 00800100 00000000  00000000 00280000 00020000 14000140   .333301.....................
  0140 0018000A C1C1C1C1 C1C1C1C1 C1C1000A  C1C1C1C1 C1C1C1C1 C5F90014 00010003   ....AAAAAAAAAA..AAAAAAAAE9.....
  0160 00040003 00040001 00000000 000027FF  03273050 2009F3F3 F3F3F0F1 00008001   ..................&..333302....
  0180 00000000 00002800 00005000 00000200  00140009 40001800 0AC1C1C1 C1C1C1C1   ..........&.....................
  01A0 C1C6C100 0AC1C1C1 C1C1C1C1 C1E9E900  14000100 03000500 03000500 01000028   AFA..AAAAAAAAZZ.................
  01C0 0000004F FF000000 00000000 00000000  00000000 00000000 00000000 00000000   ... ............................

  INDX - TF082370.VSAMDSET.DFD74280.T861DB0E.TF082370

00003A 00000049 01F3F3F3 F3F0F100 003A3050  2009861D 8A4E861D B0F00000 00000000   .....333301....&.....+...0......
  0020 00000000 00000000 00000000 C9018900  8FE3C6F0 F8F2F3F7 F04BE5E2 C1D4C4E2   ............I....TF082370.VSAMDS
  0040 C5E34BC4 C6C4F7F4 F2F8F04B E3F8F6F1  C4C2F0C5 4BE3C6F0 F8F2F3F7 F0FFFFFF   ET.DFD74280.T861DB0E.TF082370...
  0060 FFFFFFFF FF74280F 00000000 2000FFFF  FFFF0000 01000001 80000000 00000028   .....&.........................
  0080 00FFFFFF FF0000FF FFFFFFFF FFFFFF00  00000000 06000000 C0000000 00010100   ....&...........................
  00A0 00620201 00006803 010000A9 03020000  00440100 62600400 60000400 0A000A00   .....-.-.......-.-.............
  00C0 00001400 00000000 00000002 00000001  F9000000 00000000 00000000 00000000   ................9...............
  00E0 00000000 00000000 00000000 00000000  01000000 00000000 00000000 00000000   ................................
  C100 00000000 00000028 00000000 00000000  00000000 000006C3 00004703 27305020   .....&.....................0......&.
  0120 09F3F3F3 F3F0F100 00800100 00000000  00000000 00280000 00020000 14000140   .333301.....................
  0140 00190000 00000014 00010003 00060003  00060001 00000000 000027FF 03273050   .....&.....................&
  0160 2009F3F3 F3F3F0F1 00004000 00000000  00000000 00000000 00000200 00140001   ..333301.. .....................
  0180 40001900 00000000 00000000 00000000  00000000 00000000 00000000 00000000   ................................
```

8 9 8 9 10

LISTCRA DUMP NOCOMPARE Output Notes:

Note: When you specify SEQUENTIALDUMP, the same information is listed. The entries are not sorted according to type of entry, or according to alphanumeric sequence of the entryname field. See also note 7 below.

1. The 3-byte CRA control interval number, in hexadecimal form.
2. The 3-byte control interval number of the catalog record for which this is a copy, in hexadecimal form.
3. The entry-type at offset X'2C' (decimal 44), a character that identifies the type of catalog entry being listed.
4. A 44-byte field at offset X'31' (decimal 49) that contains the entry's entryname (padded with binary zero), or a 8-byte field followed by 36 bytes of binary zero that contains the volume entry's volume serial number.
5. The volume serial number of the recovery volume.
6. A 2-byte displacement value. Printing is suppressed after the last line containing data is listed, even though all catalog records are 512 bytes long.
7. As in the NAME NOCOMPARE output, the alternate index name and the fact that there is an upgrade set is given, along with the volume serial numbers of the alternate index's data and index components. (Note: If SEQUENTIALDUMP is specified, the information described with this note is not listed.)
8. The first byte identifies the alternate index's data component and the next three bytes contain its catalog control interval number.

9 The first byte identifies the alternate index's index component and the next three bytes contain its catalog control interval number.

10. Since there are two key ranges for this cluster, there are two volume information sets of fields in the cluster's data and index component entries.

Figure 26 (Part 2 of 2). An Example of LISTCRA DUMP NOCOMPARE Output

```
                    LISTING OF CATALOG RECOVERY AREA FOR VOLUME -- 333301 -- VSAM ENTRIES

(1)   CATVOLRCD  -  09/17/42   23:53:47
      CRAVOLRCD  -  10/07/74   23:35:00
      F4DSCBVSAM -  10/07/74   23:35:00
      F4DSCBDUMP -  10/07/74   23:35:00

CLUS - LR.DELETED.ESDS

* MISCOMPARES - CATALOG ENTRY HAS DIFFERENT NAME

CLUS - LR.MCHURBA.ESDS

  DATA - T73FEDA0.VSAMDSET.DFD74280.T861DAE1.T73FEDA0
    DATA VOL -
      333301

(2)* MISCOMPARES - HIGH USED RBA

CLUS - LR.MCKEYRNG.KSDS

  DATA - TF081480.VSAMDSET.DFD74280.T861DB0E.TF081480
    DATA VOL - HIGH KEY
      333301 - C1C1C1C1C1C1C1C5F9
      333301 - C1C1C1C1C1C1C1E9E9

* MISCOMPARES - HIGH USED RBA

  INDX - TF082370.VSAMDSET.DFD74280.T861DB0E.TF082370
    INDX VOL -
      333301
      333301

* MISCOMPARES - STATISTICS

NUMBER OF ENTRIES PROCESSED
 CLUS - 23
 DATA - 18
 AIX  -  8
 INDX - 11
 PATH -  5
 VOL  -  1
 UPGD -  3
 SUM  - 69

IDC0665I NUMBER OF ENTRIES THAT MISCOMPARED IN THIS CRA - 3

(3)IDC0877I NUMBER OF RECORDS THAT MISCOMPARED IN THIS CRA - 4
```

LISTCRA NAME COMPARE Output Notes:

1. **In addition to the timestamps described for the NAME NOCOMPARE output (noted previously), the timestamp in the catalog volume record, CATVOLRCD, is listed. CATVOLRCD is updated in the same manner as the CRAVOLRCD timestamp.**
2. **The MISCOMPARES message always refers to the record listed above it. See "Regaining Access to Data" under "Catalog Recovery" in the chapter on Data Security and Protection for the cause and seriousness of this MISCOMPARES message.**
3. **The number of records is sometimes larger than the number of entries, because an entry might consist of a catalog record and one or more extension records that failed to compare correctly.**

Figure 27. An Example of LISTCRA NAME COMPARE Output

```
                    LISTING OF CATALOG RECOVERY AREA FOR VOLUME -- 333301 -- VSAM ENTRIES

CLUS - LR.MCHURBA.ESDS

000020 0000002C 01F3F3F3 F3F0F100 00203050  2009861D 8A4E0000 00000000 00000000   .....333301....&.....+.........
CATRCD 0000002C 01F3F3F3 F3F0F100 00203050  2009861D 8A4E0000 00000000 00000000   .....333301....&.....+.........

  0020 00000000 00000000 00000000 C3008200  6CD3D94B D4C3C8E4 D9C2C14B C5E2C4E2   ............C...(LR.MCHURBA.ESDS
       00000000 00000000 00000000 C3008200  6CD3D94B D4C3C8E4 D9C2C14B C5E2C4E2   ............C...(LR.MCHURBA.ESDS

  0040 40404040 40404040 40404040 40404040  40404040 40404040 40404040 40FFFFFF                               ...
       40404040 40404040 40404040 40404040  40404040 40404040 40404040 40FFFFFF                               ...

  0060 FFFFFFFF FF74280F 00000F00 00000000  00000000 00020100 00004401 0006C400   ..........................D.
       FFFFFFFF FF74280F 00000F00 00000000  00000000 00020100 00004401 0006C400   ..........................D.

  0080 002D0000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   ................................
       002D0000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   ................................

  DATA - T73FEDAO.VSAMDSET.DFD74280.T861DAE1.T73FEDAO

000021 0000002D 01F3F3F3 F3F0F100 00213050  2009861D 8A4E861D AE2E0000 00000000   .....333301....&.....+.........
CATRCD 0000002D 01F3F3F3 F3F0F100 00213050  2009861D 8A4E861D AE2E0000 00000000   .....333301....&.....+.........

  0020 00000000 00000000 00000000 C4015700  8FE4F7F3 C6C5C4C1 F04BE5E2 C1D4C4E2   ............D....T73FEDAO.VSAMDS
       00000000 00000000 00000000 C4015700  8FE4F7F3 C6C5C4C1 F04BE5E2 C1D4C4E2   ............D....T73FEDAO.VSAMDS

  0040 C5E34BC4 C6C4F7F4 F2F8F04B E3F8F6F1  C4C1C5F1 4BE3F7F3 C6C5C4C1 F0FFFFFF   ET.DFD74280.T861DAE1.T73FEDAO...
       C5E34BC4 C6C4F7F4 F2F8F04B E3F8F6F1  C4C1C5F1 4BE3F7F3 C6C5C4C1 F0FFFFFF   ET.DFD74280.T861DAE1.T73FEDAO...

  0060 FFFFFFFF FF74280F 00000000 20000000  04000000 01000001 80000000 00000028   ................................
       FFFFFFFF FF74280F 00000000 20800000  04000000 01000001 80000006 00000028   ................................
*                                   **                                    **                                  *
  0080 00000000 500000FF FFFFFFFF FFFFFF00  00000000 05000000 00000000 00010100   ....&...........................
       00000000 500000FF FFFFFFFF FFFFFF00  00000000 05000000 C0000000 00010100   ....&...........................

  00A0 00620201 00006803 01000000 44010062  60000060 00000000 00000000 00140000   ................-..-............
       00620201 00006803 01000000 44010062  60000060 00000000 00000000 00140000   ................-..-............

