SC24-5144-1
File No. S370-30

Program Product

Using VSE/VSAM Commands and Macros

Program Number 5746-AM2

Release 2

IBM

Second Edition (December 1979)

This edition, SC24-5144-1, applies to Release 2 of IBM Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM), Program Product 5746-AM2, and to subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Changes are continually made to the information contained herein; before using this publication in connection with the operation of IBM systems, consult the *IBM System/370 and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

Summary of Amendments

For a list of changes, see page iii.

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

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

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

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

Summary of Amendments for Using VSE/VSAM Commands and Macros

Summary of Amendments for SC24-5144-1 Release 2

The following is a summary of major changes to VSE/VSAM for Release 2. The summary is in two parts, first each individual item is noted and then the changes to each command are listed:

- SHAREOPTIONS parameter - Now allows you to share VSAM files across VSE systems.
- The CANCEL command - Allows you to cancel the current job or job step.
- The DEDICATE parameter - Indicates that the entire space of a volume or the remaining free space on a volume is to be owned by VSAM.
- CLASS and USECLASS parameters - Give you the option of specifying an additional five classes of data space.
- The new NOALLOCATION and current REUSE parameters - Allow you to define a file into a VSAM catalog without allocating any space to the file.
- Both the new DISP operand of the DLBL job control statement and the current MACRF operand of the ACB macro - Allow you to specify status options for a file at OPEN time.
- Both the new DISP operand of the DLBL job control statement and the new CLOSDSP operand of the ACB macro - Allow you to specify the disposition of the file at CLOSE time.
- The NOALLOCATION parameter - Allows you to replace the usual system parameter defaults with your own list of defaults (via default models).
- The DEFAULTVOLUMES parameter - Instructs VSAM to select the volume(s) it needs from a previously established default list of volumes.
- The DEFINE CLUSTER NAME parameter - Allows you to predefine work files for each partition in which you anticipate using them.
- The DEFINE CLUSTER NAME parameter - Allows you to indicate that work file space is to be shared between central processors. These processors can run the same job in any partition of any processor without conflict.
- VSE/VSAM's job control - A greatly simplified way of specifying job control is now available. *(Note that the old way of specifying job control still applies, however, for reasons of clarity and brevity, the new way only is presented in this book.)* You can omit:
  a. ASSGN and EXTENT job control statements in most cases.
  b. Many of the DLBL statements and their associated *filename (dname)* parameters.
  c. Many of the FILE *(dname)* and CATALOG *(dname)* parameters.
  d. Additionally, some of the existing command parameters have been replaced with parameters that take advantage of the new simplified way of specifying job control.

ALTER Command

- CATALOG *dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- FILE *(dname)* - no longer needed because the DLBL and EXTENT statements are no longer required.
- SHAREOPTIONS - now allows you to indicate that files can be shared across VSE systems.

BLDINDEX Command

- INFILE *(dname)* - changed to INDATASET *(entryname)* because the DLBL and EXTENT statements are no longer required.
- OUTFILE *(dname)* - changed to OUTDATASET *(entryname)* because the DLBL and EXTENT statements are no longer required.
- WORKFILES*(dename)* - changed to WORKVOLUMES *(volser)* because the DLBL and EXTENT *statements are* no longer required.

CANCEL Command *(a new command)*

- Allows you to cancel the current job or job step.

DEFINE ALTERNATEINDEX

- CATALOG *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- DEFAULTVOLUMES - a new parameter used in conjunction with modeling (makes the VOLUMES parameter optional).
- FILE *(dname)* - no longer required because the DLBL and EXTENT statements are no longer required unless you specify the UNIQUE parameter.
- MODEL *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- NOALLOCATION - a new parameter that indicates no space allocation is to take place at define time.
- SHAREOPTIONS - now allows you to indicate that files can be shared across DOS systems.
- USECLASS - classes 2 to 6 have been added.

DEFINE CLUSTER

- CATALOG *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- DEFAULTVOLUME - a new parameter used in conjunction with modeling (makes the VOLUMES parameter optional).
- FILE *(dname)* - no longer required because the DLBL and EXTENT statements are no longer required unless you specify the UNIQUE parameter.
- MODEL *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- NAME *(entryname)* - has been expanded to allow you to take advantage of central processor and/or partition independence.

- NOALLOCATION - a new parameter that indicates no space allocation is to take place at define time.
- SHAREOPTIONS - allows you to indicate that files can be shared across VSE systems.
- USECLASS - classes 2 to 6 have been added.

Define MASTERCATALOG
- CLASS *(value)* - classes 2 to 6 have been added.
- DEDICATE - use this new parameter to indicate that the entire space (or remaining space) on a volume is to be allocated to the catlog.
- FILE *(dname)* - no longer needed because the DLBL and EXTENT statements are no longer required.
- ORIGIN *(tracknumber|blocknumber)* - use this new parameter to indicate the beginning point of the catalog's data space (this removes the need for an EXTENT statement).

Define NONVSAM
- CATALOG *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.

DEFINE PATH
- CATALOG *(dname)* - *dname* no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- FILE *(dname)* - no longer needed because the DLBL and EXTENT statements are no longer required).
- MODEL *(dname)* - *dname* subparameter no longer required. Specify *catname* instead of *dname* to identify the needed catalog.

DEFINE Space
- CATALOG *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- CLASS *(value)* - classes 2 through 6 have been added.
- DEDICATE - use this new parameter to indicate that the entire space (or remaining space) on a volume is to be allocated to VSAM.
- FILE *(dname)* - no longer needed because the DLBL and EXTENT statements are not required.
- ORIGIN *tracknumber|blocknumber)* - use this new parameter to indicate the beginning point of the data space (this removes the need for a DLBL and EXTENT statement).

Define USERCATALOG
- CLASS *(value)* - classes 2 through 6 have been added.
- DEDICATE - use this new parameter to indicate that the entire space (or remaining space) on a volume is to be allocated to the catalog.
- FILE *(dname)* - no longer needed because the DLBL and EXTENT statements are no longer required.
- MODEL *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- ORIGIN *(tracknumber|blocknumber)* - use this new parameter to indicate the beginning point of the catalog's data space. (This removes the need for a DLBL and EXTENT statements.)

DELETE
- CATALOG *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- FILE *(dname)* - no longer needed because the DLBL and EXTENT statements are no longer required.

EXPORT
- INFILE *(dname)* - no longer needed because the DLBL and EXTENT statements are no longer required.
- PRIMEDATADEVICE *(devtype)* - required only for a tape device.

EXPORTRA
- CRA *(dname1 dname2 dname3)* - changes to CRAVOLUMES ENTRIES *(entryname)* because the DLBL and EXTENT statements are no longer required.
- PRIMEDATADEVICE *(devtype)* - required only for a tape device.

IMPORT
- OBJECTS FILE *(dname)* - no longer needed because the DLBL and EXTENT statements are no longer required (unless you are importing a UNIQUE file).
- OBJECTS DEFAULTVOLUMES - directs VSAM to use the volumes given in a default list.
- OUTFILE *(dname)* - changed to OUTPW *(password)* because the DLBL and EXTENT statements are no longer required. OUTPW is used for password purposes only.
- PRIMEDATADEVICE *(devtype)* - required only for a tape device.

IMPORTRA
- OBJECTS FILE *(dname)* - no longer needed because the DLBL and EXTENT statements are no longer required (unless you are importing a UNIQUE file).
- OBJECTS DEFAULTVOLUMES - directs VSAM to use the volumes given in a default list.
- OUTFILE *(dname)* - no longer needed because the DLBL and EXTENT statements are no longer required.
- PRIMEDATADEVICE *(devtype)* - required only for a tape device.

LISTCAT
- CATALOG *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.

LISTCRA
- CATALOG *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- INFILE *(dname)* - changed to INVOLUMES *(volser)* because the DLBL and EXTENT statements are no longer required.

REPRO

- OUTFILE ENFIRONMENT PREDATADEVICE *(devtype)* - required only for a tape device.

RESETCAT

- CATALOG *(dname)* - *dname* subparameter no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- CRAFILES *(dname)* - changed to CRAVOLUMES *(volser)* because the DLBL and EXTENT statements are no longer required.
- WORKCAT *(dname)* - no longer required (also no DLBL and EXTENT statements are required). Specify *catname* instead of *dname* to identify the needed catalog.
- WORKFILE *(dname)* - changed to WORKVOLUMES *(volser)* because the DLBL and EXTENT statements are no longer required.

VERIFY

- FILE *(dname)* - changed to DATASET *(entryname)* because the DLBL and EXTENT statements are no longer required.

ACR Macro
New Operand

PARMS=(CLOSDSP=KEEP|DELETE|DATE) - specifies the CLOSE disposition for the file.

Preface

This book describes (1) the use of Access Method Services, a group of programs that provide utility functions vital to Virtual Storage Extended/Virtual Storage Access Method (VSE/VSAM), and (2) how to code VSAM macro instructions to process data. This publication provides the VSAM information needed to use Access Method Services in order to establish and maintain VSAM files, and information on the VSAM macros. For additional information about VSE/VSAM, see *VSE/VSAM General Information*, GC24-5143.

If you want information about the VSE/VSAM Space Management for SAM Feature, refer to *Using the VSE/VSAM Space Management for SAM Feature*, SC24-5192.

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

This book has the following major groupings of information:

- Chapter 1: Introduction to Access Method Services, which provides an overview of Access Method Services, including general language considerations and Access Method Services functions.
- Chapter 2: Using Access Method Services, which gives examples of general job control relationships and format to be used with Access Method Services commands. Each command is discussed in its own section, explaining functions performed and examples that show how to code typical cases.
- Chapter 3: Access Method Services Commands, which sets out the format for each functional and modal command.
- Chapter 4: Introduction to the VSAM Macros, which briefly discusses the VSAM macros.
- Chapter 5: VSAM Macro Instruction Format, which describes the coding of each macro and includes examples of each one.
- Appendix A: Sample Job Streams, which gives examples with complex combinations of commands.
- Appendix B: Interpreting LISTCAT Output, which provides information on the structure of LISTCAT output and shows sample output.
- Appendix C: Interpreting LISTCRA Output, which provides information on the structure of LISTCRA output and shows sample output.
- Appendix D: Invoking Access Method Services from a Problem Program, which provides information on linkage convention, control blocks, and macros needed to invoke utility.
- Appendix E: Command Parameters Summary, which lists the commands, command parameters with their abbreviations, and examples of command use.
- Appendix F: Operand Notation for VSAM Macros, which provides information on the ways to express operands.
- Appendix G: Parameter Lists for VSAM Macros, which describes the format of the parameter lists and gives the codes used for the operands of each of the macros.
- Appendix H: Making the VSAM Master Catalog Recoverable, which shows how to convert the VSAM master catalog of an existing VSE system from a nonrecoverable to a recoverable catalog.
- Appendix I: Password Requirements, which provides a summary of the passwords required with the Access Method Services commands and other password information.
- Appendix J: Acronyms and Abbreviations, which defines acronyms and abbreviations relevant to Access Method Services and VSAM.
- Appendix K: Glossary, which defines terms relevant to Access Method Services and VSAM.

Required Publications

You should be familiar with pertinent information presented in the following publication:

VSE/VSAM General Information, GC24-5143.

Related Publications

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

VSE/VSAM Messages and Codes, SC24-5146

VSE/VSAM Programmer's Reference, SC24-5145

Using the VSE/VSAM Space Management for SAM Feature, SC24-5192

VSE/VSAM Documentation Subset, SC24-5191

Contents

Figures

Chapter 1: Introduction to Access Method Services

Access Method Services is a service program that is used with VSE/VSAM (Virtual Storage Extended/Virtual Storage Access Method) to create and maintain files. Access Method Services enables you to request various functions such as defining a VSAM file and loading records into it, converting a sequential file on tape or disk or an indexed-sequential file to the VSAM format, listing VSAM catalog information or file records, copying a file for reorganization, creating a backup copy of a file, and making a file portable from one operating system to another.

You can invoke Access Method Services functions by executing the IDCAMS program (// EXEC IDCAMS) and issuing a command and its parameters in the job stream that IDCAMS processes. (See *VSE/VSAM Programmer's Reference* for information on job control.)

You can also call the IDCAMS program from within another program and pass a command and its parameters to the IDCAMS program. (See "Appendix D: Invoking Access Method Services from a Problem Program.")

If you plan on using VSE/VSAM, you must use the commands provided by Access Method Services. This chapter introduces the commands available through Access Method Services and the functions they perform.

Functions of Access Method Services

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

Function		Command
Add	a password to a new VSAM file or a catalog	DEFINE
	a password to an existing VSAM file or a catalog	ALTER
Analyze	a nonrecoverable catalog where data access is impaired	LISTCAT
	a recoverable catalog where data access is impaired	LISTCRA
Attach	a user catalog to the master catalog	DEFINE,IMPORT
Build	an alternate index	BLDINDEX
Cancel	a job	CANCEL
	a job step	CANCEL
Catalog	a VSAM file	DEFINE
	a nonVSAM file	DEFINE
Change	a file's description in the catalog	ALTER
	the device type of the volume on which the catalog resides	REPRO
	a password	ALTER
Connect	a user catalog to a master catalog	IMPORT
Convert	a SAM or ISAM file to VSAM format	REPRO
	a VSAM file to sequential format	REPRO
Copy	a file	REPRO

Figure 1-1. Functions and Commands (Part 1 of 2)

Function		Command
Create	an alternate index	DEFINE,BLDINDEX
	a backup copy of a catalog	REPRO
	a backup copy of a VSAM file	REPRO,EXPORT
	a catalog	DEFINE
	a catalog entry for a nonVSAM file	DEFINE
	a path	DEFINE
	a VSAM file	DEFINE
Define	an alternate index	DEFINE
	a catalog	DEFINE
	a data space	DEFINE
	a nonVSAM file	DEFINE
	a path	DEFINE
	a VSAM file	DEFINE
Delete	an alternate index	DELETE
	a catalog	DELETE
	a data space	DELETE
	a nonVSAM file	DELETE
	a password	ALTER
	a path	DELETE
	a VSAM file	DELETE
Disconnect	a user catalog from the master catalog	EXPORT
Enter	a file entry into the catalog	DEFINE
List	a password	LISTCAT
	a file	PRINT
	a file's catalog entry	LISTCAT
	contents of the catalog	LISTCAT
	contents of the catalog recovery area	LISTCRA
Load	records into a file	REPRO
	a catalog from an unloaded copy	REPRO
Modify	a file's description in the catalog	ALTER
Move	a catalog to another system	EXPORT,IMPORT
	a file to another system	EXPORT,IMPORT
Print	a file	PRINT
Recover	from loss of data due to improper closing of a file	VERIFY
	from possible loss of data or catalog entries by means of catalog recovery areas	EXPORTRA,IMPORTRA, and RESETCAT
Recreate	a VSAM file from a back up copy	IMPORT
Release	a user catalog from the master catalog	EXPORT
Reload	a catalog from a sequential file	REPRO
Rename	a file	ALTER
Reorganize	a file	REPRO or EXPORT,IMPORT
Uncatalog	a file	DELETE
Unload	a catalog to a sequential file	REPRO
	a file	REPRO
Verify	end-of-file	VERIFY

Figure 1-1. Functions and Commands (Part 2 of 2)

Functional Commands

There are two types of Access Method Services commands: functional commands that are used to request the actual work – for example, defining a file or listing a catalog – and modal commands that specify options and allow the conditional execution of the functional commands.

The functional commands are:

- ALTER, which is used to alter existing catalog entries.
- BLDINDEX, which is used to build alternate indexes for existing VSAM files.
- CANCEL, which is used to cancel a job or job step.
- DEFINE, which is used to create catalogs and catalog entries for files, alternate indexes, paths, and VSAM data spaces.
- DELETE, which is used to delete catalog entries.
- EXPORT, which is used to create a copy of a file for backup or to make a file or user catalog portable so that it can be used in another system.
- EXPORTRA, which is used to recover data independent of the status of a catalog by means of duplicate catalog entries in catalog recovery areas (CRAs).
- IMPORT, which is used to read a backup copy of a file or to make a file or user catalog that was previously exported from one system available for use in another system.
- IMPORTRA, which is used to make data recovered via the EXPORTRA function again available to the user.
- LISTCAT, which is used to list catalog entries.
- LISTCRA, which is used to list catalog recovery areas or compare the entries of a catalog recovery area with the appropriate entries in a catalog to check whether the catalog and the catalog recovery area are synchronized.
- PRINT, which is used to print VSAM and nonVSAM files.
- REPRO, which is used to copy files, to convert sequential files on tape or disk and indexed-sequential files to VSAM format, to convert VSAM and indexed-sequential files to sequential format, to create backup copies of VSAM catalogs, and to reload a VSAM catalog from a backup copy.
- RESETCAT, which is used to reset a recoverable catalog to the level of its owned volumes. The owned volume's CRA information is used to synchronize the catalog with the volume(s).
- VERIFY, which is used to cause a catalog to correctly reflect the end of a file. You should use this command after an error occurred which prevented normal closing of the file that may have caused the catalog to be incorrect.

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

Modal Commands

The modal commands are:

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

How to Code Access Method Services Commands (Syntax)

Notational Conventions

A uniform system of notation describes the format of Access Method Services commands. This notation (that is, [ ], { }, |, ...) 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; 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. In examples, an ellipsis may indicate that entries not relevant to the examples have been omitted.
- Underscored type indicates a default option. If the parameter is omitted and none is implied by the presence of another parameter in the set, the underscored value is assumed.
- Other punctuation (parentheses, commas, spaces, etc.) must be entered as shown. A space is indicated by ␣.
- UPPER CASE type indicates the exact characters to be entered. Such items must be entered exactly as illustrated (in upper case).
- *Italic* type specifies fields to be supplied by the user.

Syntax of the Language

All Access Method Services commands have this general structure:

VERB␣ *parameter(s) terminator*

VERB specifies the type of service requested. *Parameter(s)* further describe the service requested. *Terminator* indicates the end of the command statement. Refer to "Appendix A: Sample Job Streams" for command statement examples.

You can abbreviate many of the verbs and parameters. Permitted abbreviations are listed with each command statement or parameter, in "Appendix K: Acronyms and Abbreviations," and in "Appendix E: Command Parameter Summary."

Verbs

Code the verbs beginning at or to the right of the left margin. The default columns are column 2 through 72. You can specify different columns, if you want, with the PARM command.

You must separate each verb from its parameter(s) either with a blank or a comment.

Comments

You can add comments to a command statement anywhere a blank can appear. Comments are strings of characters preceded by a /* and followed by an */. They can contain any characters you desire except an */. You continue comments to the next line by using a hyphen (-) or plus (+) sign as the last non-blank character before or at the right margin.

Parameters

Parameters are in bold uppercase type (they must be coded exactly as shown) or in italicized lowercase type (you substitute an appropriate value or word for the parameter).

One or more parameters follow the verb; they can be *positional parameters* or *keyword parameters*. Positional parameters follow the verb in a prescribed sequence. An example of a positional parameter in this manual is the *entryname* parameter in the ALTER, DELETE, and EXPORT commands; it must always be the first parameter following its respective command. If you specify a positional parameter that consists of a list of names, you must enclose the list in parentheses (a single item does not require parentheses).

Keyword parameters are specific names that have a particular meaning to Access Method Services. You can include these parameters in any order following the positional parameter, if present, or the verb. Keyword parameters are shown in upper-case type; for example: ERASE.

You can also specify values or names with some of the keyword parameters. You enclose the value or name within parentheses following the parameter; for example: DEVICETYPE(3340).

If the value you specify contains commas, semicolons, blanks, parenthesis, or slashes you must enclose the entire value in single quotation marks.

A keyword parameter might also have a set of subparameters; for example:

```
ENV(PDEV(2400) RECFM(VARUNB) BLKSZ(5164))
```

In the preceding example, (PDEV) PRIMEDATADEVICE, (RECFM) RECORDFORMAT, and (BLKSZ) BLOCKSIZE are subparameters of the (ENV) ENVIRONMENT keyword parameter. (The parameters are abbreviated so that the example fits on a single line.)

You can code a list of similiar items (for example, volume serial numbers) in a parameter or subparameter. Separate the parameters and subparameters from one another by one or more separators (commas, blanks, or comments). The only exception is that parameters immediately preceding or following a subparameter set enclosed in parentheses do not need to be separated from the opening or closing parenthesis.

In some cases it is necessary to specify a password following the name of a catalog entry or of a job control statement. You do this by coding (1) the name, (2) a slash, and (3) the password. The name/password combination must be preceded and followed by separators; for example:

```
DELETE PAYROLL/CTLGPAY.
```

Continuing Command Statements

When you continue a command statement on more than one coding line (coding line is sometimes referred to as a record) you must use a hyphen (-) or plus (+) sign to indicate the continuation of the command statement. (The coding examples in this manual generally use a separate line for the verb and each of its parameters; this makes it easier to read the command statement.) The plus sign differs from the hyphen because not only does it indicate continuation of the command statement but it also indicates continuation of a value within the command statement.

Blank coding lines or coding lines ending with complete comments (that is, *end of comment* */) must also end with a continuation mark when they appear in the middle of a command statement and when they appear preceding or following the THEN and ELSE clauses of an IF command.

Records ending with partial comments must always end with a continuation mark.

Also, only blank characters may appear between a continuation mark and the end of the coding line.

Common Continutation Errors

The continuation rules must be used carefully when modal commands, comments, or blank records appear in the input stream. You must be careful when continuing a modal command so that you do not inadvertently specify a null clause. For information on null clauses, see "Null Clauses" in "Modal Command Format."

The following examples show common continuation errors, (see "Appendix A: Sample Job Streams" for examples of correct coding).

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

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

```
IF LASTCC = 0 -
THEN -
REPRO INFILE(F1)OUTFILE(F2)
/*ALTERNATE PATH*/
ELSE -
PRINT...
```

Because no continuation mark (hyphen) follows the comment, a null ELSE clause is assumed. The ELSE keyword will not match up with the THEN keyword, an error message will be issued, and the PRINT command ignored. Note the correct use of the continuation marks on the other records.

```
IF LASTCC = 0 -
THEN-
REPRO . . .
ELSE -
PRINT . . .
```

Because a blank line with no continuation mark (hyphen) follows the ELSE keyword, the ELSE becomes null and the PRINT command is unconditionally executed.

```
PARM TEST ( - /*COMMAND*/
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 after the comment. The command is rejected.

Terminator

The terminator indicates the end of the command. The terminator can be a semicolon or simply the absence of a continuation mark. 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 code:

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

the characters following the semicolon terminator are ignored. The continuation marks (hyphens) at the end of the first and second records cause the PRINT command to be ignored also. The first PARM command and the REPRO command are the only commands that are recognized.

Chapter 2: Using Access Method Services

Using Job Control

Chapter 2 discusses the various functions of Access Method Services and gives examples that combine job control statements with Access Method Services commands for each function. Many of the command parameters are not covered in this chapter because the intent here is to provide you with a basic understanding of the Access Method Services commands. More complex examples that combine sequences of commands are given in Appendix A.

Access Method Services DEFINE commands and job control statements are used to set up all catalogs, VSAM data spaces, VSAM files (clusters), alternate indexes, paths, and nonVSAM files.

A VSAM file exists once it has been defined in a catalog. Before you can define a file, you must have a catalog in which to define it. Generally you must also have a VSAM data space on a direct-access volume in which to suballocate space for the file (as opposed to defining a catalog or defining a file in its own unique data space). Once a VSAM file has been defined, records can be loaded into it.

When you invoke Access Method Services via job control, you must specify an EXEC job control statement:

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

You must specify the SIZE parameter, otherwise Access Method Services will terminate your job immediately. The SIZE parameter provides information to the VSE system which allows it to divide the processing partition into a static area and a GETVIS area. When you execute IDCAMS, only the first load, called the root segment, is loaded into the static area. The remainder of the partition must be left free for GETVIS area required by Access Method Services and for the subsequent modules it loads. When you specify SIZE=AUTO, the system determines the amount of virtual storage required for the IDCAMS root segment and leaves the rest of the partition free for the GETVIS area.

See *VSE/VSAM Programmer's Reference* for more information on job control.

Using DEFINE: Defining Objects in a Catalog

VSAM uses catalogs as central information points for VSAM files and the direct-access volumes on which they are stored. You can define a VSAM object in a VSAM catalog 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 specify the DEFINE command, Access Method Services builds a catalog entry that describes the object. *The VSAM objects you can define are:*

- *Master catalog,* which is the primary VSAM catalog. It contains a collection of information about VSAM objects in the system. You must create a VSAM master catalog before you can define any other object.
- *User catalog,* which is a collection of information about VSAM objects that reside on the volumes owned by the user catalog. You can create a user catalog after the VSAM master catalog is defined. A pointer to the user catalog is put in the master catalog.
- *Data Space,* which is direct-access device space used for a catalog, for VSAM clusters, for alternate indexes, and for catalog recovery areas. VSAM controls the allocation of space within each data space.
- *Cluster, or VSAM file,*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 sequentialy.
  - 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 file name for the combination of an alternate index and its base cluster or an alias for a VSAM file. A path entry can be password protected.
- *Non-VSAM file,* which is a file that is not in VSAM format, but can be cataloged in a nonrecoverable VSAM catalog.

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, and catalogs are stored in data spaces. You can allow a data space to contain many clusters and alternate indexes (sometimes called files) 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 file. In the usual case, you define a data space first, then define the files. To define a VSAM file as the only one in its data space, specify the parameter UNIQUE when you define the file.

Defining an object doesn't normally require that a volume be mounted, because Access Method Services can determine the availability of space in data spaces by examining the volume information in the catalog. A volume must be mounted whenever a data space, a unique file, or a catalog is being defined, deleted, or altered or whenever a VSAM file is being defined, deleted, or altered in a recoverable catalog. (If the volume is not mounted, VSAM issues a mount message.)

See "Volume Mounting Requirements" in *VSE/VSAM Programmer's Reference* for more information.

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. See "How to Use One Object as a Model for Another Object and Override System Defaults" in *VSE/VSAM Programmer's Reference* for more information.

Defining a Catalog

The DEFINE command is used to define the master catalog or user catalogs. User catalogs are pointed to by the master catalog.

The data space that contains the catalog is allocated when the catalog is defined. The data space may be reserved for the catalog's exclusive use or it may contain other VSAM files.

Catalog Space Estimates and Worksheet

The control intervals in a catalog are 512 bytes long, and each control interval contains one record.

If you specify (or default to) the IMBED option, each control area in the data component of the catalog takes up:

- Five tracks (min-CAs) on a 2314 (2319) or 3340,
- Three tracks on a 3330, 3330-11, or 3350, and
- Three min-CAs on a 3310 or 3370.

The first track is for the sequence set record, which is placed adjacent to (imbedded in) the control area.

If you specify the NOIMBED option, each control area in the data component of the catalog takes up:

- Four tracks on a 2314 (2319) or 3340,
- Two tracks on a 3330, 3330-11, or 3350, and
- Two min-CAs on a 3310 or 3370.

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

Variable Quantities	Formulas	Estimates
Basic Requirement	10	10
A = number of key-sequenced files	3*A	
B = number of entry-sequenced files	2*B	
C = number of relative-record files	2*C	
D = number of alternate indexes	3*D	
E = number of paths	E	
F = number of nonVSAM files	F	
G = number of volumes, depending on device, owned by the catalog: G_1 = 3* (number of 2314/19 or 3310 volumes) G_2 = 4* (number of 3330, 3340, or 3370 volumes) G_3 = 6* (number of 3330-11 or 3350 volumes)	$G=G_1+G_2+G_3$	
H = for each key-sequenced file with more than 2 volumes, 1 for each additional group of 1 to 5 volumes For example: H = 0 for each key-sequenced file of 2 volumes H = 1 for each key-sequenced file of 7 volumes H = 2 for each key-sequenced file of 8 volumes	H	
I = for each alternate index with more than 2 volumes, 1 for each additional group of 1 to 5 volumes.	I	
J = for each entry-sequenced file with more than 5 volumes, 1 for each additional group of 1 to 8 volumes.	J	
K = for each relative-record file with more than 5 volumes, 1 for each additional group of 1 to 8 volumes.	K	
L = for each group of 4 data spaces on a volume, 1.	L	
M = number of entry records (subtotal of the previous items).	M	
N = number of records in the catalog's data component.	N=1.4*M	
O = number of records in the catalog's index component where: If IMBED is specified or defaulted X = 2 for the 3310, 3330, 3330-11, 3350, or 3370 X = 4 for the 2314 (2319) or 3340 Y = 55 for the 2314 (2319) Y = 60 for the 3330, 3330-11, or 3340 Y = 81 for the 3350 Y = 96 for the 3310; 186 for the 3370 If NOIMBED is specified X = 40 for the 3330 or 3330-11 X = 44 for the 2314 (2319) X = 48 for the 3340 or 3350 X = 64 for the 3310; 124 for the 3370 (Round O up to a multiple of X.)	$O=(N/X)+Y$ $O=N/X+N/X^2$	
P = total number of records (If N+O is less than 200, P=200)	P=N+O	
Q = total number of Min-CAs (tracks) where: Q = P/11 for the 2314 (2319) Q = P/20 for the 3330 or 3330-11 Q = P/12 for the 3340 Q = P/27 for the 3350 If IMBED is specified or defaulted, round Q up to the next multiple of 5 for the 2314 (2319) or 3340, or to 3 for the 3330, 3330-11, or 3350. If NOIMBED is specified, round Q up to the next multiple of 4 for the 2314 (2319) or 3340, or to 2 for the 3330, 3330-11, or 3350. For the 3310, Q=total number of blocks, where: Q is in units of blocks and is obtained by rounding P up to the next multiple of 96 for IMBED and 64 for NOIMBED. For the 3370, Q=total number of blocks, where: Q is in units of blocks and is obtained by rounding P up to the next multiple of 186 for IMBED and 124 for NOIMBED.	Q	

The Master Catalog

The *first job you run* after you have installed VSAM and Access Method Services in your system is the one that creates your master catalog. Without a master catalog you cannot define user catalogs, data spaces, or files. The volume on which the master catalog is defined must be mounted whenever VSAM is being used; it is always on a logical unit named SYSCAT. You can have more than one master catalog at your installation; however, only one can be connected to the system at a time. It is connected to the system at IPL by the DEF SYSCAT=cuu command.

How Data Space is Assigned to a Catalog

When a master or user 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. Two important points about a catalog and the catalog's data space need to be understood:

- VSAM allocates a specific amount of data space to a catalog (the catalog "owns" this space).
- All or a portion of that specific amount of data space can be made available (suballocated) for the catalog itself. If only a portion of the total amount of data space is suballocated to the catalog, the remaining data space (still owned by the catalog) is available for a catalog recovery area (if RECOVERABLE had been specified), for future expansion of the existing catalog (only if you specify a secondary allocation value), or for other VSAM objects.

You can assign data space to a catalog in one of the following ways, by specifying:

- The DEDICATE parameter (at the catalog level only), an optional space allocation parameter(s) (at the data or index component levels), and a DLBL job control statement (required for the master catalog only).
- The ORIGIN parameter, a space allocation parameter(s), and a DLBL job control statement (required for the master catalog only). Note that the first cylinder (max-CA) on a volume is never specified in an ORIGIN parameter; one reason is because a portion of the first cylinder (by default) contains the VTOC and therefore is not available for your use.
- A space allocation parameter(s), no DEDICATE or ORIGIN parameters, and a DLBL job control statement (required for the master catalog only).

Using the DEDICATE Parameter for a Catalog: You use the DEDICATE parameter to specify that all the unowned and unallocated (free) space on a specified volume is to belong to a catalog. Note that VSAM acquires all the free space for the catalog if 16 or fewer extents exist on the volume (where extents are contiguous areas of free space); if more than 16 "free" extents exist on the volume, VSAM acquires only the first 16 extents for the catalog.

You can specify DEDICATE at the catalog level only; it is mutually exclusive with the space allocation parameters (CYLINDERS, BLOCKS, RECORDS, TRACKS). The amount of space to be suballocated to the catalog itself, depends on how (or if) you specify a space allocation parameter at the index component and/or data component levels.

If you are defining a master catalog, a DLBL job control statement is required; it can appear in the job stream or it can reside on the permanent standard label area (see "Specifying a VSAM Catalog's Job Control

Statements" in *VSE/VSAM Programmer's Reference* for further information).

- If you specify DEDICATE and no space allocation parameter at the data and index component levels (see EXAMPLE 1), the acquired free space on the designated volume is owned by the catalog. The amount of space suballocated to the catalog itself is calculated by VSAM to be enough space for a minimum size catalog (this amount is device-dependent); the remaining space is available for other VSAM objects.

```
// JOB  EXAMPLE 1
// DLBL IJSYSCT,'VSAMCAT',,VSAM
// EXEC IDCAMS,SIZE=AUTO
   DEFINE MASTERCATALOG -
          (NAME(VSAMCAT) -
           VOLUME(CATVOL) -
           DEDICATE)
/*
/&
```

- If you specify DEDICATE and a space allocation parameter at the data and index component levels (see EXAMPLE 2), the acquired free space on the designated volume is owned by the catalog, and the space suballocated to the catalog is the sum of the data and index level specifications. The space not suballocated to the catalog (if any) is available for other VSAM objects.

```
// JOB  EXAMPLE 2
// EXEC IDCAMS,SIZE=AUTO
   DEFINE USERCATALOG -
          (NAME(USER.CAT) -
           VOLUME(USRVOL) -
           DEDICATE) -
          DATA(CYLINDERS(10)) -
          INDEX(CYLINDERS(2))
/*
/&
```

- If you specify DEDICATE and a space allocation parameter at the data component level (see EXAMPLE 3), the acquired free space on the designated volume is owned by the catalog. VSAM uses your data component specification to calculate a value for the index component and it then adds this value to the data component specification. This sum becomes the amount of data space that is suballocated to the catalog. The space not suballocated to the catalog (if any) is available for other VSAM objects.

```
// JOB  EXAMPLE 3
// DLBL IJSYSCT,'VSAMCAT',,VSAM
// EXEC IDCAMS,SIZE=AUTO
   DEFINE MASTERCATALOG -
          (NAME(VSAMCAT) -
           VOLUME(CATVOL) -
           DEDICATE) -
          DATA(CYLINDERS(6))
/*
/&
```

Using the ORIGIN Parameter for a Catalog: You use the ORIGIN parameter to specify the beginning point (track number or block number) of the catalog's data space. VSAM determines the ending point of the data space from the value you specify *(at the catalog level)* for CYLINDERS, BLOCKS, RECORDS, or TRACKS (VSAM converts records to tracks). If you specify a block number value (in the ORIGIN parameter) that does not coincide with a min-CA (track) boundary, VSAM rounds the value up to the next min-CA boundary. If the ending block value (ORIGIN plus number of

blocks) does not coincide with a min-CA boundary, VSAM rounds it down to the previous min-CA boundary.

You can specify ORIGIN at the catalog level only. If you are defining a master catalog, a DLBL statement is required. It can reside on the permanent standard label area or it can appear in the job stream. You have three different ways of allocating space with the ORIGIN parameter, you can:

- Specify ORIGIN and a space allocation parameter *at the catalog level only* (see EXAMPLE 4). In this case VSAM divides the amount of space you specified between the catalog's data and index components. The entire amount of space is owned by and available to the catalog only.

```
// JOB  EXAMPLE 4
// EXEC IDCAMS,SIZE=AUTO
   DEFINE USERCATALOG -
          (NAME(USER.CAT) -
           VOLUME(USRVOL) -
           ORIGIN(20) -
           TRACKS(260))
/*
/&
```

- Specify ORIGIN *at the catalog level only* and a space allocation parameter *at the catalog, data component, and index component levels* (see EXAMPLE 6). In this case VSAM adds the data component value to the index component value to arrive at the total amount of space to be suballocated to the catalog; it (the sum) cannot be greater than the space allocation value specified at the catalog level. If the sum is less than the catalog level specification, the remainder is available for other VSAM objects.

```
// JOB  EXAMPLE 5
// DLBL IJSYSCT,'VSAMCAT',,VSAM
// EXEC IDCAMS,SIZE=AUTO
   DEFINE MASTERCATALOG -
          (NAME(VSAMCAT) -
           VOLUME(CATVOL) -
           ORIGIN(20) -
           TRACKS(60)) -
          DATA(TRACKS(40)) -
          INDEX(TRACKS(20))
/*
/&
```

- Specify ORIGIN *at the catalog level only* and a space allocation parameter *at the catalog and data component levels* (see EXAMPLE 6). In this case VSAM uses your data component specification to calculate a value for the index component. (It is up to you to provide enough space, via the catalog level specification, for the index component.) VSAM then adds the calculated value to the value specified in the data component. This amount is then suballocated to the catalog; it cannot be greater than the space allocation value specified at the catalog level. If the sum is less than the catalog level specification, the remainder is available for other VSAM objects.

```
// JOB  EXAMPLE 6
// EXEC IDCAMS,SIZE=AUTO
   DEFINE USERCATALOG -
          (NAME(USER.CAT) -
           VOLUME(USRVOL) -
           ORIGIN(20) -
           TRACKS(80)) -
          DATA(TRACKS(40))
/*
/&
```

Using Neither DEDICATE nor ORIGIN for a Catalog: By not specifying the

DEDICATE or ORIGIN parameters (see EXAMPLE 7), you indicate that you want VSAM to choose the first available extent on the volume that is large enough to contain your primary allocation (VSAM, in effect, defaults to ORIGIN). The space allocation parameter (TRACKS, CYLINDERS, RECORDS, or BLOCKS) determines the data space allocation for the VSAM catalog.

If you are defining a master catalog, a DLBL job control statement is required. It can reside on the permanent standard label area or it can appear in the job stream.

```
// JOB  EXAMPLE 7
// EXEC IDCAMS,SIZE=AUTO
   DEFINE USERCATALOG -
          (NAME(USER.CAT) -
           VOLUME(USRVOL) -
           TRACKS(260))
/*
/&
```

Allocating Space to the Catalog's CRA

If you define a catalog as recoverable, one cylinder (max-CA) of the total data space owned by the catalog is allocated to the CRA (catalog recovery area). The catalog definition will fail if insufficient space exists for the CRA. Use the following guidelines as an aid in specifying space allocation parameters for a recoverable catalog.

Allocating Space to a Recoverable Catalog via the DEDICATE Parameter: If you specify DEDICATE, VSAM (not you) sets aside space for your CRA.

Allocating Space to a Recoverable Catalog via the ORIGIN Parameter: If you specify ORIGIN and a space allocation value at the catalog level only, VSAM (not you) sets aside space for your CRA from the space you specified.

If you specify ORIGIN and a space allocation value at the catalog and data component level (or the catalog, data component, and index component levels), you must make the catalog level space allocation value large enough to contain one cylinder for the CRA and still have sufficient space for the catalog's data and index components.

Defining a VSAM Data Space

The DEFINE command is used to define VSAM data spaces or to reserve volumes for VSAM's future use.

VSAM Data Spaces On a Volume

A VSAM data space is space on a direct-access volume that is owned and managed by VSAM. When you define the volumes'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. You must define all future data spaces and VSAM objects on that volume in that same catalog.

A data space can take up all or part of the space on one volume; *it cannot take up space on more than one volume.* A volume can contain more than one data space.

When you define a data space with the DEFINE SPACE command, you must specify the volume that is to contain the data space. If you specify more than one volume, a data space of the size you specify is allocated on each volume and each volume comes under 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. The volume(s) on which data space is to be defined must be mounted for the define.

If there is space available and there is no volume ownership conflict (no other catalog owns this volume), VSAM ownership is indicated in the volume's VTOC with a format-4 label and a VSAM data space is allocated on the volume. See "VSAM Volume Ownership" in *VSE/VSAM Programmer's Reference* for more information on volume ownership.

You can assign space to one of eight performance classes, that is, you can classify space as standard (non-fixed head) space, fixed-head space, or whatever other space criteria you wish to choose. See"Data Space Classification" in *VSE/VSAM Programmer's Reference* for more information.

If the data space is the first data space on a volume owned by a recoverable catalog, VSAM also allocates one cylinder (max-CA) of space to a CRA (catalog recovery area).

VSAM Objects in a Data Space

VSAM catalogs, clusters, and alternate indexes are stored in VSAM data spaces. A *unique* object is defined as the only object in its data space. VSAM allocates the data space for a unique object when you define the object. You cannot define a unique alternate index or cluster in an "empty" volume if that volume is to be owned by a recoverable catalog; you must have already defined space on that volume (via DEFINE SPACE) so that space exists for the CRA.

A *nonunique (suballocated)* object can share a data space with other VSAM objects. That is, a catalog, clusters, and alternate indexes might also be in the same data space. To define a file that can share a data space with other files, a data space must have been defined beforehand in which to allocate space for the file.

A data space can contain more than one suballocated file, and a file can be stored in more than one data space, on the same volume or on different volumes.

Space Assignment to VSAM Objects

When VSAM suballocates space for a file, it suballocates the space from space that is available in an existing data space. Data spaces (this includes unique files and their associated data space) cannot be dynamically extended in VSE. You must define more space (or free existing space) whenever a define or extend fails with an insufficient space error. If all of the space given to a unique file at DEFINE time is used, the unique file cannot be further extended, even by the definition of new data spaces.

Three examples of defining a data space are shown below. (See also Example 2 in "Appendix A: Sample Job Stream.") Assume for these examples that the data space is to be cataloged in the master catalog and the master catalog's DLBL statement is in the standard label area. Standard format-1 and format-3 labels describing the data space are written into the volume's VTOC.

EXAMPLE 8 specifies that 209 tracks, beginning with track 19, are to be allocated to VSAM (assume that a 3330 volume is being used).

```
// JOB  EXAMPLE 8
// EXEC IDCAMS,SIZE=AUTO
   DEFINE SPACE -
          (TRACKS(209) -
```

```
            VOLUME(M3330A) -
            ORIGIN(19)) -
           CATALOG(MASTCAT/UPDPW1)
/*
/&
```

EXAMPLE 9 is a "default ORIGIN" example; by not providing the ORIGIN (or DEDICATE) parameter you cause VSAM to choose the first possible extent on the volume that can satisfy your space allocation quantity (in this case, 209 tracks).

```
// JOB  EXAMPLE 9
// EXEC IDCAMS,SIZE=AUTO
   DEFINE SPACE -
          (TRACKS(209) -
           VOLUME(M3330A)) -
          CATALOG(MASTCAT/UPDPW1)
/*
/&
```

EXAMPLE 10 indicates that the unowned and unallocated data space (maximum of 16 extents per volume) on volume VSER01 and VSER02 is to be allocated to VSAM.

```
// JOB  EXAMPLE 10
// EXEC IDCAMS,SIZE=AUTO
   DEFINE SPACE -
          (DEDICATE -
           VOLUME(VSER01 VSER02)) -
          CATALOG(MASTCAT/UPDPW1)
/*
/&
```

Defining a VSAM File (Cluster)

A VSAM file can be suballocated, not allocated (dynamic), or unique. A *suballocated* file shares a data space with other files; an *unallocated (dynamic)* file has no space allocated to it at define time; a *unique* file has a data space to itself.

To define a *suballocated* VSAM file, you first define a data space, then use the DEFINE CLUSTER command. VSAM suballocates space for the file in the data space you have set up and enters information about the file in a VSAM catalog. (No records are loaded into the file at this time; defining a file is distinct from loading records into it.) A file can be stored in more than one data space on the same volume or on different volumes. See "Defining a Suballocated File" for more information and examples.

To define a *dynamic* VSAM file, you specify the NOALLOCATION and REUSE parameters at define time. The required space (specified with a space allocation parameter at define time) is suballocated to the file when VSAM opens it. See *VSE/VSAM Programmer's Reference* for more information about dynamic files.

To define a *unique* VSAM file, you do not define the data space beforehand. You specify the parameter UNIQUE in the DEFINE CLUSTER command and assign space to the file with a space allocation parameter and the DLBL/EXTENT job control statements. The data space is acquired and assigned to the file concurrent with the file definition. The volume(s) to contain a unique file must be mounted as in defining a data space. See "Defining a Unique VSAM File" for more information and examples.

With a multivolume key-sequenced file, you may assign data to 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. (The keys could also be A-D, G-K, L-O, R-W, etc.)'

VSAM treats all files as clusters. A cluster consists of a data component only (in the case of an entry-sequenced file or a relative-record file) or of a data component and an index component (in the case of a key-sequenced file). Besides setting up a catalog entry for each component of a cluster, VSAM sets up a catalog entry for the cluster as a whole. This entry consists mainly of the name (the 44-byte *file-ID*) of the cluster which you specify in the DEFINE command, and, for a key-sequenced file, an indication of the relationship between the data component and the index component.

You can also specify names for the index and data components of a cluster in the INDEX and DATA parameters of the DEFINE CLUSTER command. (If you do not provide names for the data and index components, VSAM generates them.) These names enable you to process each component individually. For example, you may open the index of a key-sequenced file separately and process it as data (with addressed or control interval access).

You can also specify that the data and index components of a key-sequenced file are to reside on different volumes; you do this by specifying the VOLUMES parameter as an attribute of both DATA and INDEX.

A cluster is defined in the master catalog unless you specify a job catalog or indicate otherwise through the CATALOG parameter.

Specifying Information that Defines a File

When a DEFINE command is used to define a key-sequenced cluster, three entries are created in the catalog: an entry for the cluster, its data component, and its index component.

Attributes of the data and index components can be specified separately from the attributes of the cluster as a whole. 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 component specification overrides the cluster specification.

See the DEFINE CLUSTER command in "Access Method Services Commands" to identify which parameters can be used with each type of entry.

The following examples are intended to demonstrate how to use the basic DEFINE parameters to define a file without considering all of the options. See also Examples 3 and 4 in "Appendix A: Sample Job Streams."

Defining a Suballocated VSAM File

When a VSAM file is defined and space is suballocated for it in one or more existing data spaces, DLBL and EXTENT statements are not required and the volume(s) on which the file is defined need not be mounted. An example of defining a suballocated key-sequenced file is shown below. The cluster is defined in the master catalog (no DLBL job control statement is given for the master catalog because it is assumed that it was previously entered in the standard label area).

```
// JOB  EXAMPLE 11
// EXEC IDCAMS,SIZE=AUTO
   DEFINE  CLUSTER(-
             NAME(MSTRFILE)-
             VOLUMES(M3330A) -
             TRACKS(38 19)-
             KEYS(10 1)-
             RECORDSIZE(80 80))-
           CATALOG(MASTCAT/UPDPW1)
/*
/&
```

The volume on which the file will reside is indicated in the VOLUME parameter. Space to be allocated initially (primary allocation) and, optionally, space to be allocated if the file must be extended (secondary allocation) are indicated in the TRACKS, CYLINDERS, BLOCKS or RECORDS parameter. VSAM selects the data space(s) on a volume from which to suballocate space to the file. All volumes must be of the same type for each component of a file. If more volumes are specified than needed for the primary allocation, the additional volumes can be used when the file is extended. These volumes are described in the file's catalog entry as potential *candidate* volumes.

The length and offset of a cluster's key-field (KEYS) and the average and maximum length of data records (RECORDSIZE) is also specified in Example 11.

Defining a Unique VSAM File

A file can be defined at the same time as the data space(s) which will contain it. In this case, the file is called *unique* and no other file can occupy its data space(s). The data and the index of a key-sequenced unique file will occupy separate data spaces; each requires DLBL and EXTENT statements if the UNIQUE option is specified as part of the cluster definition or both components are defined as unique individually. An entry-sequenced file or relative record file occupy only one data space which also requires a DLBL and an EXTENT statement.

In Example 12, assume that a unique key-sequenced file is defined into the master catalog. (This example would fail if the unique file was to be defined in a recoverable catalog and volume M3330B had no data space defined on it as yet.) Example 12 causes four entries to be created in the master catalog: a volume, cluster, data, and index entry.

```
// JOB  EXAMPLE 12
// DLBL DFILE,,,VSAM
// EXTENT  ,M3330B,,,19,19
// DLBL    XFILE,,,VSAM
// EXTENT  ,M3330B,,,38,19
// EXEC    IDCAMS,SIZE=AUTO
   DEFINE  CLUSTER(NAME(PAYROLL1)-
            RDPW(DEPT27R)-
            VOL(M3330B)-
            RECORDSIZE(100 475)-
            KEYS(12 4))-
          DATA(NAME(PAYROLL1.DATA)-
            UNIQUE-
            CYLINDERS(1 1)-
            FILE(DFILE))-
          INDEX(NAME(PAYROLL1.INDEX)-
            UNIQUE-
            FILE(XFILE)-
            CYLINDERS(1 1))-
          CATALOG(MASTCAT/UPDPW1)
/*
/&
```

For a key-sequenced file with the UNIQUE attribute where the data and index components reside on the same volume, the FILE parameter must be specified under both DATA and INDEX. The VOLUMES parameter and the space allocation parameter (CYLINDERS, BLOCKS, TRACKS, or RECORDS) must be included in the DEFINE command (unless you specified the MODEL parameter).

If you specify the size of the area allocated to a unique index, as is done in this example, you must ensure that it is large enough. For CKD devices, the space allocated to the data in a unique file must be an integral number of cylinders and each extent must begin on cylinder boundaries.

For FBA devices, the space allocated to the data in a unique file must begin on a min-CA (track) boundary and must be a whole number of min-CAs. You use the BLOCKS parameter for space allocation.

A unique file can have a maximum of 16 extents per volume, but it cannot be extended, and space left over after the records are loaded cannot be released.

Defining a Key-Sequenced File

The following example shows the DEFINE command for setting up a key-sequenced cluster that consists of a data component and an index component. The CLUSTER, DATA, and INDEX parameters are all specified, so the data and the index components of the cluster can be explicitly named rather than letting VSAM name them. No records are loaded into the file; defining a file is distinct from loading records into it. See "Loading Records into a File" for an example of how to load a file.

The following example provides space for 10,000 data records (fixed-length, 250 bytes) on each key range. Any future extensions are to be made in increments of space for 500 records. Keys are 15 bytes long and begin in the 31st byte (defined as relative position 30 since byte 1 is relative position 0) of the records. Free space is to be 20% of each control interval and 10% of each control area. Sequence-set index records are to be placed adjacent to control areas in the file and replicated.

Assume that the cluster is to be defined in the master catalog and be in suballocated space on volumes M3330A and M3330B. These volumes need not be mounted, because VSAM can determine whether and what space is available merely by examining the master catalog, which owns these volumes.

(If the master catalog had been defined as recoverable, VSAM would have requested you to mount volumes M3330A and M3330B.)

```
// JOB   EXAMPLE 13
// EXEC  IDCAMS,SIZE=AUTO
   DEFINE  CLUSTER-
             (NAME(MYKSDS)-
             VOLUMES(M3330A M3330B)-
             KEYRANGES((A M) (N Z))-
             RECORDS(10000 500)-
             ORDERED)-
           DATA-
             (NAME(MYDATA)-
             KEYS(15 30)-
             RECORDSIZE(250 250)-
             BUFFERSPACE(8192)-
             FREESPACE(20 10))-
           INDEX-
             (NAME(MYINDEX)-
             IMBED)-
           CATALOG(MASTCAT/UPDPW1)
/*
/&
```

The INDEXED parameter, which is the default and therefore need not be specified, indicates that a key-sequenced cluster is to be defined. The RECORDS parameter indicates primary and secondary allocation quantitites. Specifying the number of records, independent of the number of physical units, such as blocks, tracks or cylinders, leaves the calculation of the number of physical units of space up to VSAM. It calculates the size of the control interval and control area to be used. You may specify the control interval size yourself, and VSAM will use it as long as it falls within the acceptable limits that VSAM calculates.

BUFFERSPACE specifies the smallest amount of virtual storage a processing program will ever provide for I/O buffers to process MYKSDS. A multiple of 512 should, however, be specified to avoid wasting space. If you do not specify BUFFERSPACE, VSAM determines control interval size first and then sets buffer space equal to the size of two data control intervals plus 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, your DEFINE will fail.

The relationship between control interval size and least amount of I/O buffer space is further discussed in "Optimizing VSAM's Performance" in *VSE/VSAM Programmer's Reference.*

With the KEYRANGES parameter, you may assign data to the various volumes of a multivolume key-sequenced file 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. The amount of space specified in the primary allocation parameter is allocated to *each* key range. If the number of volumes is larger than the number of key ranges, the excess volumes will become candidate volumes for all the key ranges. If there are fewer volumes than key ranges, the key ranges collect on the last volume. If the values specified for KEYRANGES subparameters are shorter than those specified for keys, then Access Method Services pads the low key range to the right with zeros and the high key range to the right with

X'FF's. The following example illustrates the use of the VOLUMES, ORDERED, CYLINDERS, and KEYRANGES parameters.

```
VOLUMES (A B C)
KEYRANGES ( (00 30) (31 65) (66 99) )
ORDERED
CYLINDERS (100 10)
```

A primary allocation of 100 cylinders will be made for *each* key range. The first key range will be on volume A, the second on B, and the third on C. If 100 cylinders cannot be allocated on each volume, the request is rejected. A key range can be extended only on the volume it occupies or on a candidate volume. Thus, if volume D were added to the list, all key ranges will be extended on volume D if the appropriate volume initially assigned to the key range is full. If only volumes A and B were specified, the first key range would be allocated on volume A and the second and third key ranges would be allocated on volume B.

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

A primary allocation of 50 cylinders will be made for *each* key range. VSAM will attempt to put one key range on each volume. If volume A does not have 50 cylinders available, the first key range is put on volume B and the second and third on volume C. If neither A nor B has 50 available cylinders, all three key ranges are placed on volume C. A key range will be extended first on the volume it is on, then it will be extended on any candidate volume. A candidate volume is a volume that is named in the volume list for a key-range file but was not initially assigned to a key range, that is, there may be more volumes than key ranges in the list. However, if volume D were available as a candidate volume, each key range would be extended on volume D if no more space were available on the volume of its primary allocation. A key range can cover the volume of primary allocation and any candidate volume.

The ORDERED parameter indicates that space must be suballocated on the volumes in the order in which they are listed in the VOLUMES parameter. In particular, for key-ranged files, ORDERED forces primary allocation for the first key range to be made on the first volume of the volume list, primary allocation for the second key range to be made on the second volume of the volume list, etc. If a volume cannot accommodate the space for the appropriate primary allocation, the DEFINE fails.

Thus, in the first example above, M3330A has 10,000 records allocated for key range A-M, and M3330B has 10,000 records allocated for the key range N-Z. If M3330A does not have enough space to accommodate the 10,000 records in the primary allocation, the DEFINE fails. Contrast this to the UNORDERED case in which both key ranges would be on M3330B provided M3330B has enough space, and M3330A would then become a candidate volume.

Space on each volume could also be specified in the number of records, even with variable-length records: VSAM uses the average size (250 bytes) to calculate the number of cylinders for each volume.

The following examples further illustrate the use of the VOLUMES and ORDERED parameters.

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

The 50 cylinders of primary space for the file must be available on volume A, or the request will be rejected. Volumes B and C are candidate volumes. If the file is extended, a five cylinder secondary space allocation is made on volume A if it has enough data space. Otherwise, a primary allocation of 50 cylinders is made on volume B. If volume B does not have enough data space for a primary allocation, the request for extension is rejected. When the file is subsequently extended, the secondary allocations are made on volume B if it has enough data space. Otherwise, a primary allocation is made on volume C.

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

The 50 cylinder primary allocation for the file can be made on either volume A, B, or C. However, if all 50 cylinders cannot be allocated on one volume, the request is rejected. The volumes are searched in the order they are specified. If both volumes A and B have 50 cylinders available, the allocation will be made on volume A. If the file is extended, the five cylinder secondary allocations are made on the volume the last primary allocation was made on until the volume is full. However, the first allocation to be made on any volume is always a primary allocation of 50 cylinders. If no volume is available on which a primary allocation of 50 cylinders can be made, the DEFINE request is rejected. Once again, the volumes are searched for space in the order specified.

```
VOLUMES (A B C)
ORDERED
CYLINDERS (1000 10)
```

This request will be rejected because the amount of primary space to be allocated on each volume is greater than one volume. The primary allocation *must* be small enough to be satisfied by one volume.

Defining an Entry-Sequenced File

Even though the data component of an entry-sequenced file is the only member of its cluster, you must define a cluster for it. Example 14 shows that records in the file (a cluster named ENTRY) are to be stored on three volumes.

You indicate that the cluster is to be defined in job catalog USER.CAT by specifying a IJSYSUC DLBL statement with file-ID USER.CAT. (Example 15 shows an alternate way of specifying a cluster's catalog). Records are of variable length, up to 700 bytes, with an average size of 500 bytes.