* MISCOMPARES - HIGH USED RBA
```

LISTCRA DUMP COMPARE Output Notes:

1. The 3-byte CRA control interval number in hexadecimal form.
2. Each pair of hexadecimal-data lines are the information in the catalog recovery area's copy of the catalog record, followed by the information in the catalog record, for the same displacement.
3. Asterisks are printed below the bytes in which a miscompare exists and, in the left margin, the miscompared lines are flagged with a single asterisk.
4. See "Regaining Access to Data" under "Catalog Recovery" in the chapter on Data Security and Protection for the cause and seriousness of this MISCOMPARES message.

Figure 28. An Example of LISTCRA DUMP COMPARE Output

APPENDIX E: SAMPLE OUTPUT FROM CHKLIST

Figure 29 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       01/16/75
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 29. An Example of CHKLIST Output

APPENDIX F: COMMAND PARAMETERS SUMMARY

This section contains tables that show each Access Method Services functional command's parameters, and the parameter's abbreviation, default value (if any), and example of usage. The command summaries are grouped together to allow you to remove these pages and use them for a quick reference.

ALTER Parameters: Summary

The following table shows which keywords can be used with the ALTER command.

Parameter	Abbr	Example	Notes
ALTER	—	ALTER	
Required parameter			
entryname	—	OLDDATA/OLDPASSM	
Optional parameters			
Name parameters			
CATALOG	CAT	CAT(SUPERCAT/SCATM)	
FILE	—	FILE(DD10)	
NEWNAME	NEWNM	NEWNM(NEWDATA)	
Protection and integrity parameters			
ATTEMPTS	ATT	ATT(3)	
AUTHORIZATION	AUTH	AUTH(SECURE CODE10)	
CODE	—	CODE(SWIM)	1
CONTROLPW	CTLPW	CTLPW(FINS)	1
DESTAGEWAIT	DSTGW	DSTGW	
ERASE	ERAS	ERAS	
EXCEPTIONEXIT	EEXT	EEXT(OOPSIO)	1
FOR	—	FOR(756)	
INHIBIT	INH	INH	
MASTERPW	MRPW	MRPW(SCUBA)	1
MODULE	MDLE	NULL(AUTH(MDLE))	3
NODESTAGEWAIT	NDSTGW	NDSTGW	
NOERASE	NERAS	NERAS	
NOWRITECHECK	NWCK	NWCK	
NULLIFY	NULL	NULL(RETN,CODE)	4
OWNER	—	OWNER(BIGCHIEF)	1
READPW	RDPW	RDPW(MASK)	1
RETENTION	RETN	NULL(RETN)	2
SHAREOPTIONS	SHR	SHR(3 3)	
STRING	STRG	NULL(AUTH(STRG))	3
TO	—	TO(75365)	
UNINHIBIT	UNINH	UNINH	
UPDATEPW	UPDPW	UPDPW(TRUNKS)	1
WRITECHECK	WCK	WCK	

Parameter	Abbr	Example	Notes
Allocation parameters			
ADDVOLUMES	AVOL	AVOL(PCVOL5)	
BIND	—	BIND	
BUFFERSPACE	BUFSP, BUFSPC	BUFSP(4096)	
CYLINDERFAULT	CYLF	CYLF	
FREESPACE	FSPC	FSPC(20 25)	
KEYS	—	KEYS(10 4)	
RECORDSIZE	RECSZ	RECSZ(121 121)	
REMOVEVOLUMES	RVOL	RVOL(PCVOL4)	
STAGE	—	STAGE	
Alternate Index and Path parameters			
NONUNIQUEKEY	NUNQK	NUNQK	
NOUPDATE	NUPD	NUPD	
NOUPGRADE	NUPG	NUPG	
UNIQUEKEY	UNQK	UNQK	
UPDATE	UPD	UPD	
UPGRADE	UPG	UPG	
Generation Data Group parameters			
EMPTY	EMP	EMP	
NOEMPTY	NEMP	NEMP	
NOSCRATCH	NSCR	NSCR	
SCRATCH	SCR	SCR	

Notes:

1. This parameter can also be a subparameter of NULLIFY.
2. RETENTION is a subparameter of NULLIFY.
3. MODULE and STRING are subparameters of NULLIFY (AUTHORIZATION).
4. See notes 1, 2, and 3.

BLDINDEX Parameters: Summary

The following table shows which keywords can be used with the BLDINDEX command.

Parameter	Abbr	Example	Notes
BLDINDEX	BIX	BIX ...	
Required parameters			
INDATASET	IDS	IDS(BASE.CLUSTER)	1
INFILE	IFILE	IFILE(DDBASE)	
OUTDATASET	ODS	ODS(AIX.OR.PATH)	2
OUTFILE	OFILE	OFILE(DDPATH)	2
Optional parameters			
CATALOG	CAT	CAT(UCAT1/MASTPSWD)	
EXTERNALSORT	ESORT	ESORT	4
INTERNALSORT	ISORT	ISORT	4
WORKFILES	WFILE	WFILE(DDWF1 DDWF2)	3

Notes:

See the parameter's description for restrictions.

1. Either INDATASET or INFILE must be specified.
2. Either OUTDATASET or OUTFILE must be specified.
3. If WORKFILE is specified, you must supply two DD statements.
4. When EXTERNALSORT is not specified, INTERNALSORT is the default.

CHKLIST Parameters: Summary

The following table shows which keywords can be used with the CHKLIST command.

Parameter	Abbr	Example
CHKLIST	CKLST	CKLST ...
Required parameter		
INFILE	IFILE	IFILE(DD1)
Optional parameters		
CHECKID	CHKID	CHKID(START 12)
OUTFILE	OFILE	OFILE(DD2)

CNVTCAT Parameters: Summary

The following table shows which keywords can be used with the CNVTCAT command.

Parameter	Abbr	Example	Notes
CNVTCAT	CNVTC	CNVTC...	
Required parameters			
INDATASET	IDS	IDS(OSCAT2)	1
INFILE	IFILE	IFILE(DDCAT2)	1
Optional parameters			
CATALOG	CAT	CAT(USERCAT1)	
CVOLEQUATES	CVEQU	CVEQU((UCAT2(VSER08)))	3
LIST	—	LIST	2
MASTERCATALOG	MCAT	MCAT(AMASTCAT)	
NOLIST	NLIST	NLIST	2

Notes:

1. Either INDATASET or INFILE must be specified.
2. If NOLIST is not specified, LIST is the default.
3. If CVOLEQUATES is specified, MASTERCATALOG must also be specified.

DEFINE ALIAS Parameters: Summary

The following table shows which keywords can be used with the DEFINE ALIAS command.

Parameter	Abbr	Example
DEFINE	DEF	
Required parameters		
ALIAS	—	DEF ALIAS(...)
NAME	—	NAME(PAYROLL.JUNE.CURRENT)
RELATE	REL	REL(PAYROLL.G0022V05)
Optional parameters		
CATALOG	CAT	CAT(MGMTCAT1/JPMORGAN)

DEFINE ALTERNATEINDEX Parameters: Summary

The following table shows which keywords can be used with the DEFINE ALTERNATEINDEX command. The "Usage" field specifies whether the parameter can be used with ALTERNATEINDEX (A), DATA (D), or INDEX (I).

Parameter	Abbr	Usage	Example	Notes
DEFINE	DEF			
Required parameters				
ALTERNATEINDEX	AIX	—	DEF AIX(...)	
NAME	—	A,D,I	NAME(PAYROLL.NAMES)	
RELATE	REL	A	REL(PAYROLL.MASTER)	
Allocation parameters				
CYLINDERS	CYL	A,D,I	CYL(10 2)	1
RECORDS	REC	A,D,I	REC(800 100)	1
TRACKS	TRK	A,D,I	TRK(100 20)	1
VOLUMES	VOL	A,D,I	VOL(VOL001 VOL002)	
Optional parameters				
CATALOG	CAT	A	CAT(MGMTCAT/CHIEFPW)	
DATA	—	A	DATA(...)	
INDEX	IX	A	IX(...)	
Model parameter				
MODEL	—	A,D,I	MODEL(PAYROLL.INIT)	
Allocation parameters				
• For the Alternate Index As a Whole				
CONTROLINTERVALSIZE	CISZ, CNVSZ	A,D,I	CISZ(4096)	2
FILE	—	A,D,I	FILE(DDPAY)	
NONUNIQUEKEY	NUNQK	A,D	NUNQK	3
NOREUSE	NRUS	A,D,I	NRUS	4
ORDERED	ORD	A,D,I	ORD	5
REUSE	RUS	A,D,I	RUS	4
SUBALLOCATION	SUBAL	A,D,I	SUBAL	6
UNIQUE	UNQ	A,D,I	UNQ	6
UNIQUEKEY	UNQK	A,D	UNQK	3
UNORDERED	UNORD	A,D,I	UNORD	5
• For Data Record Identification and Location				
KEYRANGES	KRNG	A,D	KRNG((A M)(N Z))	
KEYS	—	A,D	KEYS(8 8)	2, 14
• For Sequence-Set Record Location				
IMBED	IMBD	A,I	IMBD	11
NOIMBED	NIMBD	A,I	NIMBD	11
NOREPLICATE	NREPL	A,I	NREPL	12
REPLICATE	REPL	A,I	REPL	12
• For Optimized Record Processing				
BUFFERSPACE	BUFSP, BUFSPC	A,D	BUFSP(8704)	2
FREESPACE	FSPC	A,D	FSPC(10 10)	
RECORDSIZE	RECSZ	A,D	RECSZ(121 121)	2,15

Parameter	Abbr	Usage	Example	Notes
• When the Alternate Index Is On a Mass Storage Volume				
BIND	—	A,D,I	BIND	17
CYLINDERFAULT	CYLF	A,D,I	CYLF	17
STAGE	—	A,D,I	STAGE	17
Data integrity parameters:				
• When the Alternate Index's Base Cluster Is Opened				
NOUPGRADE	NUPG	A	NUPG	13
UPGRADE	UPG	A	UPG	13
• When Records Are Written Into the Alternate Index				
NOWRITECHECK	NWCK	A,D,I	NWCK	8
RECOVERY	RCVY	A,D	RCVY	9
SPEED	—	A,D	SPEED	9
WRITECHECK	WCK	A,D,I	WCK	8
• When the Alternate Index Is Deleted				
ERASE	ERAS	A,D	ERAS	7
NOERASE	NERAS	A,D	NERAS	7
• When the Alternate Index Is On a Mass Storage Volume				
DESTAGEWAIT	DSTGW	A,D,I	DSTGW	16
NODESTAGEWAIT	NDSTGW	A,D,I	NDSTGW	16
• When the Alternate Index Is Shared Between Users				
SHAREOPTIONS	SHR	A,D,I	SHR(1 3)	
• When an I/O Error Occurs				
EXCEPTIONEXIT	EEXT	A,D,I	EEXT(PAYERROR)	
Protection parameters				
ATTEMPTS	ATT	A,D,I	ATT(4)	10
AUTHORIZATION	AUTH	A,D,I	AUTH(HEYJUDE)	
CODE	—	A,D,I	CODE(BEATLES)	
CONTROLPW	CTLPW	A,D,I	CTLPW(PAUL)	
MASTERPW	MRPW	A,D,I	MRPW(JOHN)	
READPW	RDPW	A,D,I	RDPW(RINGO)	
UPDATEPW	UPDPW	A,D,I	UPDPW(GEORGE)	
Ownership/Retention parameters				
FOR	—	A	FOR(380)	
OWNER	—	A,D,I	OWNER(APPLE)	
TO	—	A	TO(75340)	

Notes:

1. You must specify one of CYLINDERS, RECORDS, or TRACKS as a parameter of ALTERNATEINDEX or data.
2. The example-values specified for RECORDSIZE, CONTROLINTERVALSIZE, and BUFFERSPACE represent an alternate index that contains records 121 bytes long, with 33 records per control interval, and buffer space for two 4096-byte data control intervals.

Notes 3 through 15 identify the default values when the parameter is not specified as a parameter of ALTERNATEINDEX:

3. NONUNIQUEKEY, when UNIQUEKEY is not specified.
4. NOREUSE, when REUSE is not specified.
5. UNORDERED, when ORDERED is not specified.
6. SUBALLOCATION, when UNIQUE is not specified.
7. NOERASE, when ERASE is not specified.
8. NOWRITECHECK, when WRITECHECK is not specified.
9. RECOVERY, when SPEED is not specified.

10. ATTEMPTS(2), when ATTEMPTS("number") is not specified.
11. NOIMBED, when IMBED is not specified.
12. NOREPLICATE, when REPLICATE is not specified.
13. UPGRADE, when NOUPGRADE is not specified.
14. KEYS(64 0), or the first 64 bytes of each data record is the alternate key, when KEYS is not specified.
15. RECORDSIZE(4086 32600), when RECORDSIZE(average maximum) is not specified.
16. NODESTAGEWAIT, when neither DESTAGEWAIT nor NODESTAGEWAIT is specified.
17. STAGE, when neither STAGE, BIND, nor CYLINDERFAULT is specified.

DEFINE CLUSTER Parameters: Summary

The following table shows which keywords can be used with the DEFINE CLUSTER command. The "Usage" column specifies whether the parameter can be used with CLUSTER (C), DATA (D), or INDEX (I).

Parameter	Abbr	Usage	Example	Notes
DEFINE	DEF			
Required parameters				
CLUSTER	CL	—	DEF CL(...)	
Allocation parameters				
CYLINDERS	CYL	C,D,I	CYL(10 2)	1
NAME	—	C,D,I	NAME(PAYROLL.MASTER)	
RECORDS	REC	C,D,I	REC(800 100)	1
TRACKS	TRK	C,D,I	TRK(100 20)	1
VOLUMES	VOL	C,D,I	VOL(VOL001 VOL002)	2
Optional parameters				
CATALOG	CAT	C	CAT(MGMTCAT/CHIEFPW)	
DATA	—	C	DATA(...)	
INDEX	IX	C	IX(...)	
Data organization parameters				
INDEXED	IXD	C	IXD	13
NONINDEXED	NIXD	C	NIXD	13
NUMBERED	NUMD	C	NUMD	13
Model parameter				
MODEL	—	C,D,I	MODEL(PAYROLL.INIT)	
Allocation parameters				
• For the Cluster As a Whole				
CONTROLINTERVALSIZE	CISZ, CNVSZ	C,D,I	CISZ(4096)	18
FILE	—	C,D,I	FILE(DDPAY)	
NOREUSE	NRUS	C,D,I	NRUS	4
REUSE	RUS	C,D,I	RUS	4
SUBALLOCATION	SUBAL	C,D,I	SUBAL	6
UNIQUE	UNQ	C,D,I	UNQ	6
• For the Key-Sequenced Cluster As a Whole				
FREESPACE	FSPC	C,D	FSPC(10 10)	
ORDERED	ORD	C,D,I	ORD	5
UNORDERED	UNORD	C,D,I	UNORD	5

Parameter	Abbr	Usage	Example	Notes
• For Optimized Record Processing				
BUFFERSPACE	BUFSP, BUFSPC	C,D	BUFSP(8704)	18
NONSPANNED	NSPND	C,D,	NSPND	3
RECORDSIZE	RECSZ	C,D	RECSZ(121 121)	15,18
SPANNED	SPND	C,D	SPND	3
• For a Cluster On a Mass Storage Volume				
BIND	—	C,D,I	BIND	17
CYLINDERFAULT	CYLF	C,D,I	CYLF	17
STAGE	—	C,D,I	STAGE	17
Data integrity parameters				
• When the Records Are Written Into the Cluster				
NOWRITECHECK	NWCK	C,D,I	NWCK	8
RECOVERY	RCVY	C,D	RCVY	9
SPEED	—	C,D	SPEED	9
WRITECHECK	WCK	C,D,I	WCK	8
• When the Cluster Is Shared Between Users				
SHAREOPTIONS	SHR	C,D,I	SHR(1 3)	
• When the Cluster Is Deleted				
ERASE	ERAS	C,D	ERAS	7
NOERASE	NERAS	C,D	NERAS	7
• When the Cluster Is On a Mass Storage Volume				
DESTAGEWAIT	DSTGW	C,D,I	DSTGW	16
NODESTAGEWAIT	NDSTGW	C,D,I	NDSTGW	16
• When an I/O Error Occurs				
EXCEPTIONEXIT	EEXT	C,D,I	EEXT(PAYERROR)	
Protection parameters				
ATTEMPTS	ATT	C,D,I	ATT(4)	10
AUTHORIZATION	AUTH	C,D,I	AUTH(SARGEANT)	
CODE	—	C,D,I	CODE(TROOPER)	
CONTROLPW	CTLPW	C,D,I	CTLPW(COMPANY)	
MASTERPW	MRPW	C,D,I	MRPW(BRIGADE)	
READPW	RDPW	C,D,I	RDPW(SQUAD)	
UPDATEPW	UPDPW	C,D,I	UPDPW(PLATOON)	
Ownership/Retention parameters				
FOR	—	C	FOR(380)	
OWNER	—	C,D,I	OWNER(BRASS)	
TO	—	C	TO(75340)	
Parameters for key-sequenced clusters only				
• For Data Record Identification and Location				
KEYRANGES	KRNG	C,D	KRNG((A E)(F M)(N Z))	2
KEYS	—	C,D	KEYS(8 8)	14
• For Sequence-Set Record Location				
IMBED	IMBD	C,I	IMBD	11
NOIMBED	NIMBD	C,I	NIMBD	11
NOREPLICATE	NREPL	C,I	NREPL	12
REPLICATE	REPL	C,I	REPL	12

Notes:

See the parameter's description for details.

1. You must specify one of CYLINDERS, RECORDS, or TRACKS as a parameter of CLUSTER.
2. When more key ranges than volumes are specified (as shown in the VOLUMES and KEYRANGES examples), the excess key ranges are allocated on the last volume specified. That is, volume VOL001 contains key range A-E, and volume VOL002 contains key ranges F-M and N-Z.

Notes 3 through 17 identify the default values when the parameter is not specified as a parameter of CLUSTER:

3. NONSPANNED, when SPANNED is not specified.
4. NOREUSE, when REUSE is not specified.
5. UNORDERED, when ORDERED is not specified.
6. SUBALLOCATION, when UNIQUE is not specified.
7. NOERASE, when ERASE is not specified.
8. NOWRITECHECK, when WRITECHECK is not specified.
9. RECOVERY, when SPEED is not specified.
10. ATTEMPTS(2), when ATTEMPTS(*number*) is not specified.
11. NOIMBED, when IMBED is not specified.
12. NOREPLICATE, when REPLICATE is not specified.
13. INDEXED, when neither NONINDEXED nor NUMBERED is specified.
14. KEYS(64 0), or the first 64 bytes of each record is the record's key value when KEYS is not specified for a key-sequenced data set.
15. RECORDSIZE (4086 32600), when SPANNED is also specified; otherwise, RECORDSIZE(4089 4089).
16. NODESTAGEWAIT, when neither DESTAGEWAIT nor NODESTAGEWAIT is specified.
17. STAGE, when neither STAGE, BIND, nor CYLINDERFAULT is specified.
18. The example-values specified for RECORDSIZE, CONTROLINTERVALSIZE, and BUFFERSPACE represent a relative-record cluster that contains fixed-length records 121 bytes long, with 33 records per control interval, and buffer space for two 4096-byte data control intervals.

DEFINE GENERATIONDATAGROUP Parameters: Summary

The following table shows which keywords can be used with the DEFINE GENERATIONDATAGROUP command.

Parameter	Abbr	Example	Notes
DEFINE	DEF		
Required parameters			
GENERATIONDATAGROUP	GDG	DEF GDG(...)	
LIMIT	LIM	LIMIT(25)	
NAME	—	NAME(DATA,GROUP)	
Optional parameters			
CATALOG	CAT	CAT(USERCAT2)	
EMPTY	EMP	EMP	1
NOEMPTY	NEMP	NEMP	1
NOSCRATCH	NSCR	NSCR	2
SCRATCH	SCR	SCR	2
FOR	—	FOR(188)	
OWNER	—	OWNER(BREAD)	
TO	—	TO(75007)	

Notes:

See the parameter's description for restrictions.

1. When EMPTY is not specified, NOEMPTY is the default.
2. When SCRATCH is not specified, NOSCRATCH is the default.

DEFINE NONVSAM Parameters: Summary

The following table shows which keywords can be used with the DEFINE NONVSAM command.

Parameter	Abbr	Example
DEFINE	DEF	
Required parameters		
NONVSAM	NVSAM	DEF NVSAM(...)
DEVICETYPES	DEVT	DEVT(3340)
NAME	—	NAME(NONVSAM.DATA.SET)
VOLUMES	VOL	VOL(DISK20)
Optional parameters		
CATALOG	CAT	CAT(UCAT125)
FILESEQUENCENUMBERS	FSEQN	FSEQN(3 4)
FOR	—	FOR(180)
OWNER	—	OWNER(WONDER)
TO	—	TO(75002)

DEFINE PAGESPACE Parameters: Summary

The following table shows which keywords can be used with the DEFINE PAGESPACE command.

Parameter	Abbr	Example	Notes
DEFINE	DEF		
Required parameters			
PAGESPACE	PGSPC	DEF PGSPC (...)	
CYLINDERS	CYL	CYL(20)	1
NAME	—	NAME(SYS1.PAGE01)	
RECORDS	REC	REC(400)	1
TRACKS	TRK	TRK(100)	1
VOLUME	VOL	VOL(DSK135)	
Optional parameters			
CATALOG	CAT	CAT(AMASTCAT)	
FILE	—	FILE(DDPGSPC)	
MODEL	—	MODEL(PGSPC1A/MPWPS1A)	
NOSWAP	NSWAP	NSWAP	3
SUBALLOCATION	SUBAL	SUBAL	2