```
// JOB   EXAMPLE 14
// DLBL  IJSYSUC,'USER.CAT',,VSAM
// EXEC  IDCAMS,SIZE=AUTO
   DEFINE  CLUSTER-
             (NAME(ENTRY)-
             VOLUMES(USRVOL U2314B U2314C)-
             ORDERED-
             CYLINDERS(10 2)-
             RECORDSIZE(500 700)-
             BUFFERSPACE(5120)-
             NONINDEXED)-
           DATA-
             (NAME(ENTRY.DATA))
/*
/&
```

The NONINDEXED parameter indicates that an entry-sequenced cluster is to be defined.

The primary allocation of 10 cylinders is only made on volume USRVOL; the additional volumes are used only when the file is extended. These volumes are described in the file's catalog entry as candidate volumes.

Defining a Relative-Record File

Like an entry-sequenced file, the data component of a relative-record file is the only member of its cluster. A relative-record file can be seen as a string of fixed-length slots or record areas, each of which is assigned a relative-record number, starting from 1.

In Example 15, records in the file (in a cluster named STOCKINV) are to be stored on a 2314 volume. The cluster is to be defined in a catalog other than the default catalog, therefore, you must use the CATALOG parameter to indicate the correct catalog (USER.CAT). The records in the file have a fixed length of 132 bytes (the average and maximum record sizes must be equal for a relative-record file). The NUMBERED parameter indicates that a relative-record file is to be defined. The primary number of tracks allocated is 50 and ten secondary tracks are allowed for each extension. Extensions must be suballocated on the volume specified as U2314B, that is, the same volume that contains the primary tracks.

```
// JOB   EXAMPLE 15
// EXEC  IDCAMS,SIZE=AUTO
   DEFINE  CLUSTER( -
             NAME(STOCKINV)-
             VOL(U2314B)-
             TRACKS(50 10)-
             RECORDSIZE(132 132)-
             NUMBERED) -
           CATALOG(USER.CAT)
/*
/&
```

Defining a File in a Recoverable Catalog

When you define a file in a recoverable catalog, the space allocated for the file is recorded in the catalog which owns the volume(s) and in the catalog recovery area of the initial index volume (for a key-sequenced file) or the initial data volume (for an entry-sequenced file or relative-record file). All volumes on the volume list must be mounted since each corresponding volume entry in the catalog recovery area is updated to reflect the fact that the file will take space on this volume. (You do not have to premount the volume(s); VSAM will issue a mount message for them.) You can identify the recovery volumes by looking at the message output at the end of a DEFINE operation. The catalog recovery information in these volumes is updated whenever parallel information in the catalog is modified.

If you want an existing file to be a member of a recoverable catalog, you should first define a recoverable catalog and then export the file from the old unrecoverable catalog and import it to the new one. See "Using EXPORT/IMPORT: Transporting or Backing Up Files" for more information. Alternatively, REPRO may be used to copy the file to a new file defined in a recoverable catalog.

In Example 16, space for file MSTRFILE is to be suballocated from data spaces on volumes U2314B and U2314C.

```
// JOB   EXAMPLE 16
// DLBL  IJSYSUC,'USER.CAT',,VSAM
// EXEC  IDCAMS,SIZE=AUTO
   DEFINE  CLUSTER(NAME(MSTRFILE)-
             VOLUMES(U2314B U2314C)-
             TRACKS(30 10)-
             KEYS (10 1)-
             RECORDSIZE(80 80)-
             ORDERED)
/*
/&
```

Loading Records into a File

After you have defined a file, you can load records into it with a processing program of your own or with the REPRO command. This section discusses only the latter.

The REPRO command causes Access Method Services to retrieve records from a sequential tape or disk file, an indexed-sequential file, or a VSAM file and store them, in VSAM format, in a key-sequenced, entry-sequenced, or relative-record file.

When records are loaded into a key-sequenced file, they must have unlike keys and be in ascending key sequence. Index entries are created and loaded into the file's index as control intervals and control areas are filled up. Free space is left and records are stored on particular volumes according to key ranges, as indicated in the file's definition in the catalog.

You can copy an entire file or only parts of it. You can specify that a certain range of records is to be copied by indicating the locations in the input file where copying is to start and stop. These locations can be indicated in several ways:

- By key for indexed-sequential or key-sequenced files (FROMKEY, TOKEY).
- By RBA for key-sequenced or entry-sequenced files (FROMADDRESS, TOADDRESS).
- By relative-record number for relative-record files (FROMNUMBER, TONUMBER).
- By number of records for any type of file (SKIP, COUNT).

The file into which records are copied may either be empty (that is, newly allocated by way of a DEFINE CLUSTER command) or may already contain records.

See "Reorganizing a File" for information about what happens when records are added to an empty or nonempty key-sequenced, entry-sequenced, relative-record, or sequential file.

You can load all of the records in one job or in several jobs. In subsequent jobs, VSAM continues to store records as before, extending the file as required.

The job control information for loading a file is the same as for other types of file processing. Since your VSAM files are already cataloged when you start a REPRO job, the function of job control here is merely to name the required files and the volumes on which they are stored and to indicate any user

catalog(s), so that VSAM can look at the definitions of the files. Example 17 shows the basic loading function of the REPRO command: taking the records of a sequential file on a disk and storing them in a newly defined VSAM file.

```
// JOB    EXAMPLE 17
// ASSGN  SYS002,234
// DLBL   SAMFILE,'SAM1'
// EXTENT SYS002,231401,1,0,120,2
// DLBL   RRDS,'STOCKINV',,VSAM
// DLBL   IJSYSUC,'USER.CAT',,VSAM
// EXEC   IDCAMS,SIZE=AUTO
   REPRO  INFILE(SAMFILE,ENV(RECFM(F),BLKSZ(100)))-
          OUTFILE(RRDS)
/*
/&
```

Job control statement:

- DLBL SAMFILE indicates the nonVSAM input file.
- DLBL RRDS indicates the VSAM file that is to be loaded with records.
- DLBL IJSYSUC indicates the job catalog in which the VSAM file (STOCKINV) is defined.

The REPRO command copies all the records from the input file SAM1 to the output file STOCKINV.

- The INFILE parameter points to the DLBL statement that identifies the source, or input, file: SAM1.
- The OUTFILE parameter points to the DLBL statement that identifies the file into which the input records are to be copied: STOCKINV.

Alternate Indexes

Apart from the prime index that VSAM automatically creates for every key-sequenced file, you can have VSAM build one or more alternate indexes over a single key-sequenced or entry-sequenced file. Each alternate index accesses the data records of a given file via a different key field (alternate key) within these records.

An alternate index, therfore, provides a unique way to gain access to the same base data, so that when you have several alternate indexes, you can access a file in several different ways without having to keep multiple copies of the same information organized differently for different applications. For example, a payroll file originally indexed on employee number can be indexed on additional fields (called alternate keys) such as employee name, department number, position code, skill code, or social security number.

The data over which the alternate index is built is referred to as the *base cluster*; it can be either a key-sequenced file or an entry-sequenced file, but not a relative-record file or a file which has been defined as reusable.

The alternate index itself is a key-sequenced cluster, namely the alternate-index cluster. It is usually referred to simply as the alternate index. Like an indexed base cluster, it consists of an index component and a data component. The index component of an alternate index is identical in structure, format, and function to the index component of a base cluster, that is, the prime index. The data component of an alternate index has a fixed format and serves to establish the relationship between the alternate index and the base cluster.

When building an alternate index, you can use as the alternate key any field in the base cluster's records which has a fixed length and a fixed position

within each record. The alternate key must be in the first segment of a spanned record. For each alternate key value, the data component of the alternate index contains a unique record. This record consists of the alternate key itself, followed by a pointer that is the prime key (for a key-sequenced base) or RBA (for an entry-sequenced base) of the base-cluster record that contains the alternate key. If more than one base-cluster record contains the same alternate key, then the alternate index record contains a pointer to each base-cluster record. These duplicate, or non-unique keys are discussed under "Alternate Keys."

Alternate-Index Path

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 you define a path (through Access Method Services), 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. A path requires a name of its own which always refers to the alternate index and the related base cluster as a pair. To use the alternate sequence to process records in the base cluster, you specify the PATH's CLUSTER name in the 44-byte file-ID of the DLBL card used to open the file.

When a path is referenced, both its alternate index and its base cluster are opened. If you want to process only the alternate index, without its base cluster, you can use the name of the alternate index instead of the path name and then process the alternate index like any other file.

Alternate-Index Record

Each record in the data component of an alternate-index cluster is a variable-length logical record that contains (1) system header information, (2) the alternate key, and (3) at least one pointer to the base cluster.

System Header Information: The system header information has a fixed length and indicates:

- Whether the alternate-index record contains one or more pointers in the form of prime keys (for key-sequenced files) or in the form of RBAs (for entry-sequenced files)
- The length of a pointer (all pointers have the same length for a given alternate index)
- The length of the alternate key
- The number of pointers.

Alternate Keys: Any field in the base cluster's records that has a fixed length and a fixed position within a record can be used as an alternate key when building an alternate index over the base cluster. If the base records span control intervals, the alternate key must be in the first control interval of the spanned record. If you build several alternate indexes over a base cluster, the alternate key fields of the different alternate indexes may overlap each other in the base-cluster records; they can also overlap the prime key.

In contrast to the prime key, a given alternate key may occur in more than one record in the base cluster. For example, if an alternate index is established by department number over a payroll file organized by employee number, all the employees with the same department number will be grouped together. This means that there will be several prime-key pointers (employee numbers) in each alternate-index record, one for each occurrence of the alternate key (department number) in the base cluster. When a given alter-

nate key occurs in more than one base record, it is said to be non-unique. If it occurs in only one base record (for example, social security number), it is called unique. You must indicate whether an alternate index will contain unique or non-unique alternate keys when you define the alternate index.

Alternate-Index Pointers: The relationship between the alternate index and its base cluster is established by two different types of pointers in the alternate-index record, depending on the type of base cluster. If the base cluster is a key-sequenced file, the pointer in the alternate-index record is the prime key of the base-cluster record in which the alternate key occurs. (A prime key pointer has the same length as the prime key field of the base cluster it points to.) If the base cluster is an entry-sequenced file, the pointer is the RBA of the base-cluster record in which the alternate key occurs. (An RBA pointer is always four bytes long.) There is only one prime key or RBA pointer allowed, unless the alternate index has the NONUNIQUEKEY attribute.

For non-unique keys, pointers are associated with a given alternate key value. The pointers are ordered by their arrival time; that is, if a record in the base cluster is updated with a key change, or if a new record is inserted with the same alternate key value, VSAM adds the new prime key pointer to the end of the alternate-index record. (In the case of a key change, VSAM deletes the old pointer.)

The fact that the pointers are ordered by arrival time implies that immediately after an alternate index has been built (and as long as it remains unchanged), its pointers are in prime-key order. The maximum number of pointers that can be associated with a given alternate key is 32,767, provided the maximum possible record length for spanned records is not exceeded.

Creating an Alternate Index

An alternate index can only be built over a non-empty VSAM base cluster. The logical steps involved in creating an alternate index are:

1. Define the alternate index and relate it to a base cluster by means of the DEFINE ALTERNATEINDEX command.
2. Build the alternate index either yourself or by means of the BLDINDEX command. The Access Method Services routines then perform the following operations:
   a. Extract the alternate key and the prime key or RBA from each record of the base cluster during a sequential scan through the base cluster.
   b. Order the extracted alternate keys, together with the associated pointers, in the collating sequence of the alternate key.
   c. Build the alternate-index records from the ordered key and associated pointers. (Records with the same alternate key are merged into a single alternate-index record.)
   d. Build the alternate index from the individual alternate-index records as a key-sequenced file.

Defining an Alternate Index

When a DEFINE command is used to define an alternate index, VSAM creates three entries in the catalog: an entry for the alternate index, its data component, and its index component.

Attributes of the data and index components can be specified separately from the attributes of the alternate index as a whole. If 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) apply to its components. If an attribute that is applicable to the data or index component is specified for both the alternate index and the component, the component specification overrides the alternate index specification.

You can also specify that the data and index components of an alternate index are to reside on different volumes; you do this by specifying the VOLUMES parameter as an attribute of both DATA and INDEX.

See the DEFINE ALTERNATEINDEX command in "Access Method Services Commands" to identify which parameters can be used with each type of entry.

The following examples are intended to demonstrate how to use the basic DEFINE parameters to define an alternate index without considering all of the options. See also Example 8 in "Appendix A: Sample Job Streams."

No DLBL and EXTENT statements are required if space for the alternate index is suballocated from one or more existing data spaces. VSAM selects which data spaces or portions of data space(s) on a volume to suballocate to the alternate index. You only indicate the volume(s) on which the alternate index is to be allocated and the amount of space to be allocated to it.

```
// JOB   EXAMPLE 18
// EXEC  IDCAMS,SIZE=AUTO
   DEFINE  ALTERNATEINDEX(NAME(DEPTIND)-
             RELATE(PAYROLL1/DEPT27R)-
             VOLUMES(M3330A)-
             TRACKS(100)-
             RECORDSIZE(100 200)-
             KEYS(12 1))
           CATALOG(MASTCAT/UPDPW1)
/*
/&
```

In Example 18, the alternate index named DEPTIND is suballocated 100 tracks from data space(s) on volume M3330A. The alternate index is related to a base cluster named PAYROLL1. The alternate key starts at position 1 (relative to 0) in the data records of the base cluster and has a length of 12 bytes.

Example 19 defines an alternate index over a base cluster. The alternate index is defined as unique, so you must provide EXTENT information together with the definition of the alternate index. The FILE parameters link the file information, which is entered into the catalog, with the space inform-

ation contained in the job control statements. By default, the alternate index is not reusable, has a key length of 64 beginning at byte 0, an average record size of 4086 bytes, and a maximum record size of 32,600 bytes.

```
// JOB     EXAMPLE 19
// DLBL    IJSYSUC,'USER.CAT',,VSAM
// DLBL    DFILE,,,VSAM
// EXTENT  ,U2314C,,,240,20
// DLBL    XFILE,,,VSAM
// EXTENT  ,U2314C,,,260,20
// EXEC    IDCAMS,SIZE=AUTO
   DEFINE  AIX-
              (NAME(AIX.PAYROLL2.NAMES)-
              RELATE(PAYROLL2)-
              RDPW(AIXRDPW)-
              VOL(U2314C)-
              UNIQUE)-
            DATA (NAME(AIX.PAYROLL2.NAMES.DATA)-
              CYLINDERS(1 1)-
              EXCEPTIONEXIT(ERREXIT1)-
              FILE(DFILE))-
            INDEX (NAME(AIX.PAYROLL2.NAMES.INDEX)-
              CYLINDERS(1 1)-
                FILE(XFILE))
/*
/&
```

Building an Alternate Index

After you have defined an alternate index, you must build it from records in the base cluster (the base cluster cannot be empty) by using the BLDINDEX command. In order to build an alternate index, BLDINDEX reads each base cluster record and forms a sort record consisting of the alternate key together with the prime key (for a key-sequenced file) or the RBA (for an entry-sequenced file). These records are then sorted into alternate index key sequence. If the caller of BLDINDEX has provided enough virtual storage, these records are sorted internally.

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

1. Sort record length = alternate key length + prime key length (for a key-sequenced file) or 4 (for an entry-sequenced file).
2. Sort record length * number of records in the base cluster.
3. Record sort area size = result of 2 above, rounded up to next multiple of 2048, or 32,768, whichever is greater.
4. Sort table size = (record sort area size divided by sort record length) * 4.

The sum of 3 and 4 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.

You can specify that BLDINDEX is to perform an external sort (in case insufficient virtual storage exists for an internal sort) by specifying the EXTERNALSORT parameter. If BLDINDEX is unable to obtain enough virtual storage to perform an internal sort and you have specified the EXTERNALSORT parameter, BLDINDEX dynamically defines two VSAM entry-sequenced files and uses them as the work files for an external sort. The sort work files will be deleted by BLDINDEX at the end of the sort. The minimum amount of virtual storage required under these circumstances is calculated as follows:

32,768 + (32,768 divided by sort record length) * 4

The amount of space that Access Method Services requests when defining each sort work file is calculated as follows:

1. Sort records per block = 2041 ÷ sort record length
2. Primary space allocation in records = (number of records in base cluster ÷ sort records per block) + 10
3. Secondary space allocation in records = (primary space allocation × .10) + 10

Both primary and secondary space allocations are requested in records with a fixed-length record size of 2041 bytes and a control interval size of 2048 bytes.

After the records have been sorted into alternate key sequence, BLDINDEX uses them to form the alternate index records. Each sort record is used to create one alternate index record unless the NONUNIQUEKEY attribute has been specified in the definition of the alternate index. In the case of UNIQUEKEY, each alternate index record contains the alternate index key and the associated base cluster prime key (key-sequenced file) or RBA (entry-sequenced file). In the case of NONUNIQUEKEY, each alternate index record contains one alternate key and all associated base cluster prime keys or RBAs representing records containing that same alternate key. If the alternate index has been defined with the REUSE attribute, BLDINDEX automatically writes the new alternate index records starting from the beginning of the file and overriding any records previously stored.

The parameters of the BLDINDEX command are used to identify the object over which the new alternate index is to be built (INDATASET), the alternate index itself (OUTDATASET), the sort work volumes (WORKVOLUMES), and the name of the catalog in which the sort work files are to be defined if they are required (CATALOG), and to specify whether an external sort is to be performed if insufficient storage exists for an internal sort (EXTERNALSORT/INTERNALSORT).

```
// JOB   EXAMPLE 20
// EXEC  IDCAMS,SIZE=AUTO
   BLDINDEX  INDATASET(EXAMPLE.KSDS2) -
             OUTDATASET(EXAMPLE.AIX/AIXUPPW)-
             CATALOG(AMASTCAT/MCATMRPW)
/*
/&
```

Defining a Path

The DEFINE PATH command is used to establish the relationship, the *path,* between an alternate index and its base cluster. A path does not occupy any data space; it is a catalog entry only. 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 establish a path directly over a base cluster, without an intermediary alternate index and with its own protection attributes. A path so defined provides access for a file under another name. You can specify NOUPDATE access for the base cluster, which bypasses allocation of the base cluster's upgrade set and thus does not cause upgrading.

Specifying Information That Defines a Path

The examples given here are intended to demonstrate how to use the basic DEFINE parameters to define a path without considering all of the options. See also Example 8 in "Appendix A: Sample Job Streams."

In Example 21, a path named PATH.PAYROLL1.NAMES is defined over an alternate index named AIX.PAYROLL1.NAMES and its base cluster (named in the RELATE parameter when the alternate index was defined).

```
// JOB   EXAMPLE 21
// EXEC  IDCAMS,SIZE=AUTO
   DEFINE  PATH-
             (NAME(PATH.PAYROLL1.NAMES)-
              PATHENTRY(AIX.PAYROLL1.NAMES)-
              MRPW(MASTER))-
            CATALOG(MASTCAT/MRCATPW1)
/*
/&
```

Accessing a Base Cluster via a Path

A path is a means for accessing a base cluster either via an alternate index or, in the case of a base-cluster-only path, via another name. Like an alternate index or cluster, a path always requires a name of its own, which you specify in the DEFINE PATH command of Access Method Services. This name is also used as the "file-ID" operand in the DLBL statement, which you use when you want to open the path. Opening a path results in associating the base cluster with the alternate index (for an alternate index path).

You may issue the same requests for an alternate-index path (GET, PUT, etc.) that you can issue against a VSAM file. However, all access to the base cluster via the alternate index by way of a path must be by key; addressed (that is, processing by RBA) and control interval processing are not permitted. All key references are to the alternate-key sequence. If an alternate key occurs in more than one base cluster record, the base records are returned in the order in which they are stored in the file.

Alternate-Index Upgrade

All changes in the base cluster that affect the contents of its alternate index or indexes should be reflected in the base cluster's alternate index(es), so that an alternate index is always 'synchronized' with its base cluster. This updating activity is referred to as alternate-index upgrade.

You may have VSAM upgrade an alternate index or you may upgrade it yourself.

To have VSAM upgrade an alternate index, specify the UPGRADE attribute when you define the alternate index. As a result, that alternate index becomes a member of the upgrade set of the associated base cluster. Whenever you open the base cluster for any type of update processing other than control-interval access, VSAM opens all of the alternate indexes in the upgrade set and updates them, if necessary.

VSAM updates the affected alternate index(es) in the upgrade set of the base cluster whenever a base record is inserted or erased, or an alternate key field is changed. This updating activity is part of the request, and VSAM completes it before returning control to your program.

If the updating of an alternate index fails because of a logical error (such as a duplicate key condition for a UNIQUEKEY alternate index), the request which caused the update operation is rejected and the base cluster, together

with the alternate indexes of its upgrade set, are restored to the status existing before the request was issued (except for the sequence of the alternate index pointers). If a physical error condition occurs, VSAM terminates the upgrading of the alternate index(es) immediately and enters the EXCEPTIONEXIT and/or SYNAD exit.

If you specify NOUPGRADE in the DEFINE ALTERNATEINDEX command or if the base cluster is modified through control-interval access, you are responsible for upgrading the alternate index yourself because VSAM considers alternate indexes to be synchronized with the base cluster at all times. When you open a base cluster, those of its alternate indexes which have the NOUPGRADE attribute will not be updated by VSAM when you insert a record into, or erase a record from the base cluster, or change an alternate key field.

Defining a NonVSAM File

You use the DEFINE NONVSAM command to catalog any existing nonVSAM file into a nonrecoverable VSAM catalog. An entry is created in a master or user catalog but no space is allocated or reserved as a result of a DEFINE NONVSAM command.

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

Example 22 shows how to define an existing nonVSAM file entry in the master catalog:

```
// JOB   EXAMPLE 22
// EXEC  IDCAMS,SIZE=AUTO
   DEFINE  NONVSAM-
             (NAME(STOCKINV)-
             VOLUMES(M3330B)-
             DEVICETYPES(3330))-
             CATALOG(MASTCAT/UPDPW1)
/*
/&
```

See also Example 4 in "Appendix A: Sample Job Streams."

Using ALTER: Altering Catalog Entries

Many of the attributes that you define when you create a catalog entry may be modified subsequently by the ALTER command.

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

Altering an object's entry doesn't normally require that the object's volume be mounted, because the object's use of space and the availability of space in the volume's data spaces can be determined by examining the catalog. The object'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. When the object is defined in a recoverable catalog, then the recovery volume must also be mounted (the recovery volume's volume serial is identified at the end of the DEFINE operation).

Specifying Information that Alters an Entry

In Example 23 the control interval and control area free space percentages of the data component of the file PAYROLL1 are altered. The ALTER of PAYROLL1.DATA shows a means of optimizing space usage for a file. The data component was originally defined with 40 percent free space. After initial loading this percentage is reduced, since further activity against this file will not be of the mass insert type.

```
// JOB   EXAMPLE 23
// EXEC  IDCAMS,SIZE=AUTO
   ALTER  PAYROLL1.DATA-
            FREESPACE(10 10)-
          CATALOG(MASTCAT/MRCATPW1)
/*
/&
```

In Example 24 the ALTER of ACCOUNTS shows a way to change the security scheme of a file. Establishing a security scheme for an existing unprotected file would be done in the same manner.

```
// JOB   EXAMPLE 24
// EXEC  IDCAMS,SIZE=AUTO
   ALTER  ACCOUNTS/DEPT27MR-
            MRPW(DEPT26M)-
            CTLPW(DEPT26C)-
            UPDPW(DEPT26U)-
            RDPW(DEPT26R)-
            AUTH(D26AUTH)
/*
/&
```

See also Example 7 in "Appendix A: Sample Job Streams."

Using DELETE: Deleting Catalog Entries

You use the DELETE command to delete any previously defined VSAM object (data space, cluster, alternate index, path, catalog) and remove its catalog entry. In addition, you can remove the entry for a nonVSAM file from a nonrecoverable catalog and optionally scratch that file from the volumes containing it.

All objects deleted by a single DELETE command must be defined in the same catalog.

Deleting a suballocated object normally doesn't require that its volume(s) be mounted, because its space allocation can be determined by examining the catalog. (If the catalog is recoverable, the recovery volume needs to be mounted.)

If an object is deleted and the ERASE parameter was specified, then the space is not only freed for use by new objects but also overwritten with binary zeros. You can delete all alternate indexes and paths related to a base cluster by deleting the base cluster alone; however, even if the ERASE parameter not only is specified for the base cluster but was also specified during definition for all other objects connected with the base cluster, only the space freed by the base cluster will be overwritten with binary zeros. To cause any related alternate indexes to be overwritten with binary zeros, you must delete them individually, before you delete the base cluster. When you delete entries from a user catalog, you may identify the catalog either with the CATALOG parameter or (for the job catalog) the IJSYSUC DLBL job control statement. If the entries are to be erased, the catalog containing them is the catalog specified in the CATALOG *catname* parameter or the default catalog.

When a unique file or unique alternate index is deleted,the cluster or alternate index entry and the data and index entries are deleted from the catalog and the data space they occupied is also deleted; however, the volume entries will not be automatically deleted from the catalog. To do so, you must specify SPACE in a separate DELETE command and all the data spaces on the volume must be empty.

A DELETE SPACE will cause all empty data spaces of the specified volume to be deleted rather than an individual data space. However, if FORCE is also specified, nonempty data spaces are deleted as well.

When a nonVSAM entry is deleted, the file entry is scratched from the VTOC of the pack on which the file resides unless the user specifies the NOSCRATCH parameter.

Example 25 deletes all of the objects defined in the user catalog and then deletes the catalog itself. The catalog cannot be deleted as long as it contains any entries (besides its own and the entry for its own data space).

```
// JOB   EXAMPLE 25
// DLBL  IJSYSUC,'USER.CAT',,VSAM
// EXEC  IDCAMS,SIZE=AUTO
   DELETE  PATH.PAYROLL2.NAMES-
           PATH
   DELETE  AIX.PAYROLL2.NAMES-
           ERASE-
           AIX
   DELETE  PAYROLL2-
           PURGE-
           CLUSTER
   DELETE  (U2314B)-
           SPACE
   DELETE  USER.CAT-
           USERCATALOG
/*
/&
```

Example 26 shows the deletion of a base cluster, alternate index, and path. The alternate index and path are deleted implicitly as a result of deleting the base cluster.

```
// JOB   EXAMPLE 26
// EXEC  IDCAMS,SIZE=AUTO
   DELETE  PAYROLL-
           PURGE-
           CLUSTER-
           CATALOG(MASTCAT/MRCATPW1)
/*
/&
```

Example 27 shows the deletion, with the ERASE option, of a unique cluster.