SWAP	—	SWAP	3
UNIQUE	UNQ	UNQ	2
Protection and integrity parameters			
ATTEMPTS	ATT	ATT(0)	
AUTHORIZATION	AUTH	AUTH(PGSPCCHK)	
CODE	—	CODE(BEATLES)	
CONTROLPW	CTLPW	CTLPW(PAUL)	
FOR	—	FOR(180)	
MASTERPW	MRPW	MRPW(JOHN)	
OWNER	—	OWNER(TEENY.BOPPER)	
READPW	RDPW	RDPW(GEORGE)	
TO	—	TO(75187)	
UPDATEPW	UPDPW	UPDPW(RINGO)	

Notes:

See the parameter's description for restrictions and defaults.

1. You must specify one of: CYLINDERS, RECORDS, or TRACKS.
2. When SUBALLOCATION is not specified, UNIQUE is the default.
3. When SWAP is not specified, NOSWAP is the default.

DEFINE PATH Parameters: Summary

The following table shows which keywords can be used with the DEFINE PATH command.

Parameter	Abbr	Example	Notes
DEFINE	DEF		
Required parameters			
PATH	—	DEF PATH(...)	
NAME	—	NAME(PATH.FINDER)	
PATHENTRY	PENT	PENT(AIX.FINDER)	
Optional parameters			
CATALOG	CAT	CAT(USER1CAT)	
Allocation parameter			
FILE	—	FILE(STEPCAT)	
Model parameter			
MODEL	—	MODEL(BEATEN.PATH)	
Update parameters			
NOUPDATE	NUPD	NUPD	1
UPDATE	UPD	UPD	1
Protection parameters			
ATTEMPTS	ATT	ATT(3)	2
AUTHORIZATION	AUTH	AUTH(PATHCHK X'F4D8')	
CODE	—	CODE(FOLKSONG)	
CONTOLPW	CTLPW	CTLPW(PAUL)	
MASTERPW	MRPW	MRPW(PETER)	
READPW	RDPW	RDPW(BANJO)	
UPDATEPW	UPDPW	UPDPW(MARY)	
Owner/Retention parameters			
FOR	—	FOR(365)	
OWNER	—	OWNER(MELLOW)	
TO	—	TO(75001)	

Notes:

1. When NOUPDATE is not specified, UPDATE is the default.
2. When ATTEMPTS is not specified, ATTEMPTS(2) is the default.

DEFINE SPACE Parameters: Summary

The following table shows which keywords can be used with the DEFINE SPACE command.

Parameter	Abbr	Example	Notes
DEFINE	DEF		
Required parameters			
SPACE	SPC	DEF SPC (...)	
CANDIDATE	CAN	CAN	1
CYLINDERS	CYL	CYL(10 2)	1
RECORDS	REC	REC(1000 200)	1,2
RECORDSIZE	RECSZ	RECSZ(80 80)	2
TRACKS	TRK	TRK(200 40)	1
VOLUMES	VOL	VOL(PSV525)	
Optional parameter			
CATALOG	CAT	CAT(USERCAT1)	
FILE	—	FILE(DDSPC)	

Notes:

See the parameter's description for restrictions.

1. One, and only one, of the following parameters must be specified: CANDIDATE, CYLINDERS, RECORDS, or TRACKS.
2. If RECORDS is specified, you must also specify RECORDSIZE. Otherwise, you cannot specify RECORDSIZE.

DEFINE USERCATALOG Parameters: Summary

The following table shows which keywords can be used with the DEFINE USERCATALOG commands. The "Usage" column specifies whether the parameter can apply to the catalog as a whole (U), to the data component (D), or to the index component (I).

Parameter	Abbr	Usage	Example	Notes
DEFINE	DEF			
Required parameters				
USERCATALOG	UCAT	—	DEF UCAT(...)	
NAME	—	U	NAME(USERCAT.TWO5)	
Allocation parameters				
VOLUME	VOL	U	VOL(UVOL25)	
TRACKS	TRK	U,D,I	TRK(100 20)	2
CYLINDERS	CYL	U,D,I	CYL(10 2)	2
RECORDS	REC	U,D,I	REC(800 150)	2
Optional parameters				
CATALOG	CAT	—	CAT(MASTER.CATLG/ MWRITEPW)	
DATA	—	—	DATA(...)	
INDEX	IX	—	IX(...)	
Model parameter				
MODEL	—	U	MODEL(USERCAT.PRIME)	
Allocation parameters				
BUFFERSPACE	BUFSP	U,D	BUFSP(8192)	1
FILE	—	U	FILE(NEWCATDD)	
NOTRECOVERABLE	NRVBL	U,D	NRVBL	4
RECOVERABLE	RVBL	U,D	RVBL	4
• Protection parameters				
ATTEMPTS	ATT	U	ATT(1)	5
AUTHORIZATION	AUTH	U	AUTH(SPRSNOOP)	
CODE	—	U	CODE(CANDYBAR)	
CONTROLPW	CTLPW	U	CTLPW(TOFFEE)	
MASTERPW	MRPW	U	MRPW(CHOKLIT)	
READPW	RDPW	U	RDPW(COCONUT)	
UPDATEPW	UPDPW	U	UPDPW(CHUNKY)	
• Data integrity parameters				
DESTAGEWAIT	DSTGW	U,D,I	DSTGW	6
NODESTAGEWAIT	NDSTGW	U,D,I	NDSTGW	6
NOWRITECHECK	NWCK	U,D,I	NWCK	3
WRITECHECK	WCK	U,D,I	WCK	3
• Ownership/Retention parameters				
FOR	—	U	FOR(9999)	
OWNER	—	U	OWNER(STICKY)	
TO	—	U	TO(99360)	

Notes:

1. When no value is specified, the default is BUFFERSPACE(3072).
2. You must specify one of CYLINDERS, RECORDS, or TRACKS as a subparameter of USERCATALOG.
3. When WRITECHECK is not specified, the default is NOWRITECHECK.
4. When RECOVERABLE is not specified, the default is NOTRECOVERABLE.
5. When no value is specified, the default is ATTEMPTS(2).
6. When no value is specified, the default is NODESTAGEWAIT.

DELETE Parameters: Summary

The following table shows which keywords can be used with the DELETE command.

Parameter	Abbr	Example	Notes
DELETE	DEL	DEL ...	
Required parameter			
entryname	—	(NEWDATA/BOB)	
Optional parameters			
ALIAS	—	ALIAS	
ALTERNATEINDEX	AIX	AIX	
CATALOG	CAT	CAT(UCAT57/MASTPW)	
CLUSTER	CL	CL	
ERASE	ERAS	ERAS	
FILE	—	FILE(DDCLU)	
FORCE	FRC	FORCE	4
GENERATIONDATAGROUP	GDG	GDG	
NOERASE	NERAS	NERAS	
NOFORCE	NFRC	NFRC	4
NONVSAM	NVSAM	NVSAM	
NOPURGE	NPRG	NPRG	2
NOSCRATCH	NSCR	NSCR	3
PAGESPACE	PGSPC	PGSPC	
PATH	—	PATH	
PURGE	PRG	PRG	2
SCRATCH	SCR	SCR	3
SPACE	SPC	SPC	1
USERCATALOG	UCAT	UCAT	1

Notes:

1. When you delete a data space or catalog, you cannot delete any other type of entry. You must identify the type of entry to be deleted, by specifying SPACE or USERCATALOG.
2. Unless you specify PURGE, the default is NOPURGE.
3. Unless you specify NOSCRATCH, the default is SCRATCH.
4. Unless you specify FORCE, the default is NOFORCE.

EXPORT Parameters: Summary

The following table shows which keywords can be used with the EXPORT command.

Parameter	Abbr	Example	Notes
EXPORT	EXP	EXP ...	
Required parameters			
entryname	—	PAYROLL/PAYPW	
DISCONNECT	DCON	DCON	1
OUTDATASET	ODS	ODS(NEWDS)	2
OUTFILE	OFILE	OFILE(DDEXP2)	2
Optional parameters			
ERASE	ERAS	ERAS	9
INFILE	IFILE	IFILE(DDEXP1)	
INHIBITSOURCE	INHS	INHS	3, 4
INHIBITTARGET	INHT	INHT	5
NOERASE	NERAS	NERAS	
NOINHIBITSOURCE	NINHS	NINHS	3
NOINHIBITTARGET	NINHT	NINHT	5
NOPURGE	NPRG	NPRG	6
PERMANENT	PERM	PERM	8
PURGE	PRG	PRG	6, 7
TEMPORARY	TEMP	TEMP	4,7,8,9

Notes:

1. When "entryname" names a user catalog, DISCONNECT is required. Otherwise, DISCONNECT cannot be used.
2. When "entryname" names a cluster or alternate index, either OUTFILE or OUTDATASET is required.
3. When INHIBITSOURCE is not specified, NOINHIBITSOURCE is the default.
4. When INHIBITSOURCE is specified, TEMPORARY must also be specified.
5. When INHIBITTARGET is not specified, NOINHIBITTARGET is the default.
6. When PURGE is not specified, NOPURGE is the default.
7. You cannot specify PURGE and TEMPORARY together.
8. When TEMPORARY is not specified, PERMANENT is the default.
9. You cannot specify ERASE and TEMPORARY together.

EXPORTRA Parameters: Summary

The following table shows which keywords can be used with the EXPORTRA command.

Parameter	Abbr	Example	Notes
EXPORTRA	XPRA	XPRA ...	
Required parameters			
CRA	—	CRA((...)(...))	
ALL	—	CRA((DD1 ALL))	1
ENTRIES	ENT	CRA((DD1 ENT(MYDS)))	1
INFILE	IFILE	CRA((DD1 ALL IFILE (DD2)))	1,2
NONE	—	CRA((DD1 NONE))	1
OUTFILE	OFILE	OFILE(DD3)	3
Optional parameters			
FORCE	FRC	FRC	4
MASTERPW	MRPW	MRPW(MCATMPW)	5
NOFORCE	NFRC	NFRC	4

Notes:

1. DD1 identifies a DD statement that describes the catalog recovery area's volume.
2. DD2 identifies a (concatenated) DD statement that describes each volume that contains a part of the MYDS cluster. The catalog entries for MYDS cluster are contained in the DD1 volume's catalog recovery area.
3. DD3 identifies a DD statement that describes a moveable storage device (that is, a disk pack or magnetic tape reel.)
4. Unless you specify FORCE, the default option is NOFORCE.
5. When the master catalog is password protected, you must supply its master password.

IMPORT Parameters: Summary

The following table shows which keywords can be used with the IMPORT command.

Parameter	Abbr	Example	Notes
IMPORT	IMP	IMP ...	
Required parameters			
CONNECT	CON	CON	1
INDATASET	IDS	IDS(BACKUP.DS)	3
INFILE	IFILE	IFILE(DDBU)	2
OUTDATASET	ODS	ODS(OLD.DS)	2
OUTFILE	OFILE	OFILE(DDOLD)	3
Optional parameters			
CATALOG	CAT	CAT(USERCAT1/ UPDPW1)	
DEVICETYPE	DEVT	DEVT(3340)	4
ERASE	ERAS	ERAS	6
FILE	—	FILE(DDKSVOL)	4
INTOEMPTY	IEMPTY	IEMPTY	8
KEYRANGES	KRNG	KRNG((A M) (N Z))	4
NEWNAME	NEWNM	NEWNM(NEW.DS)	4
NOERASE	NERAS	NERAS	6
NOPURGE	NPRG	NPRG	6
NOSAVRAC	—	NOSAVRAC	7
OBJECTS	OBJ	OBJ((...))	
ORDERED	ORD	ORD	4, 5
PURGE	PRG	PRG	6
SAVRAC	—	SAVRAC	7
UNORDERED	UNORD	UNORD	4, 5
VOLUMES	VOL	VOL(KSVOL1)	4

Notes:

1. When a user catalog is imported, CONNECT is required. Otherwise, CONNECT cannot be used.
2. When a cluster or alternate index is imported, either INFILE or INDATASET is required to identify the portable data set.
3. When a cluster or alternate index is imported, either OUTFILE or OUTDATASET is required to identify the object that is to contain the imported data.
4. This parameter is a subparameter of OBJECTS.
5. When ORDERED is not specified, UNORDERED is the default.
6. ERASE or NOERASE, and PURGE or NOPURGE can be specified only for temporarily-exported clusters.
7. When SAVRAC is not used, NOSAVRAC is the default.
8. SAVRAC, NOSAVRAC parameters when used with INTOEMPTY, apply only to paths defined over the empty data set.

IMPORTRA Parameters: Summary

The following table shows which keywords can be used with the IMPORTRA command.

Parameter	Abbr	Example	Notes
IMPORTRA	MPRA	MPRA ...	
Required parameters			
INDATASET	IDS	IDS(MYTAPEDS)	2
INFILE	IFILE	IFILE(DD1)	2
Optional parameters			
CATALOG	CAT	CAT(MYCAT2)	
DEVICETYPE	DEVT	DEVT(3330)	1
FILE	—	FILE(DD3)	1
NOSAVRAC	—	NOSAVRAC	3
OBJECTS	OBJ	OBJ((MYDS VOL(VOL001) - DEVT(3330)))	
OUTFILE	OFILE	OFILE(DD2)	
SAVRAC	—	SAVRAC	3
VOLUMES	VOL	VOL(VOL001)	1

Notes:

1. This parameter is a subparameter of OBJECTS.
2. You must specify either INFILE or INDATASET.
3. When SAVRAC is not specified, NOSAVRAC is the default.

LISTCAT Parameters: Summary

The following table shows which keywords can be used with the LISTCAT command.

Parameter	Abbr	Example	Notes
LISTCAT	LISTC	LISTC ...	
Required parameters			
There are no required parameters for LISTCAT.			
Optional parameters			
ALIAS	—	—	
ALL	—	ALL	1
ALLOCATION	ALLOC	ALLOC	1
ALTERNATEINDEX	AIX	AIX	
CATALOG	CAT	CAT(USERCAT1)	
CLUSTER	CL	CL	
CREATION	CREAT	CREAT(90)	
DATA	—	DATA	
ENTRIES	ENT	ENT(DS1 DS2)	
EXPIRATION	EXPIR	EXPIR(90)	
HISTORY	HIST	HIST	1
GENERATIONDATAGROUP	GDG	GDG	
INDEX	IX	IX	
LEVEL	LVL	LVL(SYS1)	
NAME	—	NAME	1
NONVSAM	NVSAM	NVSAM	
NOTUSABLE	NUS	NUS	
OUTFILE	OFILE	OFILE(DDPRINT2)	
PAGESPACE	PGSPC	PGSPC	
PATH	—	PATH	
SPACE	SPC	SPC	
USERCATALOG	UCAT	UCAT	
VOLUME	VOL	VOL	1

Notes:

1. NAME is the default, unless ALL, VOLUME, ALLOCATION, or HISTORY is specified.

LISTCRA Parameters: Summary

The following table shows which keywords can be used with the LISTCRA command.

Parameter	Abbr	Example	Notes
LISTCRA	LISTR	LISTR ...	
Required parameters			
INFILE	IFILE	IFILE(DDVOL1)	
Optional parameters			
CATALOG	CAT	CAT(UCAT01)	3
COMPARE	CMPR	CMPR	1,3
DUMP	—	DUMP	2
MASTERPW	MRPW	MRPW(MCATMRPW)	4
NAME	—	NAME	2
NOCOMPARE	NCMPR	NCMPR	1
OUTFILE	OFILE	OFILE(OUTDS)	
SEQUENTIALDUMP	SDUMP	SDUMP	2

Notes:

1. Unless you specify COMPARE, the default is NOCOMPARE.
2. Unless you specify DUMP or SEQUENTIALDUMP, the default is NAME.
3. When you specify COMPARE, you must identify the catalog with the CATALOG parameter.
4. When the master catalog is password protected, you must supply its master password.

PRINT Parameters: Summary

The following table shows which keywords can be used with the PRINT command.

Parameter	Abbr	Example	Notes
PRINT	—	PRINT...	
Required parameter			
INDATASET	IDS	IDS(MYDATA)	1
INFILE	IFILE	IFILE(DDMDS)	1
Optional parameters			
CHARACTER	CHAR	CHAR	2
COUNT	—	COUNT(23)	4
DUMP	—	DUMP	2
FROMADDRESS	FADDR	FADDR(122)	3
FROMKEY	FKEY	FKEY(M)	3
FROMNUMBER	FNUM	FNUM(23)	3
HEX	—	HEX	2
OUTFILE	OFILE	OFILE(DDNDS)	
SKIP	—	SKIP(20)	3
TOADDRESS	TADDR	TADDR(442)	4
TOKEY	—	TOKEY(S)	4
TONUMBER	TNUM	TNUM(30)	4

Notes:

1. You identify the cluster to be printed with either INFILE or INDATASET.
2. When CHARACTER or HEX is not specified, DUMP is the default.
3. You can identify the record at which printing is to start with: FROMKEY, FROMADDRESS, FROMNUMBER, or SKIP.
4. You can identify the record at which printing is to end with: TOKEY, TOADDRESS, TONUMBER, or COUNT.

REPRO Parameters: Summary

The following table shows which keywords can be used with the REPRO command.

Parameter	Abbr	Example	Notes
REPRO	—	REPRO...	
Required parameters			
INDATASET	IDS	IDS(MYDATA)	1
INFILE	IFILE	IFILE(DDMDS)	1
OUTDATASET	ODS	ODS(NEWDATA)	2
OUTFILE	OFILE	OFILE(DDNDS)	2
Optional parameters			
COUNT	—	COUNT(25)	6
ENVIRONMENT	ENV	ENV(DUM)	3
DUMMY	DUM		4
FROMADDRESS	FADDR	FADDR(122)	5
FROMKEY	FKEY	FKEY(M)	5
FROMNUMBER	FNUM	FNUM(25)	5
NOREPLACE	NREP	NREP	7
NOREUSE	NRUS	NRUS	
REPLACE	REP	REP	7
REUSE	RUS	RUS	
SKIP	—	SKIP(20)	5
TOADDRESS	TADDR	TADDR(442)	6
TOKEY	—	TOKEY(S)	6
TONUMBER	TNUM	TNUM(30)	6

Notes:

1. You identify the source cluster with either INFILE or INDATASET.
2. You identify the target cluster with either OUTFILE or OUTDATASET.
3. ENVIRONMENT is a subparameter of INFILE.
2. DUMMY is a subparameter of ENVIRONMENT.
4. You can identify the record at which copying is to start with: FROMKEY, FROMADDRESS, FROMNUMBER, or SKIP.
6. You can identify the record at which copying is to end with: TOKEY, TOADDRESS, TONUMBER, or COUNT.
7. When REPLACE is not specified, NOREPLACE is the default.

RESETCAT Parameters: Summary

The following table shows which keywords can be used with the RESETCAT command.

Parameter	Abbr	Example	Notes
RESETCAT	RCAT	RCAT...	
Required parameters			
CATALOG	CAT	CAT(MYCAT3/PASSCAT RCATDD)	1
CRAFILES	—	CRAFILES((...)(...))	2
ALL	—	CRAFILES(DD1 ALL)	2, 5
NONE	—	CRAFILES(DD1 ALL) (DD2 NONE))	2, 6
CRAVOLUMES	CRAVOL	CRAVOL((111111 3330) (ABC 2314))	2
Optional parameters			
IGNORE	IGN	IGN	
NOIGNORE	NIGN	NIGN	
MASTERPW	MRPW	MRPW(BIGDAD)	
WORKCAT	WCAT	WCAT(MYCAT4/CAT4PASS)	
WORKFILE	WFILE	WFILE(DD3/WFPASS)	

Notes:

1. RCATDD identifies a DD statement that describes the catalog to be reset.
2. You must specify either CRAFILES or CRAVOLUMES.
3. Unless you specify IGNORE, the default option is NOIGNORE.
4. When the master catalog is password protected, you must supply its master password.
5. DD1 identifies a DD statement that describes a volume to be used to reset the catalog. The volume contains CRA catalog records, and the corresponding records in the catalog will be reset.
6. If DD1 identifies a volume which contains a multivolume data set, other volumes of that multivolume data set may be needed during the reset operation. DD2 identifies a DD statement which describes such a volume.
7. The WORKCAT catalog must not be the same catalog as the one being reset.
8. If WORKFILE is not specified, IDCUT1 is used as a default name.

VERIFY Parameters: Summary

The following table shows which keywords can be used with the VERIFY command.

Parameter	Abbr	Example	Notes
VERIFY	VFY	VFY	
Required parameters			
FILE	—	FILE(ACCNTS)	1
DATASET	—	DATASET(MYDATA)	1

Notes:

1. You can identify the object to be verified with either FILE or DATASET.

APPENDIX G: INVOKING ACCESS METHOD SERVICES FROM A PROBLEM PROGRAM

Access Method Services can be invoked by a problem program through the use of the ATTACH, LINK or LOAD and CALL macro instructions.

The dynamic invocation of Access Method Services enables respecification of selected processor defaults as well as the ability to manage input/output operations for selected data sets.