```
// JOB   EXAMPLE 27
// EXEC  IDCAMS,SIZE=AUTO
   DELETE  ACCOUNTS-
           ERASE-
           PURGE-
           CATALOG(MASTCAT/MRCATPW1)
/*
/&
```

See also Examples 22 and 23 in "Appendix A: Sample Job Streams."

Using REPRO: For Catalog Backup and File Reorganization

Backing Up a Catalog

You can use the REPRO command to unload (create a backup copy of) a catalog. (The backup copy cannot be used as a catalog in the unloaded form.) If the catalog becomes inaccessible, you can use the REPRO command to replace (reload) it with the backup copy. To continually have a recent backup copy available, you should unload a catalog periodically.

Unloading a Catalog

You can use the REPRO command to unload a catalog to a sequential (SAM) file, or to a key-sequenced or entry-sequenced file. (You must unload a master catalog to a sequential file, or else the subsequent reload will not work.) Do not allow the catalog to be updated during the unload operation.

If you use a key-sequenced or entry sequenced file to provide backup for a user catalog, define the file that contains the unloaded catalog in *another* catalog; this allows you to recover the unloaded catalog if the original (copied) catalog is destroyed.

Use LISTCAT before unloading a catalog so that you can compare the listing of the unloaded catalog with the listing you obtain after reloading.

Reloading a Catalog

You can use REPRO to reload the backup copy of a catalog into a catalog (the "target") with the same name, volume serial number, and device type as the original catalog.

The target catalog can be either an earlier or later version of the original catalog, or a newly-defined user catalog (without any entries other than the basic catalog information). The newly-defined catalog must be able to hold at least as many entries as the original catalog could hold at the time the backup copy was made.

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) from the backup.
- Inserts, into the target catalog, entries that exist only in the backup.
- Deletes, from the target catalog, entries that exist only in the target catalog.

During reloading, up to 100 messages may be issued to indicate entries that exist only in the target catalog 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-describing information. (The assumption is that, if a catalog is new, no other VSAM data space exists on the volume under normal conditions.) If the catalog is new, the reload operation bypasses the reload of the original version of the Volume record; all the data space information previously in the Volume record at the time the catalog was unloaded is lost.

After the reload of a newly-defined catalog, the entries in the reloaded catalog may reference files previously existing on the volume.

If these files 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 the needed information to the volume record by, first, using EXPORT PERMA-

NENT to remove the file entries from the new catalog, then defining a data space large enough to accommodate the files, and then using IMPORT to put the files into the newly defined data space.

The space allocation used for the reloaded catalog is the space allocation of the target catalog whether it is a newly defined catalog or an earlier or later version of the unloaded catalog. In both cases the catalog allocation properly reflects the VSAM data space for the catalog on the volume.

A DLBL job control statement (IJSYSUC) should be used to identify the catalog to be unloaded or reloaded. This ensures that the catalog can be opened as a catalog prior to an unload or a reload operation. The define of a new catalog and the listing of a catalog should be done in separate steps from any reload operations.

Password protection of VSAM catalogs is optional. Passwords can be established at catalog definition or subsequently added. When a catalog is password protected, the master password must be provided to permit catalog unloading and catalog reloading. The password can be supplied in the REPRO command or through the system console in response to password prompting.

The following restrictions should be observed when a catalog backup operation is to be done:

- Unload the master catalog only to a sequential file.
- The unloaded version of a catalog must be reloaded into a catalog with the same name, volume serial, and device type.
- The unloaded version of a catalog must be reloaded into a catalog with the same entry capacity (i.e., number of entries cataloged). Some high key range compression may occur when reloading into a newly defined catalog; consequently, the reloaded version may require less high key range space than a multiple extent source catalog. The target catalog will not be extended by means of secondary allocations during the reload operation.
- 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.
- The catalog contents after reloading may be out of synchronization with the actual data on the volumes indicated as owned in the reloaded catalog. After reloading a recoverable catalog, the LISTCRA command with COMPARE option should be run in a separate job step immediately after the reload 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 "LISTCRA Mismatch Messages" in *VSE/VSAM Programmer's Reference.* No other jobs should be run after the reload and before the LISTCRA which access any of the files cataloged in the reloaded catalog.
- When reloading or restoring a nonrecoverable catalog, a LISTCAT command should be executed with each unload and after each reload operation (in a separate step after a reload). The LISTCAT output obtained in conjunction with unload may be necessary for comparison to the LISTCAT output obtained after reload. Both outputs may be necessary to determine the manual intervention required when mismatches are detected during reload.
- Files or volumes (in the original catalog) that are deleted and redefined or permanently exported after an unload operation will not be flagged as

changed upon reload because detection of this type of change is not possible. The deleted files or data spaces will still be defined in the restored catalog. (Entries that exist only in the backup copy are inserted into the target 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 file or data spaces have been defined or imported since the last catalog backup and the catalog is reloaded or restored, then the defined files or data spaces will not be defined in the reloaded or restored catalog. Processing these files or data spaces by means of the restored catalog is not possible since they cannot be accessed. The space formerly occupied by these VSAM files or data spaces will not be usable, but may be recovered by scratching the format-1 labels 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 IKQVDU to give up volume ownership.
- If a VSAM file 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 file has been extended within space already allocated to the file before the backup but has acquired no new extents, then you can issue the VERIFY command to update the catalog pointers, and the file may be accessed normally.
- The data in any extents that have been acquired by the file since the catalog was backed up is unrecoverable. For an entry-sequenced file the data in any new extents should consist only of records that have been added to the end of the file. Therefore, it is possible to recover all of the data in the old extents by accessing the file sequentially up to the end of the old physical space allocation. For a key-sequenced file, the new extents may be any portion of the file 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 file by means of address sequence, but you then have the problem of identifying the missing records. Individual file recovery for those files affected will be necessary.

  See the section "VSAM Recovery Techniques" in *VSE/VSAM Programmer's Reference* 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.

Reorganizing a File

You use the REPRO command to reorganize an old file by copying it into a newly-defined file of the same type. With key-sequenced files, for example, you can specify different percentages of distributed free space and different performance options for the new file when you define it.

Because data is copied as single logical records, automatic reorganization takes place when you copy a key-sequenced file into another key-sequenced file. Reorganization includes:

- Relocation of records so that their entry sequence matches their key sequence.
- Redistribution of the free space throughout the file.
- Reconstruction of the prime index.

Fixed-length, unblocked records in an indexed-sequential file are preceded by the key string when they are written into a VSAM file. Therefore, the length of the records in the output file must include the length of the key string.

The new (output) file into which records of the old (input) file are copied may either be empty or already contain records. Figure 2-1 shows what happens when records from the input file are added to an empty or nonempty file.

Input File	Empty Output File			Nonempty Output File		
	ES/S	KS	RR	ES/S	KS	RR
ES/S	1	2	5	3	4	7
KS	1	2	5	3	4	7
RR	1	2	6	3	4	8

where

ES/S = Entry-sequenced or sequential

KS = Key-sequenced

RR = Relative-record

1 = Copies in sequential order.

2 = Copies in collating sequence by defined key field and builds an index. The source records must be in key sequence.

3 = Adds records in sequential order to the end of file (VSAM ES); copies records in sequential order from the beginning of the file overlaying previous data (SAM sequential).*

4 = Merges records in collating sequence by the defined key field and updates the index. A record whose key duplicates a key in the output file is lost, unless you specify that the new record is to replace the old one.

5 = Copies records sequentially into consecutive slots, beginning with 1.

6 = Copies records in the same position they had in the input file.

7 = Will not copy a record unless the file has the REUSE attribute and the REPRO REUSE option is specified.

8 = Same as 6 except that a record in the input file that has the same number as one in the output file is lost, unless you specify that the new record is to replace the old one.

* This action will occur only if the operator responds with "DELETE" when the DOS/VSE message "OVERLAP ON UNEXPRD FILE" or "DUPLICATE FILE-ID" is encountered.

Figure 2-1. Reorganizing a File

If you use REPRO to locate errors in a file, Access Method Services terminates the copy operation after four recoverable errors have been encountered while trying to read the file. The types of errors classified as "recoverable" are duplicate keys, wrong length records, records out of sequence, and I/O errors in the data component of a VSAM file.

You can save time in reorganizing VSAM files (using the REPRO command) by specifying the optional BUFSP parameter in the DLBL statement. Ordinarily, REPRO uses the ACB macro default values of two data control-interval buffers and, for indexed files, one index control-interval buffer. By specifying an appropriate value in the BUFSP=parameter, you can override the ACB default; you can allocate additional data control-interval buffers for each of your VSAM files (if you have available virtual storage). For example, if your input file is key-sequenced with a 512 byte index control interval and a 2048 byte data control interval, you can specify BUFSP=10752 ((5 × 2048) + 512 = 10752) and cause five data control-interval buffers (and one index control-interval buffer) to be allocated when the file is opened.

For other uses of REPRO, see "Loading Records into a File" in "Using DEFINE: Defining Objects in a Catalog." See also Examples 5 and 6 in "Appendix A: Sample Job Streams."

Using EXPORT/IMPORT: Transporting or Backing Up Files

You use the EXPORT and IMPORT commands to transport VSAM files and user catalogs between VSE systems (or set of systems in a DASD sharing environment) or between DOS/VS, VSE, and OS/VS systems. The EXPORT command extracts catalog information and produces a portable copy of the file that is to be moved. The IMPORT command loads a portable file and its catalog information in the receiving system. Figure 2-2 compares volume and file portability. File portability is achieved by moving volumes or by moving individual files.

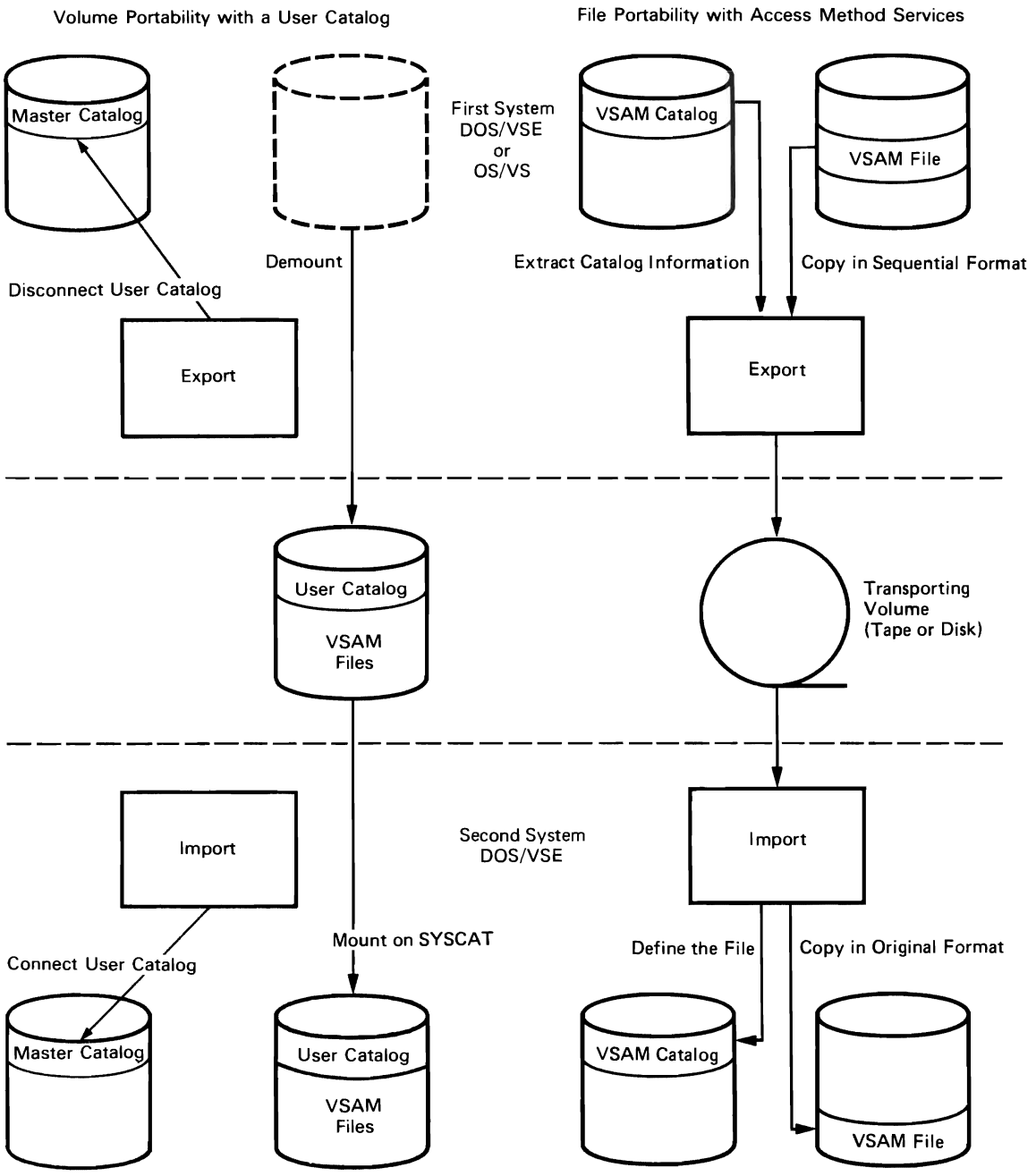


Figure 2-2. Data Portability (Achieved by Moving Volumes or by Moving Individual Files).

EXPORT and IMPORT also enable you to create a backup copy of a file and its catalog entries and reload them into the same system when they are needed. When you import a backup copy, the catalog entries are regenerated.

See "Multifile Volume Considerations" in *VSE/VSAM Programmer's Reference* for information on how to use the EXPORT/IMPORT commands to create backup copies of several VSAM files on a multifile tape volume (or volume set).

In addition, EXPORT and IMPORT enable you to transport a file defined in an unrecoverable catalog to a recoverable catalog for the purpose of making its data and catalog entries recoverable in the event of catalog failure. To do this, you define a recoverable catalog and then export the file from the old catalog and import it to the new one. Alternatively, you may use REPRO to copy the file to a file defined in a recoverable catalog. Any object marked not usable by catalog recovery is not exportable.

Master and user catalogs cannot be copied using EXPORT and IMPORT; a user catalog can, however, be imported to another system by physically transporting the volume on which it is stored. When a user catalog is to be transported, it is not copied; the user catalog remains on its original volume in its original form. When it is exported by an EXPORT DISCONNECT command, the master catalog's pointer to it is removed. When it is subsequently imported to a new system by an IMPORT CONNECT command, a pointer to it is created in the new system's master catalog. (VSE DASD Sharing supports shared master catalogs.) You don't have to issue the EXPORT DISCONNECT command to move a user catalog to another system, but must use an IMPORT CONNECT command to build a pointer to the user catalog in the new system's master catalog. If you didn't issue the EXPORT DISCONNECT command for it, then it belongs to both systems. If both systems support VSE DASD Sharing and the user catalog is on a shared volume, it can be shared. Otherwise, the user catalog can be available to only one system at a time.

When a file is exported, relevant portions of its catalog entries are copied to a movable volume along with the cluster's component(s). The catalog name cannot be moved with the file. The portable copy is a variable-blocked, spanned, sequential file; it may be stored on tape or disk. You can control the block size of the portable file. If the file is exported to tape, if it is to be other than the first file on the tape and if the tape has just been mounted, you must issue the MTC command to correctly position the tape.

Exportation of a file is either permanent or temporary. In permanent exportation, the catalog entries are deleted and storage space is freed in the original system. In temporary exportation, the sending system retains a copy of the file, but the copy is marked to indicate that there is a copy elsewhere.

When a base cluster and its alternate index(es) are permanently exported, the alternate index(es) must be permanently exported before the base cluster. Otherwise, the alternate index(es) will be deleted when the base cluster is deleted and cannot be exported. EXPORT will automatically export paths related to the cluster or alternate index being exported. When an object is exported, the statistics kept in its catalog entry are not extracted for exportation and are not available when the object is subsequently imported.

When a base cluster and its alternate index(es) are imported, the base cluster must be imported before the alternate index(es). If you are using a tape as the transporting volume, you must first forward space the tape to the base cluster and read it, then rewind to the load point and read the alternate indexes sequentially. See "Multifile Volume Considerations" in *VSE/VSAM*

Programmer's Reference for more information. A path cannot be transported as an object from one system to another, but is automatically imported when its related base cluster and alternate index are imported.

You may alter some of a file's attributes when you import it, but space allocation problems may occur if a VSAM file is imported onto a device of a type different from the type it was exported from. The space allocation quantities are recorded in terms of tracks (min-CAs) (even if you specified CYLINDERS, BLOCKS, or RECORDS in the DEFINE command) in a file's catalog entry. When the file is imported the number of min-CAs in the catalog entry is not changed to reflect the characteristics of a new device type unless you import the file into a predefined file with adjusted allocation quantities. Thus, an attempt to export a VSAM file from a 3330 and import it onto a 2314 may fail because the allocation quantities in the catalog entry specify less space on a 2314 than they did on a 3330. (If the secondary space allocation quantity is not zero, VSAM may be able to allocate enough secondary space on the 2314 to contain the file.) Conversely, if a file is exported from a 2314 and imported onto a 3330, it may be allocated more space than it needs.

You may use EXPORT and IMPORT to redefine a file for a different device with space parameters that are appropriate to the new device. You may also change protection attributes. To change the file's device type, you permanently export it (or temporarily export and delete it), redefine it (by way of DEFINE) with new allocation attributes, then import the transporting copy into the newly defined file. (If you use DEFINE before deleting the exported file, you can use the exported file's entry as a model assuming you give it a new name.) The newly defined file must meet the following conditions:

- It must be empty.
- It must be the same type of file (INDEXED, NONINDEXED, or NUMBERED) as the file being imported.
- If INDEXED, it must have the same key length and key position as the file being imported.
- It must have a maximum logical record length greater than or equal to that of the exported file.

To change devices without allocation problems when moving clusters or alternate indexes between systems or between catalogs, you may use two techniques.

The first technique is:

1. Use the EXPORT command to create the copy of the cluster or alternate index to be exported.
2. Use the DEFINE command to define a new entry for the cluster or alternate index in the catalog to which the cluster or alternate index is to be imported. Specify all the parameters used when the cluster or alternate index was originally defined, using the MODEL parameter if a catalog that contains a suitable model entry is available. 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 this file, you must delete that entry or use a different name in the DEFINE command and specify the new name in the OBJECTS parameter of IMPORT.
3. Use the IMPORT command to read the cluster or alternate index into the predefined empty cluster or alternate index. If the empty cluster or alter-

nate index was defined with a name different from that of the cluster or alternate index exported, the NEWNAME subparameter of the OBJECTS parameter must be used to rename the exported cluster or alternate index file-ID.

The second technique is:

1. Use the REPRO command to create the copy of the cluster or alternate index to be exported.
2. Follow step 2 above except do not specify a new name by means of IMPORT.
3. Load the cluster or alternate index into the predefined cluster or alternate index by issuing a REPRO command to copy it.

EXPORT: Making a File Portable

The EXPORT command enables you to copy a cluster or an alternate index in sequential form onto a storage volume to be transported to another system. The transporting volume may be magnetic tape or disk. In addition, the EXPORT command extracts information from the catalog entry that defines the object to be transported and copies the information onto the transporting volume. The information is used to define the file automatically in a VSAM catalog in the receiving system.

VSAM always defaults to an optimum buffer space value when using the EXPORT command; it ignores the buffer space value in the catalog entry and in the DLBL BUFSP statement (if specified).

See Examples 10, 11, and 12 in "Appendix A: Sample Job Streams."

IMPORT: Loading a Portable File

The IMPORT command enables you to use the catalog information extracted by EXPORT to automatically define a new cluster or alternate index in the catalog that you specify and to suballocate space for it. The object itself is stored, in its VSAM format, in the space allocated for it. Alternately, you can use DEFINE to define a new cluster or alternate index and allocate space for the object and use IMPORT to copy the object from the transporting volume into the space allocated for it, that is, import the object into an empty cluster.

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

A cluster, alternate index, or user catalog cannot be transported to a system if its name or the name of any of its components already exists in the receiving catalog. The only exceptions are when the cluster or alternate index name that already exists in the receiving catalog belongs to an object that has been temporarily exported (that is, TEMPORARY was coded when it was exported), or belongs to the predefined (empty) object that is to receive the imported object. The entry of a temporarily exported object is deleted when an object with the same name is imported; a new entry is then built for the imported object. When a base cluster and its alternate indexes are imported, the base cluster must be imported first, followed by the alternate indexes in any order. This is necessary because a base cluster must exist before an alternate index can be defined over it.

You may specify any block size that you choose for the portable file at the time of its creation by EXPORT by using the BLOCKSIZE subparameter of the ENVIRONMENT parameter. The default value is 2048 bytes per block.

When an IMPORT is issued, you must specify the same BLOCKSIZE that you specified during the EXPORT operation. If no value is given, the system assumes a value of 2048 bytes per block.

VSAM always defaults to an optimum buffer space value when using the IMPORT command; it ignores the buffer space value in the catalog entry and in the DLBL BUFSP statement (if specified).

See Examples 13, 14, 15, and 16 in "Appendix A: Sample Job Streams."

Using LISTCAT: Listing Catalog Entries

You use the LISTCAT command to list entries from a given catalog. The listing shows information about objects defined in the catalog, such as:

- Attributes of the object.
- Creation and expiration dates.
- Protection specification. Passwords and other protection information in an entry are not listed unless you specify the master password for the file defined by the entry or the master password for the catalog itself.
- Statistics, information regarding the dynamic usage or accessing of the data represented by the entry.
- Space specifications and allocations.
- Volume information.

The ENTRIES *(entryname)* parameter is used to specify the names of individual entries to be listed. The CLUSTER, DATA, INDEX, SPACE, NONVSAM, PATH, USERCATALOG, and ALTERNATEINDEX parameters are used to specify types of entries to be listed. The ALL, NAME, VOLUME, and ALLOCATION parameters specify the fields to be listed for each catalog entry.

The combination of these parameters allows you to tailor the scope of the listing to meet your needs. Refer to the chart in the LISTCAT command section to see the results of coding various combinations of *entryname* and type of entry (CLUSTER, DATA, etc.).

See Examples 3, 4, and 7 in "Appendix A: Sample Job Streams." See also "Appendix B: Interpreting LISTCAT Output Listings."

Using PRINT: Printing Data Records

You use the PRINT command to list some or all of the records of a sequential, indexed-sequential, or VSAM file in one of the following formats:

- Each byte as two hexadecimal digits (HEX).
- Each byte as a single character (CHARACTER).
- A combination of these two, side by side (DUMP).

You may specify a range of records to be printed in the same way as you do for copying records, namely:

- By key for indexed-sequential or key-sequenced files (FROMKEY, TOKEY).
- By RBA for key-sequenced or entry-sequenced files (FROMADDRESS, TOADDRESS).
- By relative-record number for relative-record files (FROMNUMBER, TONUMBER).

- By number of records for any type of file (COUNT, SKIP).

The components of a key-sequenced file can be listed individually. To list a component of a key-sequenced file, specify the component name as the file-ID in the DLBL statement. To print a user catalog as a key-sequenced file or to print any file cataloged in a user catalog, the user catalog must either be a job catalog or be specified in the CAT parameter of the DLBL job control statement.

Sequential, entry-sequenced, and relative-record files are listed in physical sequential order. Indexed-sequential and key-sequenced files can be listed in key order or in physical sequential order. A base cluster can be listed in alternate key sequence via a path.

Only the data content of 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 files.
- Its key for indexed-sequential and key-sequenced files.
- Its sequential record number for sequential and relative-record files.

If you use PRINT to locate errors in a file, Access Method Services terminates printing after four recoverable errors have been encountered while trying to read the file. The types of errors classified as "recoverable" are duplicate keys, wrong length records, records out of sequence, and I/O errors in the data component of a VSAM file.

See Examples 5, 6, and 8 in "Appendix A: Sample Job Streams." See also "Sample Output from PRINT" in the PRINT command section.

Using EXPORTRA/IMPORTRA: Recovering Catalog Entries and Data

You use the EXPORTRA command to retrieve catalog entries from catalog recovery areas placed on each volume owned by a *recoverable catalog.* VSAM catalogs may be defined with a RECOVERABLE attribute which allows a catalog to be recovered after it has been destroyed. Recovery is possible because catalog information about a given volume is recorded on that volume, as well as in the catalog itself.

Space for recovery information is automatically set aside when you define the first data space on a volume (provided you defined the catalog with the RECOVERABLE attribute). This recovery space, called the catalog recovery area (CRA), is suballocated from the first space allocated on the volume. A file with the UNIQUE attribute cannot be defined on a volume (if the catalog entry of the file is to be in a recoverable catalog) until a data space is created on the volume; either with the DEFINE SPACE, DEFINE MASTERCATALOG, or DEFINE USERCATALOG command.

There is no separate catalog or VTOC entry for the recovery space; its physical address is recorded by VSAM in the volume's format-4 label.

The recovery information in the CRA of a volume is immediately updated whenever parallel information in the catalog is modified by catalog management. To do so, the affected volume(s) must be mounted. The kind of operation which is to be performed on an object (data space, cluster, path, etc.), and which is reflected in the catalog, determines which volume(s) must be mounted:

- For a DEFINE operation, all volumes on the volume list on which space is being defined, or all volumes listed in the VOLUMES parameter list. In addition, for DEFINE PATH or DEFINE ALTERNATEINDEX, the

- By number of records for any type of file (COUNT, SKIP).

The components of a key-sequenced file can be listed individually. To list a component of a key-sequenced file, specify the component name as the file-ID in the DLBL statement. To print a user catalog as a key-sequenced file or to print any file cataloged in a user catalog, the user catalog must either be a job catalog or be specified in the CAT parameter of the DLBL job control statement.

Sequential, entry-sequenced, and relative-record files are listed in physical sequential order. Indexed-sequential and key-sequenced files can be listed in key order or in physical sequential order. A base cluster can be listed in alternate key sequence via a path.

Only the data content of 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 files.
- Its key for indexed-sequential and key-sequenced files.
- Its sequential record number for sequential and relative-record files.

If you use PRINT to locate errors in a file, Access Method Services terminates printing after four recoverable errors have been encountered while trying to read the file. The types of errors classified as "recoverable" are duplicate keys, wrong length records, records out of sequence, and I/O errors in the data component of a VSAM file.

See Examples 5, 6, and 8 in "Appendix A: Sample Job Streams." See also "Sample Output from PRINT" in the PRINT command section.

Using EXPORTRA/IMPORTRA: Recovering Catalog Entries and Data

You use the EXPORTRA command to retrieve catalog entries from catalog recovery areas placed on each volume owned by a *recoverable catalog.* VSAM catalogs may be defined with a RECOVERABLE attribute which allows a catalog to be recovered after it has been destroyed. Recovery is possible because catalog information about a given volume is recorded on that volume, as well as in the catalog itself.

Space for recovery information is automatically set aside when you define the first data space on a volume (provided you defined the catalog with the RECOVERABLE attribute). This recovery space, called the catalog recovery area (CRA), is suballocated from the first space allocated on the volume. A file with the UNIQUE attribute cannot be defined on a volume (if the catalog entry of the file is to be in a recoverable catalog) until a data space is created on the volume; either with the DEFINE SPACE, DEFINE MASTERCATALOG, or DEFINE USERCATALOG command.

There is no separate catalog or VTOC entry for the recovery space; its physical address is recorded by VSAM in the volume's format-4 label.

The recovery information in the CRA of a volume is immediately updated whenever parallel information in the catalog is modified by catalog management. To do so, the affected volume(s) must be mounted. The kind of operation which is to be performed on an object (data space, cluster, path, etc.), and which is reflected in the catalog, determines which volume(s) must be mounted:

- For a DEFINE operation, all volumes on the volume list on which space is being defined, or all volumes listed in the VOLUMES parameter list. In addition, for DEFINE PATH or DEFINE ALTERNATEINDEX, the

recovery volume, that is, the first volume of the prime index, if the base cluster is a key-sequenced file; otherwise, the first volume of the base data.

- For an ALTER operation, all volumes listed under ADDVOLUMES or REMOVEVOLUMES and the recovery volume. The recovery volume is as follows:
  - If the base cluster is a key-sequenced file, whether the object to be altered is the cluster or its related alternate index or any of their data or index components, or its related path, the recovery volume is the first volume of the prime index.
  - If the base cluster is an entry-sequenced file, whether the object to be altered is the cluster, its data component, its alternate index or the latter's data or index components, or its related path, the recovery volume is the first volume of the base data.
  - If the base cluster is a relative-record file, the recovery volume is the first volume of the base data.
- For a DELETE operation, all volumes on which space is being deleted. In addition, for a DELETE PATH or DELETE ALTERNATEINDEX, the recovery volume, that is, the first volume of the prime index, if the base cluster is a key-sequenced file; otherwise, the first volume of the base data.

See *VSE/VSAM Programmer's Reference* for the contents of CRAs and how catalog entries are placed in CRAs.

Note: If the catalog is changed by the REPRO command, or restored using a VSE utility, the CRAs owned by the catalog on volumes other than the catalog volume are unchanged. The catalog volume's CRA is restored using the VSE utility and is unchanged by REPRO. A LISTCRA list of the restored catalog reflects CRA-catalog mismatches.

If a file (alternate index or cluster) is not addressable by way of the catalog, the EXPORTRA command can be used to gain access to the file through the CRA to create a copy of the data which can be introduced back into the system by the IMPORTRA command. Selective recovery can be implemented after use of the LISTCRA command (with the NOCOMPARE and NAME options) to determine file-IDs and their associated volumes.

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 files on the restored volume. See "Resetting Catalog Entries."

The EXPORTRA command recovers the current information recorded in the CRAs of the affected volume(s). The IMPORTRA command then rebuilds complete catalog entries from the recovered information. Thus, if a destroyed volume was restored to some previous state, the catalog will, after the recovery operation, reflect that state. This implies that any changes to catalog entries made after that point (for example, to the entry of a destroyed file) will be deleted from the catalog.

The IMPORTRA command accepts portable files produced by VSE, DOS/VS, OS/VS1, or OS/VS2. However, IMPORTRA can accept only portable files produced by EXPORTRA (similarly, IMPORT can accept only portable files produced by EXPORT).

Using the information on such files, IMPORTRA reconstructs the catalog entries for all the files of the recovery volume and recreates the files themselves on the appropriate volumes.

IMPORTRA will define and load any VSAM objects found on the portable file. If the VSAM catalog is not recoverable, any nonVSAM files found on the portable file will also be defined by IMPORTRA. In addition, any user

catalog pointers found on the portable file will be defined into the receiving catalog if it is a master catalog.

VSAM always defaults to an optimum buffer space value when using the EXPORTRA/IMPORTRA commands; it ignores the buffer space value in the catalog entry and in the DLBL BUFSP statement (if specified).

See Examples 19, 20, and 21 in "Appendix A: Sample Job Streams" for the use of EXPORTRA/IMPORTRA in combination. Specific EXPORTRA examples follow.

Using EXPORTRA for Moving All Entries on One Volume

This example shows the EXPORTRA function for one volume, VSER00, owned by the VSAM master catalog. All of the files are contained wholly in the volume. All of the entries in the catalog recovery area on VSER00 and the files themselves are copied to a sequential file on a tape volume. SYS005 must be used in the ASSGN statement for magnetic tape output (OUTFILE).