Authorized Program Facility (APF)

There are five Access Method Services commands which require APF authorization; they are:

- REPRO—when using the copy-catalog or catalog unload/reload facility.
- PRINT—when the object to be printed is a VSAM catalog.
- LISTCRA
- EXPORTRA
- RESETCAT

If the problem program invoking Access Method Services needs to perform any of these functions, it must be authorized and be located in an authorized library. Additionally, a User I/O Routine must reside in an authorized library if the problem program it is associated with is authorized. See "Authorized Program Facility (APF)" in *OS/VS2 System Programming Library: Supervisor,* for information about program authorization.

Invoking Macro Instructions

The following descriptions of the invoking macro instructions are related to Figure 30, which describes the argument lists referenced by the invoking macros.

LINK or ATTACH Macro Instruction

Access Method Services may be invoked via either the LINK or the ATTACH macro instruction.

The format of the LINK or ATTACH macro instruction is:

[*name*]	{**LINK** | **ATTACH**} **EP=IDCAMS,** **PARAM=(***optionaddr* [,*ddnameaddr*] [,*pgnoaddr*] [,*iolistaddr*]), **VL=1**

EP=IDCAMS
specifies that the program to be invoked is IDCAMS.

PARAM=

specifies the addresses of the parameters to be passed to IDCAMS. These values can be coded:

optionaddr
specifies the address of an option list, which can be specified in the PARM parameter of the EXEC statement and is a valid set of parameters for the Access Method Services PARM command. If you do not wish to specify any options, this address must point to a halfword of binary zeros. Figure 30 shows the format of the options list.

ddnameaddr
specifies the address of a list of alternate ddnames for standard data sets used during IDCAMS processing. If standard ddnames are used and this is not the last parameter in the list, it should point to a halfword of binary zeros. If it is the last parameter, it may be omitted. Figure 30 shows the format of the alternate ddname list.

pgnoaddr
specifies the address of a 6-byte area which contains an EBCDIC starting page number for the system output file. If the page number is not specified, but this is not the last parameter in the list, the parameter must point to a halfword of binary zeros. If it is the last parameter, it may be omitted. If omitted the default page number is 1. Figure 30 shows the format of the page number area.

iolistaddr
specifies the address of a list of externally controlled data sets and the addresses of corresponding I/O routines. If no external I/O routines are supplied, this parameter may be omitted. Figure 30 shows the format of the I/O list.

VL=1
causes the high-order bit of the last address parameter of the PARAM list to be set to 1.

LOAD and CALL Macro Instructions

Access Method Services may also be invoked via a LOAD of the module IDCAMS, followed by a CALL to that module. The format of the LOAD macro instruction is:

[*name*]	**LOAD**	{**EP=IDCAMS** | **EPLOC=** *address of name*}

where:

EP=IDCAMS
is the entry point name of the IDCAMS program to be loaded into virtual storage.

EPLOC= *address of name*
is the address of an 8-byte character string 'IDCAMSbb'.

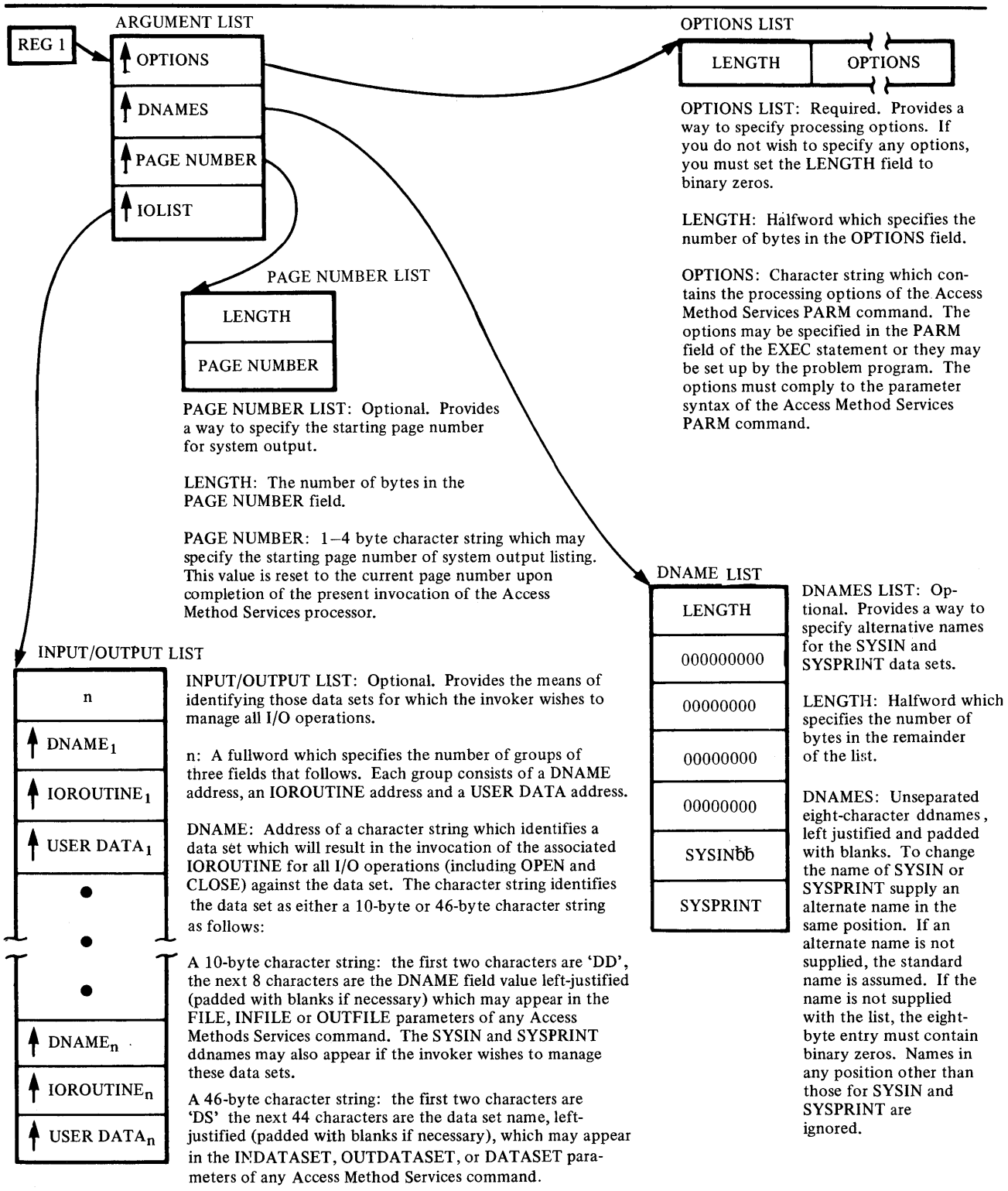


Figure 30. Processor Invocation Argument List From a Problem Program

After loading IDCAMS, CALL may be used to pass control to it. The format of the CALL macro instruction is:

[*name*]	**CALL**	**IDCAMS,** (*optionaddr* [,*ddnameaddr*] [,*pgnoaddr*] [,*iolistaddr*]), **VL**

where:

IDCAMS
is the name of the entry point to be given control.

optionaddr
specifies the address of an options list which is usually specified in the PARM parameter of the EXEC statement and is a valid set of parameters for the Access Method Services PARM command. If you do not wish to specify any options, this address must point to a halfword of binary zeros. Figure 30 shows the format of the options list.

ddnameaddr
specifies the address of a list of alternate ddnames for standard data sets used during IDCAMS processing. If standard ddnames are used and this is not the last parameter in the list, it should point to a halfword of binary zeros. If it is the last parameter, it may be omitted. Figure 30 shows the format of the alternate ddname list.

pgnoaddr
specifies the address of a 6-byte area which contains an EBCDIC starting page number for the system output file. If the page number is not specified, but this is not the last parameter in the list, the parameter must point to a halfword of binary zeros. If it is the last parameter, it may be omitted. If omitted the default page number is 1. Figure 30 shows the format of the page number area.

iolistaddr
specifies the address of a list of externally controlled data sets and the addresses of corresponding I/O routines. If no external I/O routines are supplied, this parameter may be omitted. Figure 30 shows the format of the I/O list.

VL
causes the high-order bit of the last address parameter in the macro expansion to be set to 1.

Invocation From a PL/I Program

Access Method Services may also be invoked from a PL/I program using the facilities of the IBM PL/I Optimizing Compiler Program Product. IDCAMS must be declared to the compiler as an external entry point with the ASSEMBLER and INTER options. The Access Method Services processor is loaded by issuing a FETCH IDCAMS statement, is branched to via a CALL statement, and deleted via a RELEASE IDCAMS statement. The format of the CALL statement is:

CALL	**IDCAMS**	(*options*[,*ddnames*][,*pageno*][,*iolist*]);

where:

options
specifies a valid set of parameters for the Access Method Services PARM command. If no parameters are to be specified, options should be a halfword of binary zeros. Figure 30 shows the format of the options area.

ddnames
specifies a list of alternate ddnames for standard data sets used during IDCAMS processing. If standard ddnames are used and this is not the last parameter in the list, ddnames should be a halfword of binary zeros. If it is the last parameter, it may be omitted. Figure 30 shows the format of the alternate DDnames list.

pageno
specifies a 6-byte field that contains an EBCDIC starting page number for the system output file. If the page number is not specified, but this is not the last parameter in the list, the parameter must be a halfword of binary zeros. If it is the last parameter, it may be omitted. If not specified, the default page number is 1. Figure 30 shows the format of the page number area.

iolist
specifies a list of externally controlled data sets and the addresses of corresponding I/O routines. If no external I/O routines are supplied, this parameter may be omitted. Figure 30 shows the format of the I/O list.

Processor Invocation

Figure 30 shows the processor invocation argument list as it exists in the user's area.

Entry and exit to the Access Method Services processor occurs through IDCSA01, a module of the System Adapter. Standard linkage is used; that is, register 1 points to the argument list, register 13 points to a save area, register 14 contains the return address, and register 15 contains the entry point address for IDCSA01. On exit from the Access Method Services processor, register 15 contains the value of MAXCC (see the section "Processor Condition Codes" below).

The argument list, as shown in Figure 30, can be a maximum of four fullword addresses pointing to strings of data. The last address in the list contains a "1" in the sign field. The first three possible strings of data begin with a two-byte length field. A null element in the list can be indicated by either an address of zeros or a length of zero.

Processor Condition Codes

The processor's condition code is LASTCC, which can be interrogated in the command stream. The possible values, their meanings, and examples of causes are in the following table. The table illustrates the value of LASTCC.

Code	Meaning
0	The function was executed as direct and expected. Informational messages may have been issued.
4	Some annoyance in executing the complete function was met, but it was possible to continue. The results might not be exactly what the user wants, but no permanent harm appears to have been done by continuing. A warning message was issued.
8	A function could not perform all that was asked of it. The function was completed, but specific details were bypassed.
12	The entire function could not be performed.
16	Severe error or problem encountered. Remainder of command stream is flushed and processor returns condition code 16 to the operating system.

The LASTCC condition code is reflected in its related message numbers. The first numeric character of the message number equals the condition code divided by 4. MAXCC, which can also be interrogated in the command stream, is the biggest value of LASTCC thus far encountered.

User I/O Routines

User I/O routines enable a user to perform all I/O operations for a data set which would normally be handled by the Access Methods Services processor. This makes it possible, for instance, to control the command input stream by providing an I/O routine for SYSIN.

A user I/O routine is invoked by Access Method Services for all operations against the selected data sets. The identification of the data sets and their associated I/O routines is via the input/output list of the processor invocation parameter list (see Figure 30).

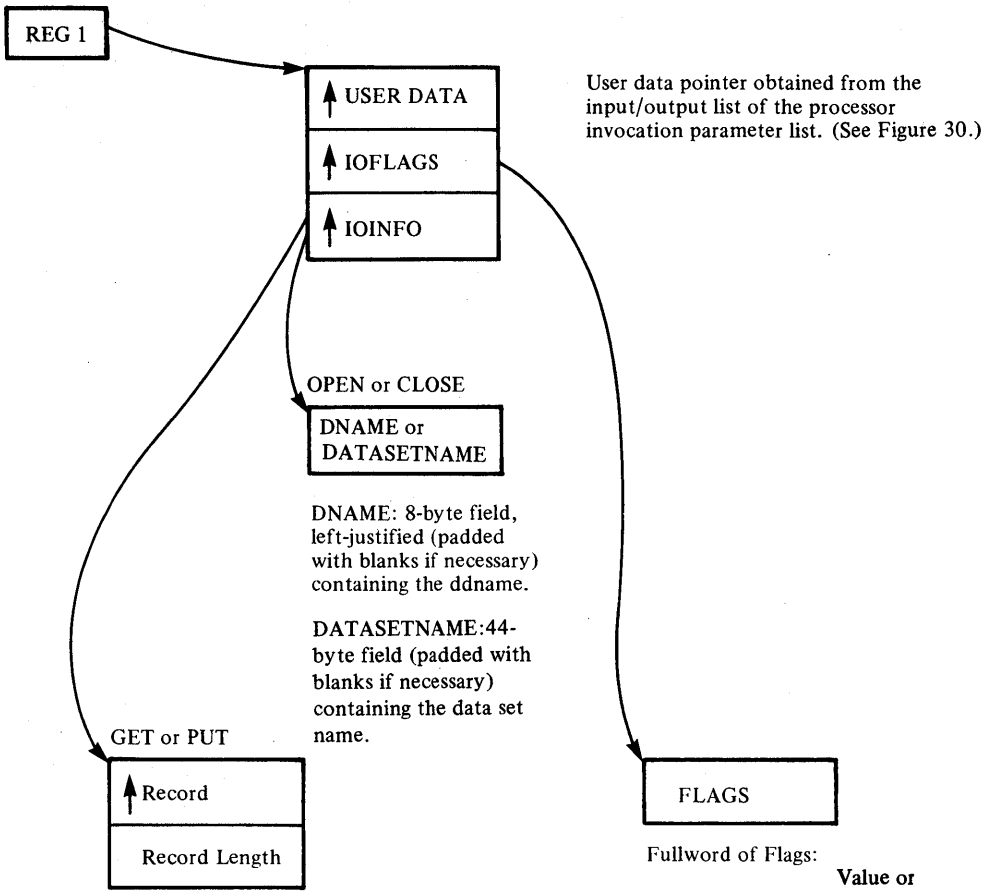
When writing a user I/O routine, the user must be aware of three things: First, the processor handles the user data set as if it were a nonVSAM data set that contains variable-length unblocked records (maximum record length is 32760 bytes) with a physical sequential organization. The processor does not test for the existence of the data set. Second, the user must know the data format so that the user's routine can be coded to handle the correct type of input and format the correct type of output. Third, each user routine must handle errors encountered for data sets it is managing and provide a return code to the processor in register 15. The processor uses the return code to determine what it is to do next.

The permissible return codes are:

- 0—operation successful
- 4—end of data for a GET operation
- 8—error encountered during a GET/PUT operation, but continue processing
- 12—do not allow any further calls (except CLOSE) to this routine

Figure 31 shows the argument list used in communication between the user I/O routine and the Access Method Services processor. The User I/O routine is invoked by the processor for OPEN, CLOSE, GET and PUT routines.

The type of operation to be performed is indicated via the IOFLAGS. The IOINFO field indicates, for OPEN and CLOSE operations, the data set name or ddname of the data set; for GET and PUT operations, the IOINFO field is used to communicate the record length and address.



For a GET this information is returned to the processor by the user's I/O routine in the 8-byte area passed to the routine.

For a PUT the processor gives this information to the user's I/O routine.

Record (GET): Address of retrieved record.

Record Length (GET): Fullword length of retrieved record.

Record (PUT): Address of record to be written.

Record Length (PUT): Fullword length of record to be written.

Fullword of Flags:

	Value or Bit Pattern	Meaning
Byte 1 (Operation)	0	OPEN
	4	CLOSE
	8	GET
	12	PUT
Byte 2	1 • • • • • • •	OPEN for input
	• 1 • • • • • •	OPEN for output
	• • 1 • • • • •	Indicates IOINFO contains the address of a ddname on OPEN.
	• • • 1 • • • •	Indicates IOINFO contains the address of a data set name on OPEN.
Bytes 3, 4 (Record type for PUT only)	0	Normal data record is to be written.
	n	Message serial number converted to binary if IDC message is to be written.

Figure 31. Arguments Passed to and From a User I/O Routine

APPENDIX H: MAKING THE MASTER CATALOG RECOVERABLE

You have the option of making your master catalog recoverable. If you wish to do so, the steps are as follows:

1. Enhanced VSAM must already be installed on your system.
2. Mount a scratch pack to hold a temporary catalog, storage index, and the page data sets.
3. Define a temporary master catalog on the scratch pack.
4. Catalog the nonVSAM data sets needed for an IPL on the temporary master catalog. Define a new storage index and page data sets on the scratch volume in the temporary catalog. The page data sets should have the same name as the original page data sets so that the IEASYS00 member in SYS1.PARMLIB need not be replaced.
5. Export (with the PERMANENT attribute) all VSAM data sets from the old master catalog. Ignore for the moment the storage index and page data sets in the old master catalog.
6. Alter the SYSCATLG member of the SYS1.NUCLEUS data set so that it points to the temporary catalog.
7. IPL the system (cold start procedure). The system now is using the temporary catalog as the master catalog, and refers to the storage index data set and page data sets cataloged in the temporary catalog.
8. Use the ALTER REMOVEVOLUMES command to remove VSAM from the volumes containing the old master catalog, the old page data sets, and the old storage index, and to remove VSAM from any volumes owned by the old master catalog.
9. Define a recoverable master catalog in place of the original.
10. Redefine the storage index and page data sets in their original locations on the new master catalog. (Allow room for the catalog recovery area.) Catalog all nonVSAM data sets needed for IPL in the new master catalog.
11. Replace the SYSCATLG member in SYS1.NUCLEUS (now cataloged in the new master catalog) so that the SYSCATLG member points to the new master catalog.
12. ReIPL the system (a cold start). The system should now be using the new recoverable master catalog, the redefined storage index, and the newly defined page data sets.
13. Clean up the scratch pack using the ALTER command with the REMOVEVOLUMES parameter.
14. Recatalog all the remaining nonVSAM data sets into the new master catalog.
15. Connect (and define aliases for) OS catalogs (CVOLs) that were connected to the old master catalog.
16. Redefine the data spaces that were deleted from the old master catalog.

17. Import the previously exported VSAM data sets into the new master catalog.
18. Import (with the CONNECT parameter) and define aliases for user catalogs that were connected to the old master catalog.

Your new master catalog is now identical to your old master catalog, and your new master catalog is recoverable (that is, its volume includes a catalog recovery area and can be processed with the LISTCRA, EXPORTRA, and IMPORTRA commands).

Figure 32 shows an example of the job stream needed to make your master catalog a recoverable catalog. The step numbers in the jobs correspond to the steps described previously.