```
// JOB   EXPORTRA FOR ONE VOLUME
// ASSGN SYS005,381
// TLBL  VOLOUT,'OUT.FILE',,TAPE01
// EXEC  IDCAMS,SIZE=AUTO
   EXPORTRA -
        OUTFILE(VOLOUT ENV(REW PDEV(2400))) -
        CRAVOLUMES (-
        VSER00 ALL)
/*
/&
```

The CRAVOLUMES parameter identifies the catalog recovery volume, VSER00. The CRA used for recovery is on this recovery volume. The ALL subparameter specifies that all files and the corresponding CRA entries are to be copied from the recovery volume specified.

No other parameters are required because all the files are contained in one volume.

Using EXPORTRA for Entries on Multiple Volumes

This example shows the EXPORTRA function for one volume owned by the VSAM master catalog. All files contained in the volume, VSER00, and their corresponding CRA entries describing these files are copied to a sequential file on a tape volume. Some of these are multivolume files but these files have their prime CRA entries on the recovery volume. Each volume that contains a portion of a multivolume file must be mounted. The entries in the CRAs of volumes other than the recovery volume are not recovered. SYS005 must be used in the ASSGN statement for magnetic tape output (OUTFILE).

```
// JOB   EXPORTRA FOR MULTIPLE VOLUMES
// ASSGN SYS005,382
// TLBL  VOLOUT,'OUT.FILE',,TAPE01
// EXEC  IDCAMS,SIZE=AUTO
   EXPORTRA -
        OUTFILE(VOLOUT ENV(REW PDEV(2400))) -
        CRAVOLUMES (-
            (VSER00 ALL) -
            (VSER01 NONE) -
            (VSER02 NONE)) -
        MASTERPW(MCATMRPW)
/*
/&
```

The CRAVOLUMES parameter identifies the catalog recovery volume, VSER00. The ALL subparameter specifies that all files on the recovery

volume and their corresponding CRA entries are to be copied. The VSER01 NONE and VSER02 NONE subparameters are needed because these volumes contain portions of multivolume files, but the CRA's on these volumes are not used to recover additional entries. Only those multivolume files that are described in the CRA of the recovery volume are recovered.

Using EXPORTRA for Selected Entries

This example shows the EXPORTRA function for selected files on volume VSER00. Three files on the volume and their corresponding CRA entries (those describing the files) are copied to a sequential file on another disk volume. The remaining files on the catalog recovery volume are not recovered. Two of the selected files are multivolume files. No files other than the multivolume files are recovered from the other volumes identified in the CRAVOLUMES parameter.

```
// JOB    EXPORTRA FOR SELECTED ENTRIES
// DLBL   VOLOUT,'OUT.FILE'
// EXTENT SYS013,231401,1,0,140,200
// EXEC   IDCAMS,SIZE=AUTO
   EXPORTRA -
         OUTFILE(VOLOUT) -
         CRAVOLUMES (-
          (VSER00 ENTRIES (-
            (LARGE.DATASET.A ) -
            (LARGE.DATASET.B ) -
            (LARGE.DATASET.C))) -
          (VSER01 NONE) -
          (VSER02 NONE))
/*
/&
```

The CRAVOLUMES parameter identifies the catalog recovery volume, VSER00. The ENTRIES subparameter identifies the three files and the corresponding CRA entries that are to be copied to a sequential file on another disk volume.

The VSER01 NONE and VSER02 NONE parameters are needed because these volumes contain portions of multivolume files, but the CRA's on these volumes are not used to recover additional entries.

Using RESETCAT: 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 would most likely use RESETCAT when a recoverable catalog or one or more of its owned volumes cannot be accessed for some catastrophic reason, such as a dropped pack, a head crash, an accidently scratched pack or some similar disaster. If such an accident were to occur, you can restore the inaccessible volume(s) from a backup copy and execute RESETCAT. CRAs contain enough information to reset the catalog entries, VSAM files owned by that catalog can again be accessed correctly.

Unlike EXPORTRA/IMPORTRA, RESETCAT is a one-step operation that enables you to recover your catalog without movement of data. RESETCAT does not check or process the data itself, only catalog, CRA, and VTOC VSAM entries are compared and reset. 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 LISTCRA (with the COMPARE option) can be run to see if there is a mismatch that requires RESETCAT to be run (see "LISTCRA: Analysis of Recoverable Catalogs" and 'Inaccessible Volume"). RESETCAT can now be used to synchronize the catalog with the volume. After access to the data has been regained, the files 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, run LISTCRA to help analyze the problem. If you find you cannot run LISTCRA or if you can and find you are unable to access your data, restore the catalog volume. Then run RESETCAT to synchronize your catalog to its owned volumes. If volumes have been added since the catalog backup was made, RESETCAT can be used to build these entries in the catalog from the volume's CRA. If volumes have been deleted since the last backup, use DELETE SPACE (FORCE) to delete the volumes space entries in the catalog and delete the files that resided on those volumes that are now marked unusable by RESETCAT 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:

1. If the unusable catalog is a user catalog, remove its catalog connector entry from the master catalog via EXPORT DISCONNECT and define a user catalog with the same name on a different volume. It must be defined with the RECOVERABLE attribute. Volumes owned by the unusable catalog should *not* be included as owned by the new user catalog. The DEFINE operation would flag this as an error condition because the volumes are already owned by a VSAM catalog.
2. Issue RESETCAT specifying (via CRAVOLUMES) all volumes owned by the previous catalog (including the unusable catalog's 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).

If some external cause, such as a power failure, were to cause RESETCAT to fail, then you must restore the volumes being reset and rerun RESETCAT. It

is advisable for you to have backup copies of your catalog and CRA volume(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.
- You must always specify a catalog, other than the catalog being reset, to be a work catalog (WORKCAT).
- 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 be reset while it is in use as a master catalog or it may be connected to the system as a user catalog.
- You need to use caution when using RESETCAT to recover accessibility of a volume that contains a portion of the 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.

Work File Space Requirements

RESETCAT requires a temporary work file for use as temporary storage while processing the command. The temporary work file is defined by RESETCAT and deleted at the end of command processing. The space required is suballocated from VSAM data spaces on the volumes assigned via WORKVOLUMES. If WORKVOLUMES is not specified, the work catalog is searched for a relative record file default model. If one is found, the work file is built on the volume(s) specified in the volumes list of the default model, otherwise, processing will terminate. Under normal conditions (no extensions) the amount of work file 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 work file data space must be provided to allow for this extension.

Considerations for Multivolume Files

The contents of a volume's CRA depend on the types of files the volume contains. It is important for you to understand on which CRA the catalog information resides for a particular object. "Catalog Recovery Area " in *VSE/VSAM Programmer's Reference* 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 file on two volumes, the volume that contains the first part of the entry sequenced file contains all the records that describe the entry sequenced file (including its allocation on the second

volume). The second volume, a secondary CRA for this object, contains information that shows that the entry sequenced file is allocated on the second volume. If the second volume had an I/O error that rendered it useless and a previous version of that volume were restored, the present catalog information may be erroneous; that is, the catalog may reflect the file's extent on the second volume. Prior to issuing a RESETCAT command compatible levels of volumes containing multi-volume files should be restored. RESETCAT would then be issued to reset the catalog to reflect the restored level of *all* files 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 file 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 file used would be the description that currently resides in the catalog. If the file is defined differently on the second volume (e.g., extents don't match), the file is marked unusable for that volume and the allocated space marked free.

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

When all volumes of a multivolume VSAM file, 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 it is, all information in the catalog is replaced for the file concerned. For all volumes of the multivolume file 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 file is defined on the volume.
- Check the file 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 file?

Although the above file checks guarantee that the catalog physically describes a file correctly, these checks cannot guarantee that the level of data in the file is at a consistent level. For instance, if a multivolume key-sequenced file 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 file, 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 file on the reset volume. Hence, only verification (no reset) occurs for these partial entries.The check ensures that the part of the file on the reset volume resides in the same physical place as described in the current catalog and is part of the same definition as the file described in the current catalog. RESETCAT cannot guarantee that the level of data *in* the file is at a consistent level among different volumes.

The catalog being reset is specified in the CATALOG parameter of the RESETCAT command. The catalog in which the work file is defined is specified in the WORKCAT parameter.

The WORKVOLUMES parameter provides a volume to define a temporary VSAM file. If WORKVOLUMES is not specified, the work catalog is searched for a relative record file default model. If one is found, the work file is built on the volume(s) specified in the volumes list of the default model, otherwise, processing will terminate.

The following example resets the user catalog "RESET.CAT" for the volume U2314G. This job might be run because the volume U2314G was destroyed and a previously copied version of the volume has been restored. At the completion of the reset operation the user catalog will correctly describe the files on the restored version of U2314G. U2314H is the second volume of some multi-volume files starting on U2314G. This volume may be needed when resetting information in the catalog pertaining to the multi-volume files.

```
// JOB  RESET A CATALOG
// EXEC IDCAMS,SIZE=AUTO
   RESETCAT-
       CATALOG(RESET.CAT/MRPASS)-
       CRAVOLUMES((U2314G ALL) (U2314H NONE))-
       WORKCAT(USER.CAT2/UPPASS2)-
       WORKVOLUMES(U2314J/WKPASS)-
       IGNORE
/*
/&
```

The next example resets the user catalog "UCAT1" for volume 333002. The WORKVOLUMES parameter has been omitted; therefore, the work catalog is searched for a relative record file default model. If one is found, the work file is built on the volume(s) specified in the volumes list of the default model; otherwise processing will terminate. (The user must have provided a relative record file default in the catalog.)

```
// JOB  RECOVER
// EXEC IDCAMS,SIZE=AUTO
   RESETCAT-
       CATALOG(UCAT1/MASTER) -
       CRAVOLUMES((333002 ALL)(333001 NONE)) -
       WORKCAT (MASTCAT/MCATMRPW) -
       MASTERPW(MCATMRPW) -
       IGNORE
/*
/&
```

Verifying a File's Accessibility

You can use the VERIFY command to protect data when a file was not closed successfully the last time it was processed. Access Method Services investigates whether an entry-sequenced file, a relative-record file, or both the data and index of a key-sequenced file or alternate index were properly closed. You cannot verify a path defined over an alternate index, but you can verify a base cluster using a NOUPDATE path defined directly over the base cluster. This prevents unnecessary allocation of the upgrade set, if one exists.

If the user has started loading into a file with the RECOVERY option (see DEFINE CLUSTER) then the VERIFY command can be used to save the file from reload. If, however, SPEED was used, VERIFY will not help since preformatting is not done. A reload of the file is required if a failure occurs

during file load with SPEED option. Note that if a VSAM file is extended (due to resume loading, adding records, etc.) VSAM always extends in RECOVERY mode. Thus VERIFY can be used to recover from a CLOSE failure.

The end of the data or of the index of a file is indicated by the end-of-file indicator and by information in the component's catalog entry. If a file is divided into key ranges, the end of each key range is indicated by an end-of-file indicator. The end may be improperly indicated in the catalog if an error prevented VSAM from closing the file. VERIFY closes the file and modifies the catalog entry, if necessary, to correspond with the file. This ensures that your data will not be overwritten inadvertently at a later date.

If a VSE failure occurs before the file is closed (that is, before your program issues CLOSE or TCLOSE), the file's catalog entry may not be updated, and therefore the entry may contain obsolete end-of-file information. The file's real end-of-file indicators are written in the file, but are not reflected in its catalog entry.

When this file is opened (first OPEN after system failure, for example), VSAM Open sets a "file improperly closed" return code. When the file is closed, VSAM Close resets the "file improperly closed" indicator but does not update erroneous catalog information that resulted from the system failure. Subsequently, when the file is opened a second time, the end-of-file information in its catalog entry is erroneous, but VSAM Open sets the "file opened correctly" return code. When your program attempts to process records beyond the point that the catalog indicates is the end of the file, a "no record found," "invalid RBA," or "end-of-data" error results. Execution of the VERIFY command before the file is opened a second time ensures that the catalog end-of-file information agrees with the end-of-file indicator in the file's data component. The following example shows how you can close the file and correct the end-of-file information in the catalog by issuing the VERIFY command.

```
// JOB   VERIFY END-OF-FILE
// EXEC  IDCAMS,SIZE=AUTO
   LISTCAT  ENTRIES(CREDITS) -
              ALL
   VERIFY   DATASET(CREDITS)
   LISTCAT  ENTRIES(CREDITS) -
              ALL
/*
/&
```

CREDITS is the name of the object whose entry is to be listed and verified. The first LISTCAT command lists the file's catalog entry, showing the file's parameters as they were when the file was last closed.

The VERIFY command updates the file's catalog entry to show the file's real end-of-file indicator(s).

The second LISTCAT command lists the file's catalog entry again. This time it shows the point at which processing stopped due to system failure. This information should help you determine how much of your data was added correctly before the system failed.

Chapter 3: Access Method Services Commands

This chapter sets out the functional and modal command formats. Each command format is shown, followed by a discussion of each parameter.

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 files and other objects that have already been defined.
- BLDINDEX command, which is used to build one or more alternate indexes over a base cluster.
- CANCEL command, which allows you to cancel either the current job or job step.
- 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 file.
- DEFINE MASTERCATALOG command, which is used to define the master catalog.
- DEFINE NONVSAM command, which is used to define a catalog entry for a nonVSAM file.
- 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 files and other objects, including catalogs and nonVSAM files.
- EXPORT command, which is used to create a portable copy of VSAM files and to disconnect user catalogs.
- EXPORTRA command, which is used to recover VSAM catalog entries from catalog recovery areas and, for VSAM clusters and alternate indexes, to recover the data itself by means of catalog recovery areas.
- IMPORT command, which is used to redefine and reload VSAM files from a portable copy and to connect user catalogs.
- IMPORTRA command, which is used to reconstruct multiple VSAM files from a file 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 specified catalog recovery areas.
- PRINT command, which is used to print both VSAM and nonVSAM files.

- REPRO command, which is used to copy and load both VSAM and nonVSAM files and VSAM catalogs.
- RESETCAT command, which is used to synchronize a recoverable catalog and its owned volumes.
- VERIFY command, which is used to verify and correct certain problems that have made your file unusable.

Notational Conventions

A uniform system of notation describes the format of Access Method Services commands. This notation (that is, [ ], { }, |, ...) 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; 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. In examples, an ellipsis may indicate that entries not relevant to the examples have been omitted.
- Underscored type indicates a default option. If the parameter is omitted and none is implied by the presence of another parameter in the set, the underscored value is assumed.
- Other punctuation (parentheses, commas, spaces, etc.) must be entered as shown. A space is indicated by ␣.
- UPPER CASE type indicates the exact characters to be entered. Such items must be entered exactly as illustrated (in upper case).
- *Italic* type specifies fields to be supplied by the user.

ALTER

The ALTER command is used to change attributes in catalog entries. To alter an entry, you need to supply its name and the attributes to be altered. For example, using this command you can change the name and passwords of an object. Any attributes not explicitly specified remain as they were originally defined or last altered.

A key-sequenced file and alternate index consist of three components: an index and data component and the cluster entry itself. An entry-sequenced and relative-record file consist of two components: a data component and the cluster entry. Most attributes of a file 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 specify the cluster name in the *entryname* parameter, only these attributes can be changed. If you specify the cluster name to alter any attribute not directly associated with the cluster entry, the ALTER request is terminated and an error message is issued. The opposite is also true; if you specify a data or index component name and attempt to alter an attribute directly associated with the cluster entry, your ALTER command is terminated and an error message is issued.

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. Use the LISTCAT command to determine the system-generated names of the data and index components.

Catalog Entry-Types That Can Be Altered

Figure 3-1 lists all the ALTER parameters and the entry-types to which each parameter applies. For example, you can specify a new value for the ATTEMPTS parameter for all entry-types except the nonVSAM entry.

See "Using ALTER: Altering Catalog Entries" and Example 7 in "Appendix A: Sample Job Streams" for a discussion of ALTER functions and examples.

ALTER Parameters	Type of Catalog Entry								
	Alternate Index			Cluster			PATH	MCAT[1] UCAT[1]	NonVSAM
		DATA	INDEX		DATA	INDEX			
ADDVOLUMES		X	X		X	X			
ATTEMPTS	X	X	X	X	X	X	X	X	
AUTHORIZATION	X	X	X	X	X	X	X	X	
BUFFERSPACE		X			X			X	
CODE	X	X	X	X	X	X	X	X	
CONTROLPW	X	X	X	X	X	X	X	X	
ERASE		X			X				
EXCEPTIONEXIT		X	X		X	X			
FOR	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	
NEWNAME	X	X	X	X	X	X	X		X
NOERASE		X			X				
NONUNIQUEKEY		X							
NOUPDATE							X		
NOUPGRADE	X								
NOWRITECHECK		X	X		X	X			
NULLIFY	X	X	X	X	X	X	X	X	
AUTHORIZATION	X	X	X	X	X	X	X	X	
CODE	X	X	X	X	X	X	X	X	
CONTROLPW	X	X	X	X	X	X	X	X	
EXCEPTIONEXIT		X	X		X	X			
MASTERPW	X	X	X	X	X	X	X	X	
MODULE	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	
RETENTION	X			X			X	X	
STRING	X	X	X	X	X	X	X	X	
UPDATEPW	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	
RECORDSIZE	X	X		X	X		X		
REMOVEVOLUMES		X	X		X	X			
SHAREOPTIONS		X	X		X	X			
TO	X			X			X	X	
UNINHIBIT		X	X		X	X			
UNIQUEKEY		X[2]							
UPDATE							X		
UPDATEPW	X	X	X	X	X	X	X	X	
UPGRADE	X[3]								
WRITECHECK		X	X		X	X			

[1] You cannot alter a catalog's data or index component entries.

[2] The data component must be empty.

[3] The alternate index must be empty.

Note: The "X" indicates you can respecify the value or attribute for the above type of VSAM catalog entry.

Figure 3-1. ALTER Parameters and the Entry-Types to Which Each Applies.

The format of the ALTER command is:

ALTER	*entryname* [/ *password*] [ADDVOLUMES (*volser* [ъ *volser* . . .])] [ATTEMPTS (*number*)] [AUTHORIZATION (*entrypoint* [ъ *string*])] [BUFFERSPACE (*size*)] [CATALOG (*catname* [/ *password*] [ъ *dname*])] [CODE (*code*)] [CONTROLPW (*password*)] [ERASE|NOERASE] [EXCEPTIONEXIT (*mname*)] [FILE (*dname*)] [FREESPACE (*cipercent* [ъ *capercent*])] [KEYS (*length* ъ *offset*)] [MASTERPW (*password*)] [NEWNAME (*newname*)] [NULLIFY ([AUTHORIZATION(MODULE|STRING)] [CODE] [CONTROLPW] [EXCEPTIONEXIT] [MASTERPW] [OWNER] [READPW] [RETENTION] [UPDATEPW])] [OWNER (*ownerid*)] [READPW (*password*)] [RECORDSIZE (*average* ъ *maximum*)] [REMOVEVOLUMES (*volser* [ъ *volser* . . .])] [SHAREOPTIONS (*value* [ъ *reserved*])] [TO (*date*) | FOR (*days*)] [UNINHIBIT|INHIBIT] [UNIQUEKEY|NONUNIQUEKEY] [UPDATE|NOUPDATE] [UPDATEPW (*password*)] [UPGRADE|NOUPGRADE] [WRITECHECK|NOWRITECHECK]

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

ALTER Parameters: Summary

Most of the ALTER parameters are the same as those of the DEFINE command. They can be divided into the following categories:

Name, which specifies:

- The entry to be altered (entryname). **This parameter is required.**
- A new name for the entry (NEWNAME).
- The catalog containing the entry to be altered (CATALOG).

Data organization, which specifies:

- The length and offset of the key-field (KEYS).

- How a path is to be accessed (UPDATE, NOUPDATE).
- Whether an alternate index is to be kept up to date with its base cluster (UPGRADE, NOUPGRADE).
- Whether an alternate key may belong to more than one data record in the base cluster (UNIQUEKEY, NONUNIQUEKEY).

Allocation, which specifies:

- The amount of free space to be left in control intervals and control areas (FREESPACE)
- The amount of buffer space to be provided (BUFFERSPACE).
- That additional volumes are to be added and removed from the list of candidate volumes (ADDVOLUMES, REMOVEVOLUMES).
- The length of records (RECORDSIZE).

Protection and integrity, which specifies:

- Passwords (MASTERPW, CONTROLPW, UPDATEPW, READPW).
- The protection attributes to be nullified (NULLIFY).
- The type of access available to do certain types of operations (UNINHIBIT, INHIBIT).
- A prompting code (CODE) and number of attempts allowed to provide the correct password in response to prompting at the operator's console (ATTEMPTS).
- A user-supplied authorization verification routine (AUTHORIZATION).
- The owner of the alternate index (OWNER).
- A retention period (FOR, TO) and whether data is to be erased when an entry is deleted (ERASE, NOERASE).
- Whether write-check operations are to be performed as records are inserted (WRITECHECK, NOWRITECHECK).
- The share options to be associated with an entry (SHAREOPTIONS).
- A user-supplied module to be given control when an exception occurs during processing (EXCEPTIONEXIT).

ALTER Parameters

entryname [/ *password*]
is a required parameter that names the object whose entry is to be altered and supplies the master password if the entry defines a password-protected object. Alternatively, the master password of the catalog that contains the entry can be specified in the CATALOG parameter. This parameter must be the first parameter following ALTER.

If KEYS is not specified and if a data or index component entry is to be altered, the master password of the cluster, component, or catalog can be supplied. If KEYS is specified, the master password of the cluster or catalog (but not the component) can be supplied.

If neither KEYS nor RECORDSIZE is specified, the entry that defines the object named is altered.

If KEYS or RECORDSIZE is specified, the entry being altered is as follows:

- If *entryname* names a path over an alternate index or a base cluster, the entry that defines the data component of the alternate index or base cluster, respectively, whichever is named in PATHENTRY in the path's definition.
- If *entryname* names an alternate index or a base cluster, the data component of the alternate index or base cluster, respectively.
- If *entryname* names a data component, the entry of that component.
- If *entryname* names an index component, an error condition results. (*entryname* can be specified for an index component only if neither KEYS nor RECORDSIZE is specified.)

If any other parameters are specified with KEYS or RECORDSIZE, they will alter the same object as KEYS or RECORDSIZE. For example, if the master password is to be altered (and neither KEYS nor RECORDSIZE is specified) and the *entryname* names a base cluster, then the master password is altered at the cluster level only. If, however, KEYS or RECORDSIZE is also specified, the master password is altered for the base cluster's data component only.

ADDVOLUMES (*volser* [*volser* . . .])
specifies volumes to be added to the list of candidate volumes associated with the data or index component entry being altered. (A candidate volume is reserved for future use by VSAM.) See footnote.

A volume serial number, *volser*, may contain one through six alphameric, national (@, #, and $), and special characters (commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs). Single quotation marks within a volume serial number must be coded as two single quotation marks. A volume serial number must be enclosed in single quotation marks if it contains a special character. For consistency with VSE job control, only alphameric characters should be used.

Volumes to be added as candidate volumes must already be owned by the catalog that contains the entry being altered; that is, (a) space must have been defined on a volume to be added via the DEFINE SPACE command, or (b) the volume must have been identified as a candidate volume in the DEFINE SPACE command.

Abbreviation: AVOL

ATTEMPTS (*number*)
changes the maximum number of times (0 through 7) 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. In this case, if the password has not been supplied through the program, the file will not be opened for processing. This value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. See footnote.

Note: Refer to Figure 3-1 for applicable entry-types to be altered.

Abbreviation: ATT

AUTHORIZATION(*entrypoint* [*string*])
specifies (or changes) a user security verification routine (USVR). See footnote.

entrypoint – specifies the name of the user security verification routine. The name may contain one through eight alphameric, national (@, #, $), or special (hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or a national character. *entrypoint* is the phase name that appears in the core image library.

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

Abbreviation: AUTH

BUFFERSPACE(*size*)
changes the minimum space to be provided for buffers. See footnote.

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 file is key-sequenced, one index component control interval.

size – is the number of bytes to be provided for buffers. This value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. If the specified value is less than the amount VSAM requires, VSAM gets the amount it requires when the file is opened.

Abbreviation: BUFSPC or BUFSP

CATALOG(*catname* [/ *password*])
specifies the name of the catalog that contains the entry to be altered. You do not have to specify this parameter (unless it is needed for password specification) when the entry to be altered exists in the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

catname – is the name of the catalog.

password – specifies the master password of the catalog. If the entry to be altered is password-protected and the catalog is also password-protected, you can supply the proper password, either through this parameter or through the *entryname* parameter. If both passwords are specified, the catalog password is used.

Abbreviation: CAT

Note: Refer to Figure 3-1 for applicable entry-types to be altered.

CODE (*code*)
specifies a new code name for the object whose entry is being altered. See footnote.

The code may contain one through eight EBCDIC characters. The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. If a single quotation mark appears within the code, it must be coded as two single quotation marks. Code may also be specified in hexadecimal (X'code') form.

CONTROLPW (*password*)
specifies a new control password for the object whose entry is being altered. See footnote.

password – a one through 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. Password may also be coded in hexadecimal form (X'password').

Abbreviation: CTLPW

ERASE | NOERASE
specifies (or changes) whether or not the data component whose entry is being altered is to be erased (overwritten with binary zeros) when its entry in the catalog is deleted. See footnote.

Abbreviations: ERAS and NERAS

EXCEPTIONEXIT (*mname*)
specifies (or changes) the name of the user module (i.e., the phase name in the core image library) to be given control when an exception occurs during the processing of the object whose entry is being altered. See footnote.

Abbreviation: EEXT

FOR (see TO)

FREESPACE (*cipercent* [b *capercent*])
changes the amount of space that is to be left free after the initial load or any extension and after any split of control intervals (*cipercent*) and control areas (*capercent*) during sequential processing. See footnote.

The amounts are specified as percentages which must contain only the numbers 0 to 100 expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Abbreviation: FSPC

INHIBIT (see UNINHIBIT)

Note: Refer to Figure 3-1 for applicable entry-types to be altered.

KEYS (*length*ƀ*offset*)

changes the length and offset of the object's key-field. See footnote.

If the entry being altered is an alternate index, *offset* applies to the alternate key-field in the data records in the base cluster. For more information, see *entryname*.

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

- The object whose entry is being altered is an alternate index, a path, a cluster, or a data component of an alternate index or key-sequenced cluster.
- 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.
- 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.

*length*ƀ*offset* - specifies the length of the key-field (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.

MASTERPW (*password*)

specifies a new master password for the entry being altered. See footnote.

See CONTROLPW for the definition of password.

Abbreviation: MRPW

NEWNAME (*newname*)

specifies a new name for the entry being altered. See footnote.

The new name may contain 1 to 44 alphameric characters, national characters (@, #, and $), and two special characters (hyphen and 12-0 overpunch). Names that contain more than eight characters must be segmented by periods; one through eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or a national character.

Abbreviation: NEWNM

NOERASE (see ERASE)

NONUNIQUEKEY (see UNIQUEKEY)

NOUPDATE (see UPDATE)

NOUPGRADE (see UPGRADE)

NOWRITECHECK (see WRITECHECK)

Note: Refer to Figure 3-1 for applicable entry-types to be altered.

NULLIFY
specifies that the protection attributes identified by the subparameters of NULLIFY are to be nullified. Any attribute specified in NULLIFY is nullified before attributes specified in other parameters take effect. If all levels of passwords are nullified and none are respecified, CODE, AUTHORIZATION, and ATTEMPTS no longer have any effect. You do not have to NULLIFY attributes that you are respecifying. See footnote.

Abbreviation: NULL

AUTHORIZATION(MODULE|STRING)
specifies that either the user authorization routine (MODULE) or the user authorization record (STRING) is to be nullified. See footnote.

When MODULE is specified, the module name is removed from the catalog record, but the module itself is not deleted. If you nullify the user authorization module, the user authorization record (character string) is also nullified. If, however, you nullify the authorization record, the corresponding module is not nullified.

Abbreviations: AUTH, MDLE, and STRG

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

CONTROLPW
specifies that the control password is to be nullified.

Abbreviation: CTLPW

EXCEPTIONEXIT
specifies that the exception exit is to be nullified.

Abbreviation: EEXT

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 existing password automatically becomes the master password.

Abbreviation: MRPW

OWNER
specifies that the owner identification is to be nullified.

READPW
specifies that the read password is to be nullified.

Abbreviation: RDPW

RETENTION
specifies that the retention period, specified in a TO or FOR parameter (of a DEFINE or ALTER command), is to be nullified.

Abbreviation: RETN

UPDATEPW
specifies that the update password is to be nullified.

Abbreviation: UPDPW

Note: Refer to Figure 3-1 for applicable entry-types to be altered.

OWNER (*ownerid*)
specifies the new owner identification of the entry being altered. See footnote.

The *ownerid* may contain one through eight EBCDIC characters. The *ownerid* must be enclosed in single quotation marks if it contains a comma, semicolon, blank, parenthesis, slash or slash/asterisk combination. 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 also be expressed in hexadecimal form (X'ownerid').

READPW (*password*)
specifies a new read password for the object whose entry is being altered. See footnote.

See CONTROLPW for the definition of password.

Abbreviation: RDPW

RECORDSIZE (*average*ƀ *maximum*)
specifies the new average and maximum lengths for the data records in the object. See footnote.

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

- The object whose entry is being altered is an alternate index, a cluster, a path, or a data component of a key-sequenced cluster or alternate index.
- 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. Refer to the DEFINE command (that defines the object) for the default value.
- 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.

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*.

Abbreviation: RECSZ

REMOVEVOLUMES (*volser* [ƀ *volser* . . .])
specifies volumes to be removed from the list of candidate volumes associated with the data or index component entry being altered. See footnote.

Note: Refer to Figure 3-1 for applicable entry-types to be altered.

Volumes specified are removed after any new volumes are added to the volume list (by way of ADDVOLUMES). If a volume to be removed contains data that belongs to the entry being altered, the volume is not removed and an error message is issued. You cannot remove the recovery volume or the last (only) volume of a NOALLOCATION file component even though the volume is indicated as a candidate.

See ADDVOLUMES for information on how to code *volser*.

Abbreviation: RVOL

SHAREOPTIONS (*value* [*reserved*])

specifies a new share-option value for a file (data or index component of a cluster or alternate index.) A file's SHAREOPTIONS values cannot be altered if the file is currently open for another program. See footnote.

value – indicates the degree of file sharing allowed:

1
specifies that any number of users on one or more VSE systems can read the file, *or* a single user can write to the file. 1 is the default value.

2
specifies that any number of users on one or more VSE systems can read the file, *and a single user can also write to the file.*

3
specifies that any number of users on one or more VSE systems can both read and write to the file.

4
specifies that any number of users on a single VSE system can read and write to the file, and any number of users on one or more other VSE systems can read the file, that is, only one VSE system has write access to the file, all other systems have read only access.

reserved – for OS/VS compatibility only. The valid OS/VS values are 3 and 4.

Abbreviation: SHR

TO (*date*) | FOR (*days*)

specifies the new retention period for the entry being altered. See footnote.

TO (*date*)

specifies the new date until which the entry is to be kept. The date has the form yyddd, where yy (00 through 99) is the year and ddd (001 through 366) is the day. If you specify an invalid date, such as (99999), VSAM returns a condition code of '0' even though the expiration date remains unchanged.

FOR (*days*)

specifies the new number of days for which the entry is to be kept. If the number specified is 0 through 1830, the cluster is retained for the number of days specified; if the number is 1831 through 9999 (maximum), the cluster is retained through the year 1999. This value

---

Note: Refer to Figure 3-1 for applicable entry-types to be altered.

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

UNINHIBIT | INHIBIT
specifies whether the entry being altered can be accessed for any operation or for read operations only. See footnote.

UNINHIBIT specifies that the read-only restriction set by a previous ALTER or EXPORT command is to be removed. INHIBIT specifies that the object defined by the entry being altered is only to be read.

Abbreviations: UNINH and INH

UNIQUEKEY | NONUNIQUEKEY
specifies whether the alternate-key value can be found in more than one of the base cluster's data records. See footnote.

UNIQUEKEY indicates that each alternate-key value is in one and only one data record in the base cluster. NONUNIQUEKEY indicates that an alternate key may be in more than one data record in the base cluster.

UNIQUEKEY can be specified only for an empty data component of an alternate index, that is, one that contains no data records.

The data component need not be empty in order to specify NONUNIQUEKEY. When NONUNIQUEKEY is specified, respecify the RECORDSIZE parameter to account for the increased record size resulting from multiple prime key pointers in the alternate index data record. Note that you may be required to rebuild the alternate index because RECORDSIZE can only be specified for an empty file.

Abbreviations: UNQK and NUNQK

UPDATE | NOUPDATE
specifies whether or not the upgrade set belonging to the base cluster of the path is to be updated. See footnote.

Abbreviations: UPD and NUPD

UPDATEPW (*password*)
specifies a new update password for the object whose entry is being altered. See footnote.

See CONTROLPW for the definition of password.

Abbreviation: UPDPW

UPGRADE | NOUPGRADE
specifies whether VSAM is to keep the alternate index up to date with its base cluster or you assume responsibility to do so. See footnote.

UPGRADE indicates that whenever a change to the base cluster (other than through control interval access) calls for updating the alternate index, VSAM is to do it. If NOUPGRADE is specified, you are responsible for maintaining the alternate index.

Note: Refer to Figure 3-1 for applicable entry-types to be altered.

If the base cluster is open when the ALTER command is issued, the UPGRADE or NOUPGRADE attribute is not effective until the next open of the base cluster that creates a new set of control blocks.

UPGRADE can be specified only for an empty alternate index, that is, one that is defined but not built. NOUPGRADE can be specified for an alternate index at any time.

Abbreviations: UPG and NUPG

`WRITECHECK | NOWRITECHECK`
specifies whether to check the data transfer of records written in the object. See footnote.

Abbreviations: WCK and NWCK

Note: Refer to Figure 3-1 for applicable entry-types to be altered.

BLDINDEX

The BLDINDEX command (BIX) is used to build one or more alternate indexes over a base cluster. Both the alternate index and the base cluster must have been previously defined as related to each other by a path, and the base cluster must have been loaded. The base cluster over which the alternate index is built must contain either an entry- or a key-sequenced file, and this file cannot be reusable. See "Building an Alternate Index" and Example 8 in "Appendix A: Sample Job Streams" for a discussion of BLDINDEX functions and examples.

The format of the BLDINDEX command is:

BLDINDEX	INDATASET (*entryname* [/ *password*]) INFILE (*dname* [/ *password*]) OUTDATASET (*entryname* [/ *password*] . . .) OUTFILE (*dname* [/ *password*] . . .) [CATALOG (*catname* [/ *password*])] [EXTERNALSORT | INTERNALSORT] [WORKFILES (*dname1* [*dname2*)] [WORKVOLUMES (*volser* . . .)]

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

BLDINDEX Parameters

CATALOG (*catname* [/ *password*])
specifies the name and the update password, if required, of the catalog in which sort work files are to be defined if an external sort is performed. (See "Building an Alternate Index" for a discussion of sorting.)

CATALOG is required only when the catalog is password protected.

Abbreviation: CAT

EXTERNALSORT | INTERNALSORT
specifies whether an external or internal sort is to be performed.

Abbreviations: ESORT, ISORT

Default: INTERNALSORT

INDATASET (*entryname* [/ *password*])
specifies the *entryname* of (a) the base cluster or (b) the path that points to the base cluster. The entry must be in the default catalog. If the base cluster is password protected, you may give any one of its passwords. If a path over the base cluster is specified, the password must be that of the path. **This parameter is required.**

Abbreviation: IDS

OUTDATASET (*entryname* [/ *password*] . . . .)

specifies the *entryname* of the alternate index or the path that points to the alternate index. If the alternate index is password protected, you must give its update or higher-level password. If a path over the alternate index is specified, the password must be that of the path. More than one alternate index may be built simultaneously by specifying multiple *entryname* subparameters. The entries must be in the default catalog. **This parameter is required.**

Abbreviation: ODS

WORKVOLUMES (*volser* [*volser* . . .])

specifies up to ten volume serial numbers on which to suballocate space for two work files if an external sort is to be performed. If enough storage space is available, BLDINDEX performs an internal sort, and volumes for the work files are not necessary. You can omit this parameter if an entry-sequenced file default model exists in the catalog. In this case, the needed volume(s) are selected from the volume list of the default model. The sort work files are deleted by BLDINDEX prior to the end of processing the command.

Abbreviation: WVOL

CANCEL

The CANCEL command allows you to cancel either the current job or job step. You can use this command along with the modal commands, IF-THEN-ELSE, to conditionally terminate the current job or job step. If Access Method Services was invoked through a subroutine call, the invoker will receive control with a code of 16 in register 15. The CANCEL command will issue an appropriate termination message.

The format of the CANCEL command is:

CANCEL	{STEP|JOB}

CANCEL Parameters

STEP
specifies that the current job step is to be canceled. STEP is the default.

JOB
specifies that the current job is to be canceled.

DEFINE ALTERNATEINDEX

The DEFINE ALTERNATEINDEX command (DEF AIX) defines and relates an alternate index to an existing VSAM file (base cluster). The base cluster over which the alternate index is built may contain either an entry- or a key-sequenced file. An alternate index cannot be built over a relative-record file, another alternate index, a catalog, or a reusable or empty cluster. The alternate index is built using the BLDINDEX command and kept up to date, if desired, by VSAM. **An alternate index is a spanned, key-sequenced file.** See "Defining an Alternate Index" and Example 8 in "Appendix A: Sample Job Streams" for examples and more information.

The format of the DEFINE ALTERNATEINDEX command is shown below. Note that the organization of the command consists of three groupings. These groupings are referred to as *levels* in this book; note that some parameters appear in more than one level. (The optional CATALOG parameter stands alone; it does not belong to any level.)

Alternate Index Level	
DEFINE	ALTERNATEINDEX (NAME (*entryname*) [ATTEMPTS (*number*)] [AUTHORIZATION (*entrypoint* [Þ *string*])] [BUFFERSPACE (*size*)] [CODE (*code*)] [CONTROLINTERVALSIZE (*size*)] [CONTROLPW (*password*)] [CYLINDERS (*primary* [Þ *secondary*]) | † BLOCKS (*primary* [Þ *secondary*]) | † RECORDS (*primary* [Þ *secondary*]) | † TRACKS (*primary* [Þ *secondary*])] † [ERASE | NOERASE] [EXCEPTIONEXIT (*mname*)] [FILE (*dname*)] † [FOR (*days*) | TO (*date*)] [FREESPACE (*cipercent* [Þ *capercent*])] [IMBED | NOIMBED] [KEYRANGES ((*lowkey*Þ *highkey*) [Þ (*lowkey*Þ *highkey*) . . .])] [KEYS (*length*Þ *offset*)] [MASTERPW (*password*)] [MODEL (*entryname* [/ *password*] [Þ *catname* [/ *password*] [Þ *dname*]])] [ORDERED | UNORDERED] [OWNER (*ownerid*)] [READPW (*password*)] [RECORDSIZE (*average*Þ *maximum*)]

† This parameter is required under certain conditions. See the following parameter explanations or Appendix E for additional information.

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

(continued on next page)

Alternate Index Level (continued)