```
//JOB1 JOB ...
//*  STEP 1 - INSTALL VSAM ON YOUR SYSTEM
//*  STEP 2 - MOUNT A SCRATCH PACK
//STEP3         EXEC PGM=IDCAMS
//SYSPRINT  DD  SYSOUT=A
//CATPAK  DD UNIT=3330,VOL=SER=CATPAK,DISP=OLD
//SYSIN  DD  *
  DEFINE  MASTERCATALOG -
    (NAME(TEMPCAT)  -
    FILE(CATPAK)  -
    VOLUME(CATPAK)  -
    CYLINDERS (12 1))
/*
//STEP4 EXEC PGM=IDCAMS,COND=(0,LT)
//STEPCAT DD DSN=TEMPCAT,DISP=OLD
//SYSPRINT  DD  SYSOUT=A
//CATPAK  DD UNIT=3330,VOL=SER=CATPAK,DISP=OLD
//SYSIN DD *
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.LPALIB ) VOL(SG2001)  DEV(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.LINKLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.DSSVM ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.SVCLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.NUCLEUS ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.DCMLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.INDMAC ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.CMDLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.CMDLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.HELP ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.SAMPLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.MACLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.PROCLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.TELCMLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.UADS ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.VTAMLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.IMAGELIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.PARMLIB ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.BRODCAST ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.MANX ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.MANY ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.DUMP00 ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.DUMP01 ) VOL(SG2001)  DEVT(3330))
  DEF CAT(TEMPCAT)  NVSAM  (NAME(SYS1.LOGREC ) VOL(SG2001)  DEVT(3330))
  DEF CLUSTER  -
    (NAME(SYS1.STGINDEX)    -
    FILE (CATPAK)   -
    KEYS(12,8)  -
    CYLINDERS (6,0)  -
    RECORDSIZE(2041,2041)  -
    VOLUME(CATPAK)  -
    UNIQUE)  -
    DATA(CONTROLINTERVALSIZE(2048))  -
    CATALOG(TEMPCAT)
```

Figure 32 (Part 1 of 4). Sample Job Stream to Make the Master Catalog Recoverable

```
  DEFINE PAGESPACE  -
    (NAME(SPOOL1P1)  -
    FILE(CATPAK)  -
    CYLINDERS(80)  -
    VOLUME(CATPAK)  -
    UNIQUE)  -
    CATALOG(TEMPCAT)
  DEFINE PAGESPACE  -
    (NAME(SPOOL1P2)  -
    FILE(CATPAK)  -
    CYLINDERS(40)  -
    VOLUME(CATPAK)  -
    UNIQUE)  -
    CATALOG(TEMPCAT)
/*
//STEP5  EXEC   PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//*       DD  statements that describe theoutput data set
//SYSIN    DD   *
       EXPORT commands required to export the VSAM data sets and user
    catalog connector entries from the old master catalog
/*
//STEP6A EXEC PGM=IEHPROGM,COND=(0,LT)
//SG2001   DD  DISP=OLD,VOL=SER=SG2001,UNIT=3330
//SYSPRINT   DD  SYSOUT=A
//SYSIN    DD   *
  SCRATCH DSNAME=SYS1.NUCLEUS,VOL=3330=SG2001,MEMBER=SYSCATLG
/*
//STEP6B EXEC PGM=IEBGENER,COND=(0,LT)
//SYSIN    DD DUMMY
//SYSPRINT   DD  SYSOUT=A
//SYSUT2 DD  DSN=SYS1.NUCLEUS(SYSCATLG),DISP=(OLD,KEEP),
//  DCB=BLKSIZE=13030,VOL=SER=SG2001,UNIT=3330
//SYSUT1 DD *
CATPAK   (See Note 1)
/*
//*  STEP 7 - IPL THE SYSTEM
//
//JOB2    JOB ...
//STEP8 EXEC PGM=IDCAMS
//SYSPRINT  DD  SYSOUT=A
//DD3330 DD UNIT=3330, DISP=OLD,VOL=SER=(SG2001,SPOOL1)
//SYSIN DD *
  ALTER TEMPCAT RVOL  ( SG2001 SPOOL1)  FILE  ( DD3330 )
/*
//STEP9 EXEC PGM=IDCAMS,COND=(0,LT)
//SYSPRINT  DD  SYSOUT=A
//SG2001 DD UNIT=3330,VOL=SER=SG2001,DISP=OLD
//SYSIN  DD *
```

Figure 32 (Part 2 of 4). Sample Job Stream to Make the Master Catalog Recoverable