```
   [RECOVERY|SPEED]
   [RELATE( entryname [/ password ]) ]                    †
   [REPLICATE|NOREPLICATE]
   [REUSE|NOREUSE]
   [SHAREOPTIONS( value [ƀ reserved ]) ]
   [UNIQUE|SUBALLOCATION|NOALLOCATION]
   [UNIQUEKEY|NONUNIQUEKEY]
   [UPDATEPW( password ) ]
   [UPGRADE|NOUPGRADE]
   [USECLASS( primary [ƀsecondary]) ]
   [VOLUMES( volser [ƀ volser . . . ])
     |DEFAULTVOLUMES]
   [WRITECHECK|NOWRITECHECK]
)
```

Data Component Level

```
[DATA
(
   [ATTEMPTS( number ) ]
   [AUTHORIZATION( entrypoint [ƀ string ]) ]
   [BUFFERSPACE( size ) ]
   [CODE( code ) ]
   [CONTROLINTERVALSIZE( size ) ]
   [CONTROLPW( password ) ]
   [CYLINDERS( primary [ƀ secondary ]) |                  †
     BLOCKS( primary [ƀ secondary ]) |                    †
     RECORDS( primary [ƀ secondary ]) |                   †
     TRACKS( primary [ƀ secondary ]) ]                    †
   [ERASE|NOERASE]
   [EXCEPTIONEXIT( mname ) ]
   [FILE( dname ) ]                                       †
   [FREESPACE( cipercent [ƀ capercent ]) ]
   [KEYRANGES( ( lowkeyƀ highkey )
     [ƀ( lowkeyƀ highkey ) . . . ]) ]
   [KEYS( lengthƀ offset ) ]
   [MASTERPW( password ) ]
   [MODEL( entryname [/ password ]
     [ƀ catname [/ password ] [ƀ dname ] ]) ]
   [NAME( entryname ) ]
   [ORDERED|UNORDERED]
   [OWNER( ownerid ) ]
   [READPW( password ) ]
```

† This parameter is required under certain conditions. See the following parameter explanations or Appendix E for additional information.

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

(continued on next page)

Data Component Level (continued)

```
   [RECORDSIZE( average[,]maximum )]
   [RECOVERY|SPEED]
   [REUSE|NOREUSE]
   [SHAREOPTIONS( value [,reserved ] )]
   [UNIQUE|SUBALLOCATION|NOALLOCATION]
   [UNIQUEKEY|NONUNIQUEKEY]
   [UPDATEPW( password )]
   [USECLASS( primary [,secondary])]
   [VOLUMES( volser [,volser...])|
     DEFAULTVOLUMES]
   [WRITECHECK|NOWRITECHECK]
)]
```

Index Component Level

```
[INDEX
(
   [ATTEMPTS( number )]
   [AUTHORIZATION( entrypoint [,string ] )]
   [CODE( code )]
   [CONTROLINTERVALSIZE( size )]
   [CONTROLPW( password )]
   [CYLINDERS( primary [,secondary ] )|          †
     BLOCKS( primary [,secondary ] )|            †
     RECORDS( primary [,secondary ] )|           †
     TRACKS( primary [,secondary ] )]            †
   [EXCEPTIONEXIT( mname )]
   [FILE( dname )]                               †
   [IMBED|NOIMBED]
   [MASTERPW( password )]
   [MODEL( entryname [/ password ]
     [,catname [/ password ] [,dname ] ])]
   [NAME( entryname )]
   [ORDERED|UNORDERED]
   [OWNER( ownerid )]
   [READPW( password )]
   [REPLICATE|NOREPLICATE]
   [REUSE|NOREUSE]
   [SHAREOPTIONS( value [,reserved ] )]
   [UNIQUE|SUBALLOCATION|NOALLOCATION]
   [UPDATEPW( password )]
   [USECLASS( primary [,secondary])]
   [VOLUMES( volser [,volser...])|
     DEFAULTVOLUMES]
   [WRITECHECK|NOWRITECHECK]
)]
```

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

† This parameter is required under certain conditions. See the following parameter explanations or Appendix E for additional information.

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

DEFINE ALTERNATEINDEX Parameters: Summary

The parameters of the DEF AIX command can be divided into the following categories:

Name, which specifies:

- The name of the alternate index (NAME) and, optionally, of its components. **This parameter is required.**
- The name of the catalog in which the alternate index is to be defined (CATALOG).
- The name of the alternate index to be used as a model for this alternate index (MODEL). See "How to Use One Object as a Model for Another Object," in *VSE/VSAM Programmer's Reference* for more information.
- The identity of the base cluster over which the alternate index is to be built (RELATE). **This parameter is required except for default models.**

Data organization, which specifies:

- Whether an index record is to be written as many times as it will fit on a track (REPLICATE, NOREPLICATE).
- Whether the index's sequence-set records (the lowest level of the index) are to be placed with the data component of the alternate index (IMBED, NOIMBED). See "Optimizing VSAM's Performance" in *VSE/VSAM Programmer's Reference* for more information on the index options.
- The length and offset of the alternate-key field in the base cluster (KEYS).
- Whether the alternate index is to be reuseable (REUSE, NOREUSE).
- Whether the alternate-key value may belong to more than one data record in the base cluster (UNIQUEKEY, NONUNIQUEKEY).
- Whether VSAM is to upgrade the alternate index each time its base cluster is modified (UPGRADE, NOUPGRADE).

Allocation, which specifies:

- The volume(s) used for space allocation (DEFAULTVOLUMES, VOLUMES). FILE must be specified to provide data space extents only if UNIQUE is specified. **VOLUMES or DEFAULTVOLUMES can be specified at the alternate index level, data component level, or data and index component level.**
- The amount of space to be allocated (BLOCKS, CYLINDERS, RECORDS, TRACKS). **One of these parameters is required (unless MODEL is specified).**
- Whether an alternate index is to reside in either a unique or previously-defined data space (UNIQUE, SUBALLOCATION), or have no space allocated to it (NOALLOCATION).
- The particular class of space to be suballocated to the alternate index (USECLASS). See "Data Space Classification" in *VSE/VSAM Programmer's Reference* for more information.
- The percentage of free space to be distributed throughout the data component of an alternate index (FREESPACE).

- Whether the data is to be ranged by key across volumes (KEYRANGES) and, if so, whether the volumes are to be allocated in the order specified (ORDERED, UNORDERED).
- The space to be provided for buffers (BUFFERSPACE) and the size of control intervals (CONTROLINTERVALSIZE).
- The average and maximum lengths of data records in the alternate index (RECORDSIZE).

Protection and integrity, which specifies:

- The passwords to be associated with the alternate index or its components (MASTERPW, CONTROLPW, UPDATEPW, READPW).
- A prompting code (CODE) and the number of attempts allowed to provide the correct password in response to prompting at the operator's console (ATTEMPTS).
- A user-supplied authorization verification routine (AUTHORIZATION).
- That a user-supplied module is to be given control when an exception occurs during alternate-index processing (EXCEPTIONEXIT).
- The owner of the alternate index or component (OWNER).
- A retention period (FOR, TO) and whether the alternate index or data component is to be erased when its entry is deleted (ERASE, NOERASE).
- The share options to be associated with the alternate index or component (SHAREOPTIONS).
- Whether space is to be preformatted before data is initially loaded (RECOVERY, SPEED) and whether write-check operations are to be performed as records are inserted in the alternate index (WRITECHECK, NOWRITECHECK).

DEFINE ALTERNATEINDEX Parameters

ALTERNATEINDEX (*options*)
specifies that an alternate index is to be defined. ALTERNATEINDEX is followed by the parameters specified for the alternate index as a whole; they are optionally followed by the DATA and/or INDEX parameters and their subparameters.

Abbreviation: AIX

ATTEMPTS (*number*)
specifies the maximum number of times (0 through 7) 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 value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n'). This parameter only has an effect when the alternate index is password protected.

Abbreviation: ATT

Default: 2

AUTHORIZATION (*entrypoint* [*string*])
specifies that, in addition to passwords, you are supplying a USVR (user security verification routine) to check the authority of a processing pro-

gram to access your alternate index (file). VSAM transfers control to the USVR only after the program trying to open the file gives a correct password other than the master password. (The USVR is always bypassed whenever a correct master password is specified.) See *VSE/VSAM Programmer's Reference* for more information.

entrypoint – specifies the entry point of the USVR. The entry point may contain one through eight alphameric, national (@, #, and $), or special (hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or a national character.

string – specifies up to 255 characters that are to be passed to the USVR when it receives control to verify authorization. The string must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a string must be coded as two single quotation marks if the string is enclosed in single quotation marks. If the string is specified in hexadecimal, it must be preceded by X and enclosed in single quotation marks (X'string').

Abbreviation: AUTH

BLOCKS (*primary* [*secondary*])

specifies, *for FBA devices only*, the number of blocks to be allocated to the alternate index. Each block is a fixed size of 512 bytes. This parameter can be specified at either the alternate index level, the data component level, or at both the data and index component levels. If you specify secondary allocation at the data component level, you should also specify it at the index component level.

primary [secondary] – specifies the number of blocks to be made available for primary and secondary allocation. If secondary is specified, space for a component can be expanded to include a maximum of (1) 123 extents or (2) 16 extents per volume for a reuseable file. For a component with the UNIQUE attribute, 16 extents are allowed per volume. Primary and secondary can be specified in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

If you specify a primary or secondary value that does not coincide with a min-CA (track) boundary, VSAM will round it up to a value that equals a min-CA boundary. For example, if you specify BLOCKS (40 20) for the 3310, VSAM will round 40 up to 64 and 20 up to 32.

A further consideration applies if you specify the UNIQUE parameter in the DEFINE ALTERNATEINDEX command. Data space is acquired concurrently with the DEFINE command via the 'beginning block' and 'number of blocks' specification in the EXTENT statement. If the beginning block does not coincide with a min-CA boundary, VSAM will round it up to the next min-CA boundary; if the ending block (beginning block plus number of blocks) does not coincide with a min-CA boundary, VSAM will round it down to the previous min-CA boundary.

For example, if you specify 40 for the beginning block and 100 for the number of blocks for the 3310, VSAM rounds 40 up to 64 blocks and 139 (40 + 100 - 1 = 139) down to 127. (The beginning block is 64: the ending block is 127.) Be aware that in some instances, after VSAM rounds the values you specified, zero space is the resultant allocation. For example, if you specify 40 as the "beginning block" value and 50 as the "number of blocks" value, VSAM will round 40 up to 64 as before. However when VSAM rounds the ending block value of 89 (40 + 50 - 1 = 89) down to the closest min-CA boundary, the value (63) is less than the beginning block value. Therefore, no data space is available for allocation. (Beginning block is 64: the ending block is 63.)

TRACKS or CYLINDERS cannot be specified for FBA devices; the RECORDS parameter is valid for FBA devices. For further information on min-CAs and FBA devices see "Min-CA, Max-CA" in *VSE/VSAM Programmer's Reference.*

Abbreviations: BLK or BLOCK

BUFFERSPACE (*size*)

specifies the minimum space to be provided for buffers. Do not specify BUFFERSPACE at both the alternate index and the data component levels. See "I/O Buffer Space" in *VSE/VSAM Programmer's Reference* for more information.

size – is the number of bytes to be provided for buffers. This value 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 one index component control interval. If you specify an insufficient value, the command terminates.

Abbreviations: BUFSPC or BUFSP

Default: If BUFFERSPACE is omitted, a default value is set in the catalog. This default value, the minimum amount of buffer space allowed by VSAM, is enough space for two data component control intervals and one index component control interval.

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

identifies the catalog that contains the entry of the base cluster named in the RELATE parameter. You do not have to specify this parameter (unless it is needed for password specification) when the entry is to be defined in the default catalog. (The default catalog is: (1) the job catalog, if one is being used, or (2) the master catalog.) The alternate index is always cataloged in the same catalog as its base cluster.

catname – is the name of the catalog.

password – specifies the catalog's update or higher-level password. If no password is specified and the catalog is password protected, VSAM asks the operator for the correct password. If the base cluster's master password is not specified with the RELATE parameter, then the catalog's master password must be specified.

Abbreviation: CAT

CODE (*code*)

specifies a code name for the alternate index or component. If an attempt is made to access the alternate index without a password, the code name is used in a prompting message to the operator rather than the name of the alternate index. The operator can then provide the correct password.

The code may contain from one through eight EBCDIC characters. The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. If a single quotation mark appears within the code, it must be coded as two single quotation marks. Code may also be specified in hexadecimal (X'code') form.

If the alternate index or the component is password protected and CODE is not specified and an attempt is made to access the alternate index or component without supplying a password, the operator is prompted with

the name of the alternate index or component. The operator can then provide the correct password. If the cluster's master password is not specified with the RELATE parameter, then the catalog's master password must be specified.

CONTROLINTERVALSIZE(*size*)

specifies, in number of bytes, the size of the control intervals for the alternate index or the component. You can specify this value in decimal (size), hexadecimal (X'size'), or binary (B'size') form. Note that it is usually undesirable to specify CONTROLINTERVALSIZE at the alternate index level because *size* will then apply to both the data and index components. Instead, specify the control interval size at the data component level and either specify a size at the index component level or better yet, let VSAM choose the index component size for you (which is based on the data component control interval size, control area size, and key length).

The size of a control interval for a *data component* ranges from 512 to 32,768 bytes; you must use a multiple of 512 when the size is between 512 and 8192 bytes, and a multiple of 2048 when the size is between 8192 and 32,768 bytes. In either case, the control interval size must be at least ten bytes greater than the average record size. The size of a control interval for an *index component* can be any multiple of 512, up to 8192 bytes. If you code an improper multiple for size, VSAM selects the next higher multiple.

If CONTROLINTERVALSIZE is not coded, VSAM determines the size of control intervals as follows: if you specified a value for RECORDSIZE and the size of your data records permits, VSAM uses 2048 for the data component size and 512 for the index component size. If you did not specify RECORDSIZE, VSAM uses 4096 for the data component size.

A file with a data component *physical-record size*other than .5, 1, 2,or 4K, or an index component control interval size other than .5, 1, 2, or 4K cannot be directly processed by OS/VS; however, file portability between VSE and OS/VS can be retained via the EXPORT/IMPORT (or EXPORTRA/IMPORTRA) commands.

See "Control Interval Size" in *VSE/VSAM Programmer's Reference* for additional information on control intervals and physical-record sizes.

Abbreviation: CNVSZ or CISZ

CONTROLPW(*password*)

specifies a control password for the alternate index or component. The control password permits read and write operations using control interval access and all operations permitted by the update and read passwords. See "Passwords to Authorize Access" in the *VSE/VSAM Programmer's Reference* for more information.

password – is a one through 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. Password may also be coded in hexadecimal form (X'password').

Abbreviation: CTLPW

CYLINDERS (*primary* [*secondary*])

specifies the number of cylinders to be allocated. Either this parameter or the BLOCKS, TRACKS, or RECORDS parameter **must be specified** (unless you specified the MODEL parameter) whenever you are defining an alternate index—even when the alternate index has the attribute UNIQUE. Do not specify CYLINDERS or TRACKS for FBA devices or BLOCKS for CKD devices. This parameter can be specified at either the alternate index level, the data component level, or at both the data component and index component levels. If secondary allocation is provided at the data component level, it should also be provided at the index component level.

The actual allocation is rounded upward to the nearest control area boundary. The control area size is equal to either 1) the lesser value of the primary or secondary allocation, or 2) the primary allocation value in the case where the secondary allocation is zero. In either case, the control area cannot exceed one cylinder.

primary [*secondary*] – specifies the number of cylinders available for primary and secondary allocation. The amount(s) can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. If secondary is specified, space for a component can be expanded to include a maximum of 123 extents or 16 extents per volume for a reusable (dynamic) file. For a component with the UNIQUE attribute, 16 extents are allowed per volume.

If KEYRANGES is not specified and the data component is to be contained on more than one volume, the *primary* value given above is allocated on one volume only (when the component is defined). When the component increases in size (due to secondary allocations) so as to extend on to additional volumes, the first allocation on each overflow volume is the *primary* value. If KEYRANGES is specified and the data component is to be contained on more than one volume, the *primary* value is immediately allocated on each volume required for the key ranges.

The *secondary* value is allocated on all volumes on an as-needed basis regardless of the specification of KEYRANGES.

Abbreviation: CYL

DATA (*options*)

specifies attributes of the data component of the alternate index; if these attributes are specified for the alternate index as a whole, they are overridden by the options specified in the DATA parameter. If a name is not specified for the data component, a name is generated and listed for it.

DEFAULTVOLUMES (see VOLUMES)

ERASE | NOERASE

specifies whether the data component being defined is to be erased (overwritten with binary zeros) when its entry in the catalog is deleted.

Abbreviations: ERAS and NERAS

Default: NOERASE

EXCEPTIONEXIT(*mname*)
specifies the name of the user module (i.e., the phase name in the core image library) to be given control when an exception occurs during processing of the alternate index or its components. An exception is any condition which causes a SYNAD exit to be taken. The user exception exit is taken before the SYNAD exit, if both are specified.

An abnormal termination occurs if the exception exit is to be loaded but cannot be because it is not in the core image library. If it is in the core image library but the load operation fails because of insufficient virtual storage, VSAM ignores the exception exit. The SYNAD exit, if specified, is taken independently.

Abbreviation: EEXT

FILE(*dname*)
specifies the *filename* of the DLBL job control statement that, together with an EXTENT statement, identifies the volumes to be used for space allocation. You must specify the FILE parameter only if you specify UNIQUE. The pertinent volumes must be mounted during the define operation.

A FILE parameter must be specified for each unique component, individually or on the alternate index level. If both components of an alternate index are unique and reside on separate volumes, a single FILE parameter may be specified at the alternate index level. This case requires one DLBL statement with EXTENT statements for each of the two components. You must specify allocation information in the EXTENT statement(s) that corresponds to a unique component.

If both components are unique and reside on the same volume, FILE must be specified for both DATA and INDEX. In this case the components cannot share the same DLBL/EXTENT statements. Note that the symbolic unit on the EXTENT statement is not required.

Note: This parameter explanation applies to the job control simplification introduced in Release 2. If you are using Release 1 job control, refer to the appropriate explanation in Appendix J.

FOR(*days*) | TO(*date*)
specifies the retention period for the alternate index. If neither TO nor FOR is specified, the alternate index can be deleted at any time.

FOR(*days*)
specifies the number of days for which the alternate index is to be kept. If the number specified is 0 through 1830, the alternate index is retained for the number of days specified; if the number is 1831 through 9999 (maximum), the alternate index is retained through the year 1999. This value can be expressed in decimal (n), hexadecimal (X'n'), or binary B('n') form.

TO(*date*)
specifies the date until which the alternate index is to be kept. The date has the form yyddd, where yy (00 through 99) is the year and ddd (001 through 366) is the day.

FREESPACE (*cipercent* [ƀ *capercent*])

specifies the amount of space that is to be left empty after the initial load or any extension and after any split of control intervals (cipercent) or control areas (capercent) during sequential processing. The empty space is available for data records that are subsequently updated and inserted into the file. The amounts are specified as percentages. The percentages, which must be equal to or less than 100, may be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n'). However, even if 100% is specified, space for at least one record in a control interval and one control interval in a control area is provided.

Abbreviation: FSPC

Default: 0 percent.

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, the sequence-set record for each control area is written as many times as it will fit on the first track (min-CA) of the control area. If the control area is less than one cylinder (max-CA), inefficient utilization of file space can result.

Abbreviations: IMBD and NIMBD

Default: NOIMBED

INDEX (*options*)

specifies attributes of the index component of the alternate index; if these attributes are specified for the alternate index as a whole, they are overridden by the INDEX parameter. If a name is not specified for the index component, a name is generated and listed for the index component.

Abbreviation: IX

KEYRANGES ((*lowkey*ƀ *highkey*) [ƀ (*lowkey*ƀ *highkey*) . . .])

specifies that portions of the alternate index's data component are to be placed on different volumes. Each portion of the alternate index 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 - you are not allowed to insert records within the gap. See ***VSE/VSAM Programmer's Reference*** for more information.

The KEYRANGES parameter interacts with other DEFINE ALTERNATEINDEX parameters. You should take care to insure 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 on its own volume in its own VSAM data space. 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 cannot 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 keylength specified in the KEYS parameter.

The values lowkey and highkey can be 1 to 64 characters long or, if coded in hexadecimal, 1 to 128 characters long. All EBCDIC characters are allowed. Keys consisting of EBCDIC 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 coded (X'key'). See *VSE/VSAM Programmer's Reference* for more information.

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.

KEYRANGES cannot be specified when REUSE is specified.

Abbreviation: KRNG

KEYS (*length*ƀ*offset* | 64 0)
specifies information about the alternate key-field in the base cluster's data record.

*length*ƀ*offset* – specifies the length of the key-field in bytes (between 1 and 255), and its displacement (offset) from the beginning of the base cluster's data record. The sum of length and offset cannot exceed the length of the shortest record nor the length of the first segment of a spanned record. These values can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Default: 64 0

MASTERPW (*password*)
specifies a master password for the alternate index or component. If MASTERPW is not specified and if other passwords exist, the highest-level existing password automatically becomes the master password. The master password allows all operations. See CONTROLPW for the definition of password.

Abbreviation: MRPW

MODEL (*entryname* [/ *password*]
[ƀ *catname* [/ *password*]])

specifies that the catalog entry of an already-defined alternate index is to be used as a model for the entry being built. The already-defined alternate index can have space allocated to it or not. See "How to Use One Object as a Model for Another Object and Override System Defaults" in *VSE/VSAM Programmer's Reference* for more information about MODEL.

entryname – specifies the name of the entry to be used as a model. The entry to be used as a model must be of the same entry type as the entry being built. For example, if you are using this parameter at the data component level, the object used as a model must be a data component.

password – specifies a password. If the alternate index or component whose entry is to be used as a model is password protected and is cataloged in a password protected catalog, either the alternate index's or component's password or its catalog's password is required. If both the entry password and the catalog password are specified, the catalog password will be used. If the protection attributes are to be copied, give the master password of either the model (following *entryname*) or the catalog (following *catname*). If the model's passwords are not to be copied, any password of either the model or its catalog can be used.

catname – specifies the name of the catalog that contains the entry to be used as a model. This parameter is required if (1) you are going to specify the password of the catalog that defines the alternate index or component instead of specifying the password of the alternate index or component itself, or (2) the catalog is not the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

Restrictions: Your job may be terminated because of allocation problems if you use the MODEL parameter and (1) specify a device type different from that specified for the model through the VOLUMES parameter, (2) change the length of keys through the KEYS parameter, (3) change the size of records, buffer space, or control intervals through the RECORDSIZE, BUFFERSPACE, or CONTROLINTERVALSIZE parameter, (4) change from UNIQUE to SUBALLOCATION, or vice versa, or (5) change the unit of allocation through the TRACKS, BLOCKS, CYLINDERS, or RECORDS parameter.

If modeling causes the alternate index or component to be defined as unique, FILE must also be specified.

NAME (*entryname*)

specifies the name of the alternate index or component. **NAME must be specified for the alternate index**; it can optionally be specified for a data or index component. If no name is specified for a component, a name is generated and listed for you. If components are not named, you are obliged to give, in a subsequent ALTER command, the long system-generated name. (This may be enough reason for you to specify the component name.) Because an alternate index, data component, and index component are individually named, they can be processed individually.

The name may contain from 1 through 44 alphameric characters, national characters (@, #, and $), and special characters (hyphen and 12-0 overpunch). Names containing more than eight characters must be segment-

ed by periods; one through eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or a national character.

NOALLOCATION
indicates that no space allocation is to take place for this define. This allows you to create (a) a *dynamic file* (*alternate index*), (b) a *model* (*default list of parameters*) that can be used for the future definitions of all alternate indexes, or (c) a *model* that can be used for the future definition of one or more specific alternate indexes. See *VSE/VSAM Programmer's Reference* for more information on dynamic files and modeling.

- You create a *dynamic file* (*alternate index*) by specifying the NOALLOCATION parameter together with REUSE. No space is suballocated to the dynamic file at define time, rather, the required space (specified with a space allocation parameter at define time) is suballocated to the file when VSAM opens it.

  When a dynamic file is closed, the file is reset to empty (high-used RBA set to 0, and all space is deallocated) if (1) the ACB specifies (a) PARMS=(CLOSDSP=DELETE) or (b) PARMS=(CLOSDSP=DATE) and the expiration date has been reached, or (2) the DLBL statement specifies DISP=(,DELETE) or DISP=(,DATE) and the expiration date has been reached.

- You create a *default model for the future definitions of all alternate indexes* by specifying in this DEFINE command (a) the NOALLOCATION parameter (b) a reserved entryname (DEFAULT.MODEL.AIX) and (c) the specific list of parameters that you want the model to have. The parameters in this model can be viewed as a system default because they effectively override the actual system default when the parameter is not explicitly specified.

  No data space is ever occupied by this type of model, it exists solely as an entry in a catalog. This also applies to any space allocation parameters you specify with NOALLOCATION, VSAM uses them for space allocation in subsequent defines.

- You create a *model* that can be explicitly specified via the MODEL parameter for one or more specific alternate indexes by specifying in the DEFINE command (a) the NOALLOCATION parameter (b) an ordinary *entryname*, and (c) the specific parameters to be included in the model.

  This differs from the preceding case because here in this case you can have more than one model for alternate indexes. You use this type of model by specifying its *entryname* (instead of a *reserved* entryname) in the MODEL parameter of a subsequent DEFINE ALTERNATEINDEX command.

In the case of default models, the BUFFERSPACE, RECORDSIZE, CONTROLINTERVALSIZE, RELATE, or the 4 password parameters (READPW, etc.) are not modeled.

If you specify NOALLOCATION for the data component of an alternate index, you must also specify it for the index component; you can, and normally would, specify it only at the alternate index level in which case it propagates to both the data and index component levels.

Abbreviation: NAL

NOERASE (see ERASE)

NOIMBED (see IMBED)

NONUNIQUEKEY (see UNIQUEKEY)

NOREPLICATE (see REPLICATE)

NOREUSE (see REUSE)

NOUPGRADE (see UPGRADE)

NOWRITECHECK (see WRITECHECK)

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, DEFINE processing is terminated.

Abbreviations: ORD and UNORD

Default: UNORDERED

OWNER (*ownerid*)
identifies the owner of the alternate index. The *ownerid* may contain one through 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 also be coded in hexadecimal form. If specified in hexadecimal, it must be coded (X'ownerid').

READPW (*password*)
specifies a read password for the alternate index or component. The read password permits read operations. See CONTROLPW for the definition of password.

Abbreviation: RDPW

RECORDS (*primary* [*secondary*])
specifies the number of records for which space is to be allocated. If RECORDS is specified at the alternate index or data component level, the space must be large enough to include the index records or the file will not hold the expected number of data records.

See the CYLINDERS parameter for the information and restrictions that apply to CKD devices. See the BLOCKS parameter for information on FBA devices.

Abbreviation: REC

RECORDSIZE(*average*ƀ *maximum*)

specifies the average and maximum lengths, in bytes, of the data record of an alternate index. RECORDSIZE (*average*) must account for five bytes of control information plus the length of the alternate key plus the length of at least one base cluster pointer (the length of a base cluster pointer is the length of the prime key for a key-sequenced file or the length of an RBA (always four bytes) for an entry-sequenced file). If NONUNIQUEKEY is also used, RECORDSIZE must account for the increased record size resulting from the multiple prime key or RBA pointers in the alternate index data record.

These values can be expressed in decimal (n), hexadecimal (X'n'), or binary(B'n'). The maximum record size cannot exceed control area size minus 10 bytes of control information for each control interval contained in the control area.

Abbreviation: RECSZ

Default: The default values 4086 and 32600 are used for the average and maximum record sizes. Default value 4086 overrides the data component CONTROLINTERVALSIZE default value of 2048.

RECOVERY | SPEED

specifies whether control areas allocated to the data component are to be preformatted before alternate-index records are loaded into them. The RECOVERY and SPEED parameters apply only to initial loading.

When RECOVERY is specified or defaulted to, 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, your program may be able to 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). The BLDINDEX, IMPORT, IMPORTRA, and REPRO command do not provide the resume-loading capability.

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 or default to 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.

You cannot use the VERIFY command to recover from a system failure that occured during the initial load of a file unless you specify or default to RECOVERY.

Abbreviation: RCVY

Default: RECOVERY

RELATE (*entryname* [*/password*])

names the alternate index's *base cluster.* The base cluster is an entry-sequenced file or a key-sequenced file that the alternate index is to be organized over. **This parameter is required except when you are defining a default model.**

entryname – names the base cluster - it must not refer to a relative record file or to a cluster with the REUSE or NOALLOCATION attribute .

password – specifies the base cluster's master password. If the base cluster is password-protected, its master password is required. Alternatively the master password of the catalog in which the base cluster is defined may be specified in the CATALOG parameter.

Abbreviation: REL

REPLICATE | NOREPLICATE

specifies whether each index record is to be written on a track (min-CA) as many times as it will fit to reduce rotational delay and improve performance. The sequence set is implicitly replicated if IMBED is specified, even if REPLICATE is not specified.

Abbreviations: REPL and NREPL

Default: NOREPLICATE

REUSE | NOREUSE

specifies whether the alternate index can be opened repetitively as a new file, that is, with its high-used RBA set to 0. When a reusable alternate index is opened for output, its high-used RBA is set to 0 if you open it with (1) an ACB which specifies MACRF=RST or (2) a DLBL statement that specifies DISP=NEW. When you use the Access Method Services BLDINDEX command to rebuild a reusable alternate index, the high-used RBA is always reset to 0 when the alternate index is opened by BLDINDEX.

When a reusable alternate index is closed, its high-used RBA is set to 0 if (1) the ACB specifies (a) PARMS=(CLOSDSP=DELETE) or (b) PARMS=(CLOSDSP=DATE) and the expiration date has been reached, or (2) the DLBL statement specifies DISP=DELETE or DISP=DATE and the expiration date has been reached.

Reusable files can be multi-volumed and are restricted to a maximum of 16 physical extents per volume. If a base cluster is defined as reusable, it must not have alternate indexes associated with it; however, it is permissible to define reusable alternate indexes that are related to a nonreusable base cluster.

REUSE must not be specified together with UNIQUE or KEYRANGES.

Abbreviations: RUS and NRUS

Default: NOREUSE

SHAREOPTIONS (*value* [ↄ *reserved*])

specifies how a file (data or index component of an alternate index) can be shared within one or more VSE systems. Files that are shared across VSE systems must reside on a shared DASD device (this excludes the 23xx series of devices; they can share files within one system only). A file

cannot be deleted, reset, or deallocated, or its share option value altered, if it is currently open for another program regardless of the sharing option value specified.

During the initial load of a file (and irrespective of the share-option value specified) VSAM treats the share-option specification as if it were SHAREOPTIONS (1). After the file is loaded and successfully closed, VSAM uses the original share-option value.

value – specifies the degree of file sharing allowed on one or more VSE systems. If a file cannot be shared for the type of sharing you specify, your request to open it is denied. The values that can be specified are:

1
specifies that the file being defined can be opened by any number of users for input processing; once the file is opened for input it cannot be opened for output processing. Conversely once the file has been opened for output processing, no other user can open it for input or output until that (the first) output processing has been completed. This option ensures full read and write integrity. The default value is 1.

2
specifies that the file being defined can be opened by any number of users for input processing even if one user is using it for output processing. No more than one user can open the file for output at one time. This option ensures write integrity only, because the file might be modified while records are being retrieved from it. Therefore, each user must ensure his own file's read integrity.

3
specifies that the file can be opened by any number of users for both input and output processing. Except for initial load processing, VSAM does nothing to ensure read or write integrity; it only ensures that the opened file is not deleted or reset, and that its share option value is not altered.

4
specifies that the file can be opened by any number of users for input processing. At the same time the file can be opened by one or more users on a *single* system for output processing. The system that gains exclusive control of the file for output processing will be the one that first issues a request for output processing. VSAM ensures write integrity each time a record is updated or inserted as in share option 2.

You can ensure read integrity by retrieving records with the update option; if you do not use the update option, some records in control intervals being updated concurrently by more than one program might be missed or skipped by VSAM because each program might retrieve a different copy of the control interval.

reserved – for OS/VS compatibility only. The valid OS/VS values for this subparameter are 3 and 4. This parameter refers to cross system sharing in OS/VS; it has no effect in VSE except that VSAM stores its value in the catalog. (This subparameter takes effect in OS/VS in those cases where the file is imported into an OS/VS system.) The default value is 3.

Abbreviation: SHR

SPEED (see RECOVERY)

SUBALLOCATION (see UNIQUE)

TO (see FOR)

TRACKS (*primary* [*secondary*])
specifies the number of tracks to be allocated. If the space is allocated to an alternate index with the UNIQUE attribute, the space is rounded up to the nearest cylinder. If you specify a number equal to or greater than the number of tracks per cylinder, then the allocation is rounded in terms of the number of cylinders needed to contain the tracks specified. See the CYLINDERS parameter for more information and restrictions.

Abbreviation: TRK

UNIQUE | SUBALLOCATION | NOALLOCATION
specifies whether the alternate index's components are allocated space of their own, whether a portion of previously-defined VSAM data space is to be used for each component, or whether no space is to be allocated to the alternate index (see NOALLOCATION).

UNIQUE cannot be specified together with REUSE. You cannot define a unique alternate index in a volume that does not already contain a VSAM data space if the catalog entry of the unique alternate index is to be in a recoverable catalog (a CRA must exist on the volume before you define a unique alternate index).

If SUBALLOCATION is specified, a data space must exist on the volume on which the alternate index's components are to reside. The name of the data space, not of the component, appears in the VTOC.

If UNIQUE is specified:

- The alternate index's components are allocated space of their own and their names appear in the VTOC of the volume(s) under their own names.
- For the data component and the data component resides on a 3330 volume, the index component must reside on a 3330 volume.
- Each volume specified through the VOLUMES parameter must have at least one and no more than 16 EXTENT statements.
- For CKD devices, data and index component extents must begin on cylinder boundaries.
- A FILE parameter must be specified for each unique component individually or on the alternate index level.
- And KEYRANGES is also specified, each key range must be on a different volume and there cannot be fewer volumes specified through the VOLUMES parameter than there are key-range pairs specified through KEYRANGES.

Abbreviations: UNQ and SUBAL

Default: SUBALLOCATION

UNIQUEKEY | NONUNIQUEKEY

specifies whether the alternate key-value can be found in one or more than one of the base cluster's data records.

UNIQUEKEY indicates that each alternate key is in one and only one data record in the base cluster. When you are building the alternate index via the BLDINDEX command, Access Method Services issues a warning message if multiple records are found with the same alternate key. All but the first multiple record are ignored.

NONUNIQUEKEY indicates that an alternate key can be in more than one data record in the base cluster. If you specify NONUNIQUEKEY, then RECORDSIZE must account for the increased record size resulting from the multiple prime key or RBA pointers in the alternate index data record.

Abbreviations: UNQK and NUNQK

Default: NONUNIQUEKEY

UNORDERED (see ORDERED)

UPDATEPW (*password*)

specifies an update password for the alternate index or component. The update password permits read and write operations. See CONTROLPW for the definition of password.

Abbreviation: UPDPW

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.

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.

If you specify UNIQUEKEY and a duplicate key condition is detected during alternate index upgrading, VSAM issues a return code and backs out any changes already made to other alternate indexes in the upgrade set.

Abbreviations: UPG and NUPG

Default: UPGRADE

USECLASS (*primary* [*secondary*])

specifies the class of data space to be occupied by a nonunique (suballocated) alternate index or component. For the *primary allocation* of space:

- 0 specifies that the alternate index or component is to occupy class 0 data space. Data spaces defined under OS/VS and DOS/VS Release 34 (and prior releases) are class 0 data spaces. 0 is the default space classification.
- 1 specifies that the alternate index or component is to occupy class 1

data space. You are urged, for consistency, to use class 1 for fixed-head areas of direct-access storage.

2-7 specifies that the alternate index or component is to occupy the specified class of data space (user-defined).

For the *secondary allocation* of space:

0 specifies class 0 data space is to be used by the alternate index or component.

P specifies that the primary space classification is to be used. P is the default.

Whenever the alternate index or component increases in size so as to extend on to additional volumes, the first suballocation on each overflow volume is directed to the class of space specified by the primary subparameter. All subsequent allocations on each overflow volume use the class of space that is specified by the secondary subparameter. If the alternate index or component is extended after you IMPORT CONNECT to OS/VS or DOS/VS Release 34 (or earlier), no USECLASS assignments are made (the alternate index or component is extended into any available data space because class specifications apply to VSE/VSAM only). The alternate index or component is still processable under VSE/VSAM, but the LISTCAT output may be inaccurate.

If you assign a non-zero USECLASS value to a unique alternate index or component, the DEFINE will terminate.

If USECLASS is specified only at the alternate index level, it is applied also to the data and index component levels. If USECLASS is specified at the data level only, it is applied also to the index level. If you want different classifications assigned to the data and index components of an alternate index you must specify (or default) different values at both the data and index levels.

Abbreviation: USCL

VOLUMES (*volser* [ъ *volser* . . .] |) DEFAULTVOLUMES

specifies the volumes to contain the alternate index or components.

VOLUMES

can be specified as a parameter of ALTERNATEINDEX or DATA, or of both DATA and INDEX. If you want the data and index components to reside on different volumes, you must specify the VOLUMES parameter together with both DATA and INDEX. If more than one volume is specified with a single VOLUMES parameter, they must be of the same device type, except if you are establishing (defining) a default model (you can specify different device types in the model).

You can specify up to 123 volumes for each component. A volume serial number may contain one to six alphameric, national (@,#, and $) and special characters (commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs). Single quotation marks within a volume serial number must be coded as two single quotation marks. A volume serial number must be enclosed in single quotation marks if it contains a special character. For consistency with VSE job control, only alphameric characters should be used.

DEFAULTVOLUMES
is used in conjunction with the MODEL parameter to indicate that VSAM is to select the volume(s) it needs from a list of volume(s) established in a default model instead of from volumes specified in the named model or current define (see "Default Volumes" in *VSE/VSAM Programmer's Reference* for more information.) A default model of the correct type is required to be in the relevent catalog.

An error occurs if you specify DEFAULTVOLUMES and VOLUMES together at the same level. You can specify DEFAULTVOLUMES at any level and in any combination of levels; if specified at the alternate index level, DEFAULTVOLUMES propagates to the data and index component levels. If DEFAULTVOLUMES propagates to the data and index component levels where a VOLUMES parameter has been specified, the VOLUMES parameter takes precedence.

If DEFAULTVOLUMES is specified (or defaulted to) at any level with the ORDERED or UNIQUE parameters, VSAM indicates an error.

Abbreviations: VOL and DFVOL

Default: DEFAULTVOLUMES

WRITECHECK | NOWRITECHECK
specifies whether to check the data transfer of records written in the alternate index. If WRITECHECK is specified, a record is written and then read, without data transfer, to test for the data check condition. If NOWRITECHECK is specified, a record is written, but no checking occurs.

Abbreviations: WCK and NWCK

Default: NOWRITECHECK

DEFINE CLUSTER

The DEFINE CLUSTER command (DEF CL) is used to define key-sequenced, entry-sequenced, or relative-record files. For a key-sequenced file, three entries are created in the catalog: an entry for the cluster, its data component, and its index component. For an entry-sequenced or a relative-record file, two entries are created: one for the cluster and one for its data component. See "Defining a VSAM File" and Examples 3 and 4 in "Appendix A: Sample Job Streams" for examples and more information.

The format of the DEFINE CLUSTER command is shown below. Note that the organization of the command consists of three groupings. These groupings are referred to as *levels* in this book; note that some parameters appear in more than one level. (The optional CATALOG parameter stands alone; it does not belong to any level.)

Cluster Level	
DEFINE	CLUSTER (NAME(*entryname*) [ATTEMPTS(*number*)] [AUTHORIZATION(*entrypoint* [Þ *string*])] [BUFFERSPACE(*size*)] [CODE(*code*)] [CONTROLINTERVALSIZE(*size*)] [CONTROLPW(*password*)] [CYLINDERS(*primary* [Þ *secondary*]) | † BLOCKS(*primary* [Þ *secondary*]) | † RECORDS(*primary* [Þ *secondary*]) | † TRACKS(*primary* [Þ *secondary*])] † [ERASE|NOERASE] [EXCEPTIONEXIT(*mname*)] [FILE(*dname*)] † [FOR(*days*)|TO(*date*)] [FREESPACE(*cipercent* [Þ *capercent*])] [IMBED|NOIMBED] [INDEXED|NONINDEXED|NUMBERED] [KEYRANGES((*lowkey*Þ *highkey*) [Þ(*lowkey*Þ *highkey*)...])] [KEYS(*length*Þ *offset*)] [MASTERPW(*password*)] [MODEL(*entryname* [/ *password*] [Þ *catname* [/ *password*] [Þ *dname*]])] [ORDERED|UNORDERED]

† This parameter is required under certain conditions. See the following parameter explanations or Appendix E for additional information.

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

(continued on next page)

Cluster Level (continued)

```
    [OWNER ( ownerid ) ]
    [READPW ( password ) ]
    [RECORDFORMAT ( format ) ]
    [RECORDSIZE ( average b maximum ) ]
    [RECOVERY | SPEED]
    [REPLICATE | NOREPLICATE]
    [REUSE | NOREUSE]
    [SHAREOPTIONS ( value [ b reserved ] ) ]
    [SPANNED | NONSPANNED]
    [UNIQUE | SUBALLOCATION | NOALLOCATION]
    [UPDATEPW ( password ) ]
    [USECLASS ( primary [ b secondary ] ) ]
    [VOLUMES ( volser [ b volser . . . ] ) |
      DEFAULTVOLUMES]
    [WRITECHECK | NOWRITECHECK]
)
```

Data Component Level

```
[DATA
(
    [ATTEMPTS ( number ) ]
    [AUTHORIZATION ( entrypoint [ b string ] ) ]
    [BUFFERSPACE ( size ) ]
    [CODE ( code ) ]
    [CONTROLINTERVALSIZE ( size ) ]
    [CONTROLPW ( password ) ]
    [CYLINDERS ( primary [ b secondary ] ) |          †
      BLOCKS ( primary [ b secondary ] ) |            †
      RECORDS ( primary [ b secondary ] ) |           †
      TRACKS ( primary [ b secondary ] ) ]            †
    [ERASE | NOERASE]
    [EXCEPTIONEXIT ( mname ) ]
    [FILE ( dname ) ]                                 †
    [FREESPACE ( cipercent [ b capercent ] ) ]
    [KEYRANGES ( ( lowkey b highkey )
      [ b ( lowkey b highkey ) . . . ] ) ]
    [KEYS ( length b offset ) ]
    [MASTERPW ( password ) ]
    [MODEL ( entryname [ / password ]
      [ b catname [ / password ] [ b dname ] ] ) ]
    [NAME ( entryname ) ]
    [ORDERED | UNORDERED]
    [OWNER ( ownerid ) ]
    [READPW ( password ) ]
    [RECORDSIZE ( average b maximum ) ]
```

† This parameter is required under certain conditions. See the following parameter explanations or Appendix E for additional information.

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

(continued on next page)

Data Component Level (continued)

```
   [RECOVERY|SPEED]
   [REUSE|NOREUSE]
   [SHAREOPTIONS( value [ reserved ] ) ]
   [SPANNED|NONSPANNED]
   [UNIQUE|SUBALLOCATION|NOALLOCATION]
   [UPDATEPW( password ) ]
   [USECLASS( primary [ secondary ] ) ]
   [VOLUMES( volser [ volser ... ] ) |
     DEFAULTVOLUMES]
   [WRITECHECK|NOWRITECHECK]
)]
```

Index Component Level