```
  DEFINE MASTERCATALOG -
    (NAME(AMASTCAT)  -
    FILE(SG2001)  -
    VOLUME(SG2001)  -
    CYLINDERS(12 1)  -
    RECOVERABLE)
//STEP10 EXEC PGM=IDCAMS,COND=(0,LT)
//STEPCAT DD DSN=AMASTCAT,DISP=OLD
//SYSPRINT  DD  SYSOUT=A
//SPOOL1 DD UNIT=3330,VOL=SER=SPOOL1,DISP=OLD
//SG2001 DD UNIT=3330,VOL=SER=SG2001,DISP=OLD
//SYSIN DD *
  DEF CLUSTER  -
    (NAME(SYS1.STGINDEX)  -
    FILE (SG2001)  -
    KEYS(12,8)  -
    CYLINDERS(6,0)  -
    RECORDSIZE(2041,2041)  -
    VOLUME(SG2001)  -
    UNIQUE)  -
    DATA(CONTROLINTERVALSIZE (2048))  -
    CATALOG(AMASTCAT)
  DEFINE PAGESPACE  -
    (NAME(SPOOL1P1)  -
    FILE(SPOOL1)  -
    CYLINDERS(140)  -
    VOLUME(SPOOL1)  -
    UNIQUE)  -
    CATALOG(AMASTCAT)
  DEFINE PAGESPACE  -
    (NAME(SPOOL1P2)  -
    FILE(SPOOL1)  -
    CYLINDERS(40)  -
    VOLUME(SPOOL1)  -
    UNIQUE)  -
    CATALOG(AMASTCAT)
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.LPALIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.LINKLIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.DSSVM ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.SVCLIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.NUCLEUS ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.DCMLIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.INDMAC ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.CMDLIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.HELP ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.SAMPLIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.MACLIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.PROCLIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.TELCMLIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.UADS ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.VTAMLIB ) VOL(SG2001) DEVT(3330))
```

Figure 32 (Part 3 of 4). Sample Job Stream to Make the Master Catalog Recoverable

```
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.IMAGELIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.PARMLIB ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.BRODCAST ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.MANX ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.MANY ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.DUMP00 ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.DUMP01 ) VOL(SG2001) DEVT(3330))
  DEF CAT(AMASTCAT) NVSAM (NAME(SYS1.LOGREC ) VOL(SG2001) DEVT(3330))
/(L*
//STEP11A EXEC PGM=IEHPROGM,COND=(0,LT)
//SG2001  DD  DISP=OLD,VOL=SER=SG2001,UNIT=3330
//SYSPRINT  DD  SYSOUT=A
//SYSIN  DD  *
  SCRATCH  DSNAME=SYS1.NUCLEUS,VOL=3330=SG2001,MEMBER=SYSCATLG
/(L*
//STEP11B EXEC PGM=IEBGENER,COND=(0,LT)
//SYSIN  DD  DUMMY
//SYSPRINT  DD  SYSOUT=A
//SYSUT2  DD  DSN=SYS1.NUCLEUS(SYSCATLG),DISP=(OLD,KEEP),
// DCB=BLKSIZE=13030,VOL=SER=SG2001,UNIT=3330
//SYSUT1  DD  *
SG2001  (SEE NOTE 1)
/(L*
//
//JOB2  JOB ...
//(L* STEP 12 - REIPL THE SYSTEM
//STEP13  EXEC PGM=IDCAMS
//SYSPRINT  DD  SYSOUT=A
//CATPAK  DD  UNIT=3330,VOL=SER=CATPAK,DISP=OLD
//SYSIN DD *
  ALTER AMASTCAT -
      FILE(CATPAK) -
      REMOVEVOLUMES(CATPAK)
/(L*
//
```

Note 1: This data record has the volume serial number of the volume containing the alternate catalog in columns 1 to 6.

Figure 32 (Part 4 of 4). Sample Job Stream to Make the Master Catalog Recoverable

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.

alias entry: An entry that relates an alias (alternate entryname) to the real entryname of a user catalog or nonVSAM data set.

alternate index: An ordered collection of records, each consisting of a key (called the alternate key) and one or more pointers. An alternate index is used by VSAM to sequence and locate the records of a key-sequenced or entry-sequenced VSAM data set. An alternate index is organized as a key-sequenced data set. (*See also* alternate key, base cluster, and path.)

alternate-index entry: A catalog entry that contains information about an alternate index. An alternate index is conceptually a key-sequenced cluster, and is cataloged in the same way. An alternate-index entry points to a data entry and an index entry to describe the alternate index's components, and to a cluster entry to identify the alternate index's base cluster. (*See also* cluster entry.)

alternate-index record: A collection of items used to sequence and locate one or more data records in a base cluster. Each alternate-index record contains an alternate-key value and one or more pointers. When the alternate index supports a key-sequenced data set, each data record's prime key value is the pointer. When the alternate index supports an entry-sequenced data set, the data record's RBA value is the pointer. (*See also* alternate index, alternate key, base cluster, and key.)

alternate key: One or more characters within a data record, used to identify the data record or control its use. Unlike the prime key, the alternate key can identify more than one data record. (*See also* key and key field.)

application: As used in this publication, the use to which an access method is put or the end result that it serves; contrasted to the internal operation of the access method.

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

base cluster: The VSAM cluster whose data records are to be accessed through a path. Usually, a base cluster is the key-sequenced or entry-sequenced data set which an alternate index supports (that is, an alternate index is used by VSAM to sequence and locate the data records of a base cluster). (*See also* alternate index and path.)

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 cleanup: A process that allows you to delete entries if their volume is no longer available; catalog cleanup also allows you to delete a catalog even though it isn't empty. Catalog cleanup is a function of the DELETE command.

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).

catalog recovery area: (*See* CRA.)

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.

component: The data portion or, for a key-sequenced cluster, alternate index, or VSAM catalog, the index portion or a VSAM object. In this book, the components of an object are usually referred to as the object's data component and index component.

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 access: The retrieval and storage of a VSAM data set's contents, based on the RBA of a control interval

(that is, the user's program processes a control interval, rather than a data record, as a logical entity).

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.

CRA: Catalog recovery area. An entry-sequenced data set that exists on each volume owned by a recoverable catalog, including the volume on which the catalog resides. The CRA contains copies of the catalog's records, and can be used to recover a damaged catalog.

data component: That part of a VSAM data set, alternate index, or catalog that contains the object's data records.

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 name: A unique name for each component or object as it is identified in a catalog. The entryname is the same as the *dsname* in a DD statement that describes the object.

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.

exception: An abnormal condition such as an I/O error encountered in processing a data set.

exception exit: A user-written exit routine that attempts to recover from exceptional processing situations and is processed before the SYNAD exit routine gets control. The exception exit routine is similar to the SYNAD error exit routine, except that the exception exit routine can be tailored by the user for the data or index component of a VSAM data set or alternate index.

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.

external sort: Sorting of data records into a new sequence using a small amount of virtual storage and two temporary sortfiles (entry-sequenced data sets) on a direct-access storage volume. (*See also* internal sort.)

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 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 component: That part of a key-sequenced data set, catalog, or alternate index, that establishes the sequence of the data records within the object it indexes. The index is used to locate each record in the object's data component, based on the record's key value.

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.

index upgrade: The process of updating an alternate index to reflect the changes made to the contents of its base cluster.

integrity: (*See* data integrity.)

internal sort: Sorting of data records into a new sequence using virtual storage and no temporary data sets on a direct-access storage device. (*See also* external sort.)

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.

job catalog: A catalog made available for a job, by means of a JOBCAT DD statement, or for a job step, by means of a STEPCAT DD statement.

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.

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.

mass storage volume: The unit of mass storage in the 3850 Mass Storage System.

nonunique keys: An alternate key value in a base cluster's data record that equals the alternate key value of another data record. When an alternate index is created such that it allows nonunique keys, more than one data record might be located with the same key value.

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).

object: A logical entity created by VSAM, such as a cluster (VSAM data set) and its components, an alternate index and its components, a VSAM catalog and its components, a path, or a VSAM data space.

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 or a computer operator at the console must supply to meet security requirements before the program gains access to a data set.

path: A data set name for the combination of an alternate index and its base cluster, or an alias for a VSAM data set.

path entry: A catalog entry that contains information about a path, and that points to the path's related objects.

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.

primary space allocation: Initially allocated space on a direct-access storage device, occupied by or reserved for a particular data set. (*See also* secondary space allocation.)

prime index: The index component of a key-sequenced data set. (*See also* index and alternate index.)

prime key: (*See* key.)

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.)

record replacement: An enhancement of the REPRO command which allows the user to merge two data sets and, when duplicate records are found (that is, the key values of two records are equal) the data record from the source data set replaces the data record in the target data set.

recoverable catalog: A catalog defined with the recoverable attribute. Duplicate catalog entries are put into CRAs that can be used to recover data in the event of catalog failure. (*See also* CRA.)

recovery volume: The first volume of a prime index, if the VSAM data set is key-sequenced cluster; otherwise, the first volume of the data set, if ESDS.

relative byte address: (*See* RBA.)

relative record: A data record whose position depends on its placement within a a group of data records; its position, or record number, is its displacement, in records, from the beginning of the data set.

relative-record data set: A data set whose records are loaded into fixed-length slots.

relative-record number: A number that identifies not only the slot, or record space, in a relative-record data set but also the record occupying the slot.

replication: (*See* index replication.)

reusable data set: A VSAM data set that can be used as a workfile regardless of its old contents.

secondary space allocation: A contiguous space on a direct-access device, occupied by or reserved for a particular data set, which is allocated after space in the primary extent has been exhausted. (*See also* primary space allocation.)

security: (*See* data security.)

sequence checking: The process of verifying the order of a set of records relative to some field's collating sequence.

sequence set: The lowest level of the index of a key-sequenced data set; it gives the locations of the control intervals in the data set and orders them by the key sequence of the data records they contain. The sequence set and the index set together comprise the index.

sequential access: The retrieval or storage of a data record in either its entry sequence 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.

slot: The space for a data record in a relative-record data set.

sort: (*See* external sort, internal sort, and collating sequence.)

spanned record: A logical record whose length exceeds control interval length, and crosses (or spans) one or more control interval boundaries within a control area.

stage: (verb) To transmit data from a mass storage volume to a direct-access storage staging drive.

step catalog: (*See* job catalog.)

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

upgrade set: All the alternate indexes that VSAM has been instructed to update whenever there is a change to the data component of the base cluster.

user catalog: 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 cleanup: The process of deleting all VSAM data spaces from a volume and removing a VSAM catalog's ownership of the volume, even though the volume contains VSAM data records.

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 and the catalog is no longer available.

INDEX

For additional information about any subject listed in this index, refer to the publications that are listed under the same subject *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. GC26-3841-1

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)

OS/VS2 Access Method Services
GC26-3841-1

Reader's Comment Form

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

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

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

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

Fold and Staple

First Class Permit
Number 2078
San Jose, California

Business Reply Mail

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

Postage will be paid by:

IBM Corporation
System Development Division
Programming Publishing—Department J57
1501 California Avenue
Palo Alto, California 94304

Fold and Staple

OS/VS2 Access Method Services (File No. S370-30) Printed in U.S.A. GC26-3841-1

IBM®

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

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

OS/VS2 MVS Supervisor Performance #2 Newsletter

VS2.03.807

This Newsletter No. GN26-0847
Date May 28, 1976

Base Publication No. GC26-3841-1
File No. S370-30

Previous Newsletters None

OS/VS2 ACCESS METHOD SERVICES

© IBM Corp. 1975, 1976

This OS/VS2 MVS Supervisor Performance #2 Newsletter provides replacement pages for the subject publication. These replacement pages remain in effect unless specifically altered.

Do *not* replace the indicated pages in the publication unless you have installed OS/VS2 MVS Supervisor Performance #2.

Pages supplied with this newsletter replace pages in the publication if no previous newsletter in the series VS2.03.8*nn* has replaced pages affected by this newsletter. If a newsletter page from the series VS2.03.8*nn* (see upper corner of the page) has replaced a page in the publication, do not remove the page but *add* the corresponding page from this newsletter. This will result in two pages with the same page number.

Only replace table of contents, summary of amendments, lists of illustrations, and index pages in the publication with pages from a newsletter with a *higher date.* Newsletters with the same date contain equivalent information.

Pages to be replaced or inserted are:

3 - 12 (3.1 added)
19 - 20.1 (20.1 added)
53, 53.1 (53.1 added)
54
97, 98
101 - 103 (101.1 added)
115, 116
133 - 134 (133.1 added)
233 - 236.1 (236.1 added)
241, 242
273, 274
297 - 300 (299.1 added)
307 - 308.1 (308.1 added)
385 - 388
393 - 394.1 (394.1 added)
397 - 398 (397.1 added)
407 - 412
447, 448
453, 454
475 - 490

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

Summary of Amendments

- Support of Auxiliary Storage Manager (ASM) Redesign.
- Support of Resource Access Control Facility (RACF).

For a complete list of publications that support OS/VS2 MVS Supervisor Performance #2, see *OS/VS2 MVS Supervisor Performance #2 System Information,* GC28-0727.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

IBM Corporation, Programming Publishing, P. O. Box 50020, San Jose, California 95150

Printed in U.S.A.

IBM OS/VS2 MVS Data Management Newsletter VS2.03.808

This Newsletter No. GN26-0871
Date July 30, 1976

Base Publication No. GC26-3841-1
File No. S370-30

Previous Newsletters GN26-0847

OS/VS2 Access Method Services

© IBM Corp. 1975, 1976

This OS/VS2 MVS Data Management Newsletter provides replacement pages for the subject publication. These replacement pages remain in effect unless specifically altered.

Do *not* replace the indicated pages in the publication unless you have installed the OS/VS2 MVS Data Management Selectable Unit.

Pages supplied with this newsletter replace pages in the publication if no previous newsletter in the series VS2.03.8*nn* has replaced pages affected by this newsletter. If a newsletter page from the series VS2.03.8*nn* (see upper corner of the page) has replaced a page in the publication, do not remove the page but *add* the corresponding page from this newsletter. This will result in two pages with the same page number.

Only replace table of contents, summary of amendments, lists of illustrations, and index pages in the publication with pages from a newsletter with a *higher date*. Newsletters with the same date contain equivalent information.

Pages to be replaced or inserted are:

7 – 28 (22.1 added)
35,36
41 – 44
59 – 64 (60.1 and 63.1 added)
105 – 110
113 – 116.5 (114.1 and 116.1 – 116.5 added)
273, 274
344.1-344.6 (added)
435
455 – 458 (456.1 and 457.1 added)
475 – 490

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

Summary of Amendments

This newsletter contains a description of RESETCAT, an Access Method Services command that is used to reset a catalog to the level of its owned volumes without moving data, and the changes required for extended CVOL support, which provides a CVOL function equivalent to OS/VS2 SVS Release 1 while retaining the VSAM master catalog as the only system master catalog.

For a complete list of publications that support OS/VS2 MVS Data Management, see *OS/VS2 Data Management System Information,* GC26-3861.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150

Printed in U.S.A.

OS/VS2 Access Method Services
Data Management
GC26-3841-1

Reader's Comment Form

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

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

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

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

Fold and Staple

First Class Permit
Number 6090
San Jose, California

Business Reply Mail

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

Postage will be paid by:

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and Staple

IBM®

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

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

OS/VS2 Access Method Services: Data Management (File No. S370-30) Printed in U.S.A. GC26-3841-1