```
[INDEX
(
   [ATTEMPTS( number ) ]
   [AUTHORIZATION( entrypoint [ string ] ) ]
   [CODE( code ) ]
   [CONTROLINTERVALSIZE( size ) ]
   [CONTROLPW( password ) ]
   [CYLINDERS( primary [ secondary ] ) |          †
     BLOCKS( primary [ secondary ] ) |            †
     RECORDS( primary [ secondary ] ) |           †
     TRACKS( primary [ secondary ] ) ]            †
   [EXCEPTIONEXIT( mname ) ]
   [FILE( dname ) ]                               †
   [IMBED|NOIMBED]
   [MASTERPW( password ) ]
   [MODEL( entryname [ / password ]
     [ catname [ / password ] [ dname ] ] ) ]
   [NAME( entryname ) ]
   [ORDERED|UNORDERED]
   [OWNER( ownerid ) ]
   [READPW( password ) ]
   [REPLICATE|NOREPLICATE]
   [REUSE|NOREUSE]
   [SHAREOPTIONS( value [ reserved ] ) ]
   [UNIQUE|SUBALLOCATION|NOALLOCATION]
   [UPDATEPW( password ) ]
   [USECLASS( primary [ secondary ] ) ]
   [VOLUMES( volser [ volser ... ] ) |
     DEFAULTVOLUMES]
   [WRITECHECK|NOWRITECHECK]
)]
```

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

† This parameter is required under certain conditions. See the following parameter explanations or Appendix E for additional information.

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

DEFINE CLUSTER Parameters: Summary

The parameters of the DEF CL command can be divided into the following categories:

Name, which specifies:

- The name of the cluster (NAME) and, optionally, of its components. **This parameter is required.**
- The name of the catalog in which the cluster is to be defined (CATALOG).
- The name of the cluster entry to be used as a model for this cluster (MODEL). See *VSE/VSAM Programmer's Reference* for more information.

Data organization, which specifies:

- Whether the file is to be key-sequenced (INDEXED), entry-sequenced (NONINDEXED), or relative record (NUMBERED).
- Whether (for a key-sequenced file only) an index record is to be written as many times as it will fit on a track (REPLICATE, NOREPLICATE).
- Whether (for a key-sequenced file only) the index's sequence-set records (the lowest level of the index) are to be placed with the data component of the file (IMBED, NOIMBED). See "Optimizing VSAM's Performance" in *VSE/VSAM Programmer's Reference* for more information on the index options.
- The length and offset of the cluster's key-field (KEYS).
- Whether the cluster is to be reuseable (REUSE, NOREUSE).

Allocation, which specifies:

- The volume(s) for space allocation (DEFAULTVOLUMES, VOLUMES). FILE must be specified to provide data space extents only if UNIQUE is specified. **VOLUMES or DEFAULTVOLUMES can be specified at the cluster level, data component level, or data and index component level.**
- The amount of space to be allocated (TRACKS, CYLINDERS, BLOCKS, or RECORDS). **One of these parameters is required (unless MODEL is specified.**
- The particular class of space to be suballocated to the file (USECLASS). See "Data Space Classification" in *VSE/VSAM Programmer's Reference* for more information.
- For an indexed cluster, the percentage of free space to be distributed through the data component of the cluster (FREESPACE).
- Whether a cluster is to reside in a unique or previously-defined data space (UNIQUE, SUBALLOCATION), or have no space allocated to it (NOALLOCATION).
- For an indexed cluster, whether data is to be ranged by key across volumes (KEYRANGES) and, if so, whether the volumes are to be allocated in the order specified (ORDERED, UNORDERED).
- The space to be provided for buffers (BUFFERSPACE) and the size of control intervals (CONTROLINTERVALSIZE).
- The average and maximum lengths of data records (RECORDSIZE).

- Records that may be larger than a control interval and may span control intervals (SPANNED, NONSPANNED).

Protection and integrity, which specifies:

- The passwords to be associated with the cluster or its components (MASTERPW, CONTROLPW, UPDATEPW, READPW).
- A prompting code (CODE) and the number of attempts allowed to provide the correct password in response to prompting at the operator's console (ATTEMPTS).
- A user-supplied authorization verification routine (AUTHORIZATION).
- The owner of a cluster or its components (OWNER).
- A retention period (FOR, TO) and whether the cluster or data component is to be erased when its entry is deleted (ERASE, NOERASE).
- The share options to be associated with the cluster or its components (SHAREOPTIONS).
- Whether space is to be preformatted before data is initially loaded (RECOVERY, SPEED) and whether write-check operations are to be performed as records are inserted in the data or index components (WRITECHECK, NOWRITECHECK).
- That a user-supplied module is to be given control when an exception occurs during file processing (EXCEPTIONEXIT).

DEFINE CLUSTER Parameters

ATTEMPTS (*number*)

specifies the maximum number of times (0 through 7) 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 value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Abbreviation: ATT

Default: 2

AUTHORIZATION (*entrypoint* [*string*])

specifies that, in addition to passwords, you are supplying a USVR (user security verification routine) to check the authority of a processing program to access your file. VSAM transfers control to the USVR only after the program trying to open the file gives a correct password other than the master password. (The USVR is always bypassed when a correct master password is specified.) See *VSE/VSAM Programmer's Reference* for more information.

entrypoint – specifies the entry point name of the USVR. The entrypoint name may contain one through eight alphameric, national (@, #, $), or special (the hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or a national character.

string – specifies up to 255 characters that are to be passed to the USVR when it receives control to verify authorization. The string must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a string must be coded as two single quotation marks if the string is enclosed in single

quotation marks. If the string is specified in hexadecimal, it must be coded (X'string').

Abbreviation: AUTH

BLOCKS (*primary* [*secondary*])
specifies, *for FBA devices only*, the number of blocks to be allocated to the cluster. Each block is a fixed size of 512 bytes. This parameter can be specified at either the cluster level, the data component level, or at both the data and index component levels. If you specify secondary allocation at the data component level, you should also specify it at the index component level.

primary [*secondary*] – specifies the number of blocks to be made available for primary and secondary allocation. If secondary is specified, space for a component can be expanded to include a maximum of (1) 123 extents or (2) 16 extents per volume for a reuseable file. For a component with the UNIQUE attribute, 16 extents are allowed per volume. Primary and secondary can be specified in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

If you specify a primary or secondary value that does not coincide with a min-CA (track) boundary, VSAM will round it up to a value that equals a min-CA boundary. For example, if you specify BLOCKS (40 20) for the 3310, VSAM will round 40 up to 64 and 20 up to 32.

A further consideration applies if you specify the UNIQUE parameter in the DEFINE CLUSTER command. Data space is acquired concurrently with the DEFINE command via the 'beginning block' and 'number of blocks' specification in the EXTENT statement. If the beginning block does not coincide with a min-CA boundary, VSAM will round it up to the next min-CA boundary; if the ending block (beginning block plus number of blocks) does not coincide with a min-CA boundary, VSAM will round it down to the previous min-CA boundary.

For example, if you specify 40 for the beginning block and 100 for the number of blocks for the 3310, VSAM rounds 40 up to 64 blocks and 139 (40 + 100 - 1 = 139) down to 127. (The beginning block is 64: the ending block is 127.) Be aware that in some instances, after VSAM rounds the values you specified, zero space is the resultant allocation. For example, if you specify 40 as the 'beginning block' value and 50 as the 'number of blocks' value, VSAM will round 40 up to 64 as before. However when VSAM rounds the ending block value of 89 (40 + 50 - 1 = 89) down to the closest min-CA boundary, the value (63) is less than the beginning block value. Therefore no data space is available for allocation. (Beginning block is 64: the ending block is 63.)

TRACKS or CYLINDERS cannot be specified for FBA devices; the RECORDS parameter is valid for FBA devices. For further information on min-CAs and FBA devices see "Min-CA, Max-CA" in *VSE/VSAM Programmer's Reference.*

Abbreviations: BLK or BLOCK

BUFFERSPACE (*size*)
specifies the minimum space to be provided for buffers. Do not specify BUFFERSPACE at both the cluster and the data component levels. See

"I/O Buffer Space" in *VSE/VSAM Programmer's Reference* for more details.

size – is the number of bytes to be provided for buffers. This value 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 file is key-sequenced, one index control interval. If you specify an insufficient value, the command terminates.

Abbreviations: BUFSPC or BUFSP

Default: If BUFFERSPACE is omitted, a default value is set in the catalog. This default value, the minimum amount of buffer space allowed by VSAM, is enough space for two data component control intervals and, if the file is key-sequenced, one index component control interval.

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

identifies the catalog in which the cluster is to be defined. You do not have to specify this parameter (unless it is needed for password specification) when the cluster is to be defined in the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

catname – is the name of the catalog.

password – specifies the update or higher-level password. If no password is specified and the catalog is password protected, VSAM asks the operator for the correct password.

Abbreviation: CAT

CLUSTER (*options*)

specifies that a cluster is to be defined. CLUSTER is followed by the parameters specified for the cluster as a whole. They are optionally followed by the DATA and/or INDEX parameters and their subparameters.

Abbreviation: CL

CODE (*code*)

specifies a code name for the cluster or component. If an attempt is made to access the cluster without a password, the code name is used in a prompting message to the operator rather than the name of the cluster. The operator can then provide the correct password. The code may contain from one through eight EBCDIC characters. The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. If a single quotation mark appears within the code, it must be coded as two single quotation marks. Code may also be specified in hexadecimal (X'code') form.

If the cluster or the component is password protected and CODE is not specified and an attempt is made to access the cluster or component without supplying a password, the operator is prompted with the name of the cluster or component. The operator can then provide the correct password.

CONTROLINTERVALSIZE (*size*)

specifies, in number of bytes, the size of the control intervals for the cluster or the component. Note that it is usually undesirable to specify CONTROLINTERVALSIZE at the cluster level because *size* will then apply to both the data and index components. Instead, specify the control interval

size at the data component level and either specify a size at the index component level, or better yet, let VSAM choose the index component size for you (which is based on the data component control interval size, control area size, and key length). You can specify the value in decimal (size), hexadecimal (X'size'), or binary (B'size') form.

The size of a control interval for a *data component* ranges from 512 to 32,768 bytes; you must use a multiple of 512 when the size is between 512 and 8192 bytes, and a multiple of 2048 when the size is between 8192 and 32,768 bytes. For nonspanned files, the control interval size must be at least seven bytes greater than the maximum record size; for spanned files, the control interval size must be at least ten bytes greater than the average record size. The size of a control interval for an *index component* can be any multiple of 512 up to 8192 bytes. If you code an improper multiple for size, VSAM selects the next higher multiple.

If CONTROLINTERVALSIZE is not coded, VSAM determines the size of control intervals as follows: if you specified a value for RECORDSIZE and the size of your data records permits, VSAM uses 2048 for the data component size and 512 for the index component size. If you did not specify RECORDSIZE, VSAM uses 4096 for the data component size.

A file with a data component *physical-record size*other than .5, 1, 2, or 4K, or an index component control interval size other than .5, 1, 2, or 4K, cannot be directly processed by OS/VS; however, file portability between VSE and OS/VS can be retained via the EXPORT/IMPORT (or EXPORTRA/IMPORTRA) commands.

See "Control Interval Size" in *VSE/VSAM Programmer's Reference* for additional information on control intervals and physical-record sizes.

Abbreviations: CNVSZ or CISZ

CONTROLPW (*password*)
specifies a control password for the cluster or component. The control password permits opening the file for read and write operations using control interval access and all operations permitted by the update and read passwords. See "Passwords to Authorize Access" in *VSE/VSAM Programmer's Reference* for more information.

password – is a one through 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. Password can also be coded in hexadecimal (X'*password*') form.

Abbreviation: CTLPW

CYLINDERS (*primary* [ъ *secondary*])
specifies the number of cylinders to be allocated. Either this parameter or the BLOCKS, RECORDS or TRACKS parameter **must be specified** (unless you specified the MODEL parameter) whenever you are defining a cluster—even when the cluster has the attribute UNIQUE. Do not specify CYLINDERS or TRACKS for FBA devices or BLOCKS for CKD devices. This parameter can be specified at either the cluster level, the data component level, or at both the data component and index component levels. If secondary allocation is provided at the data component level, it should also be provided at the index component level.

The actual allocation is rounded upward to the nearest control area boundary. The control area size is equal to either (1) the lesser value of the primary or secondary allocation, or (2) the primary allocation value in the case where the secondary allocation is zero. In either case, the control area cannot exceed one cylinder.

primary [ð *secondary*] – specifies the number of cylinders available for primary and secondary allocation. The amount(s) can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n'). If secondary is specified, space for a component can be expanded to include a maximum of 123 extents or 16 extents per volume for a reusable (dynamic) file. For a component with the UNIQUE attribute, 16 extents are allowed per volume.

If KEYRANGES is not specified and the data component is to be contained on more than one volume, the *primary* value given above is allocated on one volume only (when the component is defined). When the component increases in size (due to secondary allocations) so as to extend on to additional volumes, the first allocation on each overflow volume is the *primary* value. If KEYRANGES is specified and the data component is to be contained on more than one volume, the *primary* value is immediately allocated on each volume required for the key ranges.

The *secondary* value is allocated on all volumes on an as-needed basis regardless of the specification of KEYRANGES.

Abbreviation: CYL

DATA (*options*)
specifies attributes of the data component of the cluster; if these attributes are specified for the cluster as a whole, they are overridden by the options specified in the DATA parameter. If a name is not specified for the data component, a name is generated and listed for it.

DEFAULTVOLUMES (*see* VOLUMES)

ERASE | NOERASE
specifies whether the data component being defined is to be erased (overwritten with binary zeros) when its entry in the catalog is deleted.

Abbreviations: ERAS and NERAS

Default: NOERASE

EXCEPTIONEXIT (*mname*)
specifies the name of the user module (that is, the phase name in the core image library) to be given control when an exception occurs during processing of the cluster or its components. An exception is any condition which causes a SYNAD exit to be taken. The user exception exit is taken before the SYNAD exit, if both are specified.

An abnormal termination occurs if the exception exit is to be loaded but cannot be because it is not in the core image library. If it is in the core image library but the load operation fails because of insufficient virtual storage, VSAM ignores the exception exit. The SYNAD exit, if specified, is taken independently.

Abbreviation: EEXT

FILE(*dname*)

specifies the *filename* of the DLBL job control statement that, together with an EXTENT statement, identifies the volumes to be used for space allocation. You must specify the FILE parameter only if you specify UNIQUE. The pertinent volumes must be mounted during the define operation.

A FILE parameter must be specified for each unique component individually or on the cluster level. If both components of a key-sequenced file are unique and reside on separate volumes, a single FILE parameter may be specified at the cluster level. This case requires one DLBL statement with EXTENT statements for each of the two components. You must specify allocation information in the EXTENT statement(s) that corresponds to a unique component. Note that you can omit the symbolic unit from the EXTENT statement.

If both components are unique and reside on the same volume, FILE must be specified for both DATA and INDEX. In this case the components cannot share the same DLBL/EXTENT statements.

Note: This parameter explanation applies to the job control simplification introduced in Release 2. If you are using Release 1 job control, refer to the appropriate explanation in Appendix J.

FOR(*days*) | TO(*date*)

specifies the retention period for the cluster.

FOR(*days*) specifies the number of days for which the cluster is to be kept. If the number specified is 0 through 1830, the cluster is retained for the number of days specified; if the number is 1831 through 9999 (maximum), the cluster is retained through the year 1999. 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, respectively, and enclosed in single quotation marks.

TO(*date*) specifies the date until which the cluster is to be kept. The date has the form yyddd, where yy (00 through 99) is the year and ddd (001 through 366) is the day.

Default: If neither TO nor FOR is specified, the cluster can be deleted at any time.

FREESPACE(*cipercent* [*capercent*])

specifies, for a key-sequenced file, the amount of space that is to be left empty after the initial load or any extension and after any split of control intervals (cipercent) or control areas (capercent) during sequential processing. The amounts are specified as percentages. The percentages, which must be equal to or less than 100, may be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. However, even if 100% is specified, space for a least one record in a control interval and one control interval in a control area is provided.

FREESPACE cannot be specified when NUMBERED or NONINDEXED is specified.

Abbreviation: FSPC

Default: 0 percent.

IMBED | NOIMBED

specifies, for a key-sequenced file only, whether the sequence set, the lowest level of the index, is to be placed with the data component. If IMBED is coded, the sequence-set record for each control area is written as many times as it will fit on the first track (min-CA) of the control area. If the control area is less than one cylinder (max-CA), inefficient utilization of the file space can result.

Abbreviations: IMBD and NIMBD

Default: NOIMBED

Restriction: Cannot be specified when NUMBERED or NONINDEXED is specified.

INDEX (*options*)

specifies, for a key-sequenced file, attributes of the index component of the cluster; if these attributes are specified for the cluster as a whole, they are overridden by the INDEX parameter. If a name is not specified for the index component, a name is generated and listed for the index component.

Abbreviation: IX

Restriction: INDEX must not be specified when NUMBERED or NONINDEXED is specified at the cluster level.

INDEXED | NONINDEXED | NUMBERED

specifies that the cluster being defined is for a key-sequenced file (INDEXED), an entry-sequenced file (NONINDEXED), or a relative-record file (NUMBERED). If INDEXED is specified, an index component is automatically defined and cataloged even if the INDEX parameter is not specified.

If you specify NUMBERED, you cannot specify SPANNED or NONSPANNED. If you specify NUMBERED or NONINDEXED, you cannot specify IMBED, NOIMBED, INDEX, KEYRANGES, KEYS, REPLICATE, or NOREPLICATE.

Abbreviations: IXD, NIXD, and NUMD

Default: INDEXED

KEYRANGES ((*lowkey*ъ *highkey*) [ъ (*lowkey*ъ *highkey*) . . .])

specifies that portions of a key-sequenced file are to be placed on different volumes; each portion is called a *key-range.*

The maximum number of key ranges is 123. Key ranges must be in ascending order, and are not allowed to overlap. A gap can exist between two key ranges—you are not allowed to insert records within the gap. See *VSE/VSAM Programmer's Reference* for more information.

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 cannot 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.

The values *lowkey* and *highkey* can be 1 to 64 characters long or, if coded in hexadecimal, 1 to 128 hexadecimal characters long. All EBCDIC characters are allowed. Keys consisting of EBCDIC 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 coded (X'key'). See *VSE/VSAM Programmer's Reference* for more information.

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.

KEYRANGES cannot be specified when REUSE, NUMBERED, or NONINDEXED is specified.

Abbreviation: KRNG

KEYS (*length*ƀ *offset*) | 64 0
specifies information about the key field of records in a key-sequenced file.

length ƀ *offset* – specifies the length of the key-field in bytes (between 1 and 255), and its displacement (offset) from the beginning of the data record, in bytes. The sum of the length and offset cannot exceed the length of the shortest record nor the length of the first segment of a spanned record. These values can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

KEYS cannot be specified when NUMBERED or NONINDEXED is specified.

Default: 64 0

MASTERPW (*password*)
specifies a master password for the cluster or component. The master password allows all operations. If MASTERPW is not specified and if other passwords exist, the highest-level existing password automatically

becomes the master password. See CONTROLPW for the definition of password.

Abbreviation: MRPW

MODEL (*entryname* [/ *password*] [*catname* [/ *password*]])
specifies that the catalog entry of an already-defined cluster is to be used as a model for the entry being built. The model (already-defined cluster) can have space allocated to it or not. See "How to Use One Object as a Model For Another Object and Override System Defaults" in *VSE/VSAM Programmer's Reference* for more information about MODEL.

entryname – specifies the name of the entry to be used as a model. The entry to be used as a model must be of the same entry type as the entry being built. For example, if you are using this parameter at the data component level, the object used as a model must be a data component.

password – specifies a password. If the cluster or component whose entry is to be used as a model is password protected and is cataloged in a password protected catalog, either the cluster or component's password or its catalog's password is required. If both the entry password and the catalog password are specified, the catalog password will be used. If the protection attributes are to be copied, give the master password of either the model (following *entryname*) or the catalog (following *catname*). If the model's passwords are not to be copied, any password of either the model or its catalog can be used.

catname – specifies the name of the catalog that contains the entry to be used as a model. This parameter is required if (1) you are going to specify the password of the catalog that defines the cluster or component instead of specifying the password of the cluster or component itself, or (2) the catalog is not the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

Restrictions: Your job may be terminated because of allocation problems if you use the MODEL parameter and (1) specify a device type different from that specified for the model through the VOLUMES parameter, (2) change the length of keys through the KEYS parameter, (3) change the size of records, buffer space, or control intervals through the RECORDSIZE, BUFFERSPACE, or CONTROLINTERVALSIZE parameter, (4) change from UNIQUE to SUBALLOCATION, or vice versa, or (5) change the unit of allocation through the TRACKS, BLOCKS, CYLINDERS, or RECORDS parameter.

If modeling causes the cluster or component to be defined as UNIQUE, FILE must also be specified.

NAME (*entryname*)
specifies the name of the cluster or component. NAME **must be specified for the cluster**; it can optionally be specified for a data or index component. If no name is specified for a component, a name is generated and listed for you. If components are not named, you are obliged to give, in a subsequent ALTER command, the long system-generated name. (This may be enough reason for you to specify the component name). Because a cluster, data component, and (for a key-sequenced file) index component are individually named, they can be processed individually.

You can specify NAME(*entryname)* to:

a. Simply identify (name) the cluster or component (this is the most common usage).

b. Predefine the workfiles for each partition in which you anticipate to use them (called partition independence).

c. Indicate that workfile space is to be shared between central processors with the capability to run the same job in any partition *and* either processor without conflict (called partition and central processor independence).

a. *To simply name the cluster or component*, you specify a name that contain from 1 through 44 alphameric 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 through eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or a national character.

b. *To take advantage of partition independence*, you specify *entryname* as either:

name.pid

name contains a maximum of 27 characters with the same rules as above.

pid is one of the following identifiers: BG, F1, F2, F3, F4, F5, F6, F7, F8, F9, FA, FB.

or

%name

% (percent sign) indicates that you want VSAM to append a partition identifier to the name you specify. In this case the DEFINE command must execute in the partition where you want the file to be defined. For example if you specify %MY.FILE, VSAM generates the following: MY.FILE.BG (this assumes that the job was run in the BG partition).

If you want the workfile to occupy no data space while it is not open, specify the NOALLOCATION parameter in this DEFINE.

name contains a maximum of 27 characters with the same rules as above.

c. *To take advantage of partition and central processor independence,* you specify *entryname* as either:

name.pid.cpu-id.model

name (27 characters) and *pid* have the same meanings as above.

cpu-id is the six-character CPU identification number (you must precede it with a C).

model is the four-character model number (you must precede it with an M).

or

%%name

%% indicates that you want VSAM to append a unique CPU identification and partition identifier to the name you specify. In this case

the DEFINE command must execute in the partition and the system where you want the file to be defined. For example if you specify %%MY.FILE, VSAM generates the following:

MY.FILE.BG.C010658.M4341 (this assumes that the job was run in the BG partition on a processor that had an identification number of 010658 and a model number of 4341).

If you want the workfile to occupy no data space while it is not open, specify the NOALLOCATION parameter in this DEFINE.

name (27 characters) has the same meaning as above.

NOALLOCATION

indicates that no space allocation is to take place for this define. This allows you to create (a) a *dynamic file,* (b) a *model (default list of parameters*) that can be applied, in subsequent define operations, to all the files of a particular file organization, or (c) a *model* that can be applied, in subsequent define operations, to a specific file. See *VSE/VSAM Programmer's Reference* for more information on dynamic files and modeling.

- You create a *dynamic file* by specifying the NOALLOCATION parameter together with REUSE. No space is suballocated to a dynamic file at define time, rather, the required space (specified with a space allocation parameter at define time) is suballocated to the file when VSAM opens it. Dynamic files can be entry-sequenced, key-sequenced, or relative-record files.

  When a dynamic file is closed, the file is reset to empty (high-used RBA set to 0, and all space is deallocated) if (1) the ACB specifies (a) PARMS=(CLOSDSP=DELETE) or (b) PARMS=(CLOSDSP=DATE) and the expiration date has been reached, or (2) the DLBL statement specifies DISP=(,DELETE) or DISP=(,DATE) and the expiration date is reached.

- You create a *default model for the files of a particular file organization* (key-sequenced, entry-sequenced, relative record) by specifying in this DEFINE command, the NOALLOCATION parameter, a reserved entryname (see below) and the specific list of parameters that you want the model to have. The parameters in this model can be viewed as a system default because they effectively override the actual system default when the parameter is not explicitly specified.

  No data space is ever occupied by this type of model; it exists solely as an entry in the catalog. This also applies to any space allocation parameters you specify with NOALLOCATION; VSAM uses them for space allocation in subsequent defines. Each catalog can contain one model for each of the five different file organizations. The reserved entrynames that you specify for a particular file organization are:

Reserved Entryname	File Organization
DEFAULT.MODEL.KSDS	Key-sequenced file
DEFAULT.MODEL.ESDS	Entry-sequenced file
DEFAULT.MODEL.RRDS	Relative record file
DEFAULT.MODEL.AIX	Alternate index
DEFAULT.MODEL.ESDS.SAM	Managed SAM file*

* Refer to *Using the VSE/VSAM Space Management for SAM Feature.*

- You create a *model* that applies to one specific file (and occupies no data space) by specifying in the DEFINE command, the NOALLOCA-

TION parameter, an ordinary *entryname*, and the specific parameters to be included in the model.

This differs from the preceding case because in this case you can have more than one model for a particular file organization. You use this type of model by specifying its *entryname* (instead of a *reserved* entryname) in the MODEL parameter of a subsequent DEFINE command.

In the case of default models, the BUFFERSPACE, RECORDSIZE, CONTROLINTERVALSIZE, or the four password parameters (READPW, etc.) are not modeled.

If you specify NOALLOCATION for the data component of a key-sequenced file you must also specify it for the index component; you can, and normally would, specify it only at the cluster level in which case it propagates to both the data and index component levels.

Abbreviation: NAL

NOERASE (see ERASE)

NOIMBED (see IMBED)

NONINDEXED (see INDEXED)

NONSPANNED (see SPANNED)

NOREPLICATE (see REPLICATE)

NOREUSE (see REUSE)

NOWRITECHECK (see WRITECHECK)

NUMBERED (see INDEXED)

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, DEFINE processing is terminated.

Abbreviations: ORD and UNORD

Default: UNORDERED

OWNER (*ownerid*)
identifies the owner of the cluster. The *ownerid* may contain one through 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 also be coded in hexadecimal form (X'ownerid').

READPW (*password*)
specifies a read password for the cluster or component. The read password permits read operations. See CONTROLPW for the definition of password.

Abbreviation: RDPW

RECORDFORMAT (*format*)
(see *Using the VSE/VSAM Space Management for SAM Feature,* SC24-5192)

RECORDS (*primary* [*secondary*])
specifies the number of records for which space is to be allocated. If RECORDS is specified at the cluster or data component level, the space must be large enough to include the index records or the file will not hold the expected number of data records. See the CYLINDERS parameter for the information and restrictions that apply to CKD devices. See the BLOCKS parameter for information on FBA devices.

Abbreviation: REC

RECORDSIZE (*average maximum*)
specifies the average and maximum lengths, in bytes, of the data record. These values can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. The minimum record size that can be specified is one.

Control interval size will be adjusted, if necessary, to accomodate the maximum record size plus 7 (non-spanned) or average record size plus 10 (spanned). The maximum control interval size is 32,768. For spanned records, the maximum record size cannot exceed control area size minus 10 bytes of control information for each control interval contained in the control area.

When defining a *relative-record file,* the average and maximum record sizes must be equal.

Abbreviation: RECSZ

Default: The default values 4089 and 4089 are used for the average and maximum record sizes if SPANNED is not specified, 4086 and 32600 if SPANNED is specified. This default (4086 or 4089) overrides the data component CONTROLINTERVALSIZE default value of 2048.

RECOVERY | SPEED
specifies whether control areas allocated to the data component are to be preformatted before records are inserted into them. SPEED/RECOVERY applies only to initial loading.

When RECOVERY is specified or defaulted to, 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, your program may be able to 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.) The BLDINDEX, IMPORT, IMPORTRA, and REPRO commands do not provide the resume-loading capability.

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 or default to 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.

You cannot use the VERIFY command to recover from a system failure that occurred during the initial load of a file unless you specify or default to RECOVERY.

Abbreviation: RCVY

Default: RECOVERY

REPLICATE | NOREPLICATE
specifies, for a key-sequenced file only, whether each index record is to be written on a track (min-CA) as many times as it will fit to reduce rotational delay and improve performance. The sequence set is implicitly replicated if IMBED is specified, even if REPLICATE is not specified.

Abbreviations: REPL and NREPL

Default: NOREPLICATE

REUSE | NOREUSE
specifies whether the cluster can be opened repetitively as a new file, that is, with its high-used RBA set to 0. This parameter allows the use of a VSAM file as a work file. When a reusable file is opened for output, its high-used RBA is set to 0 if you open it with (1) an ACB that specifies MACRF=RST, or (2) a DLBL statement that specifies DISP=NEW. When a reusable file is closed, its high-used RBA is set to 0 if (1) the ACB specifies (a) PARMS=(CLOSDSP=DELETE) or (b) PARMS=(CLOSDSP=DATE) and the expiration date has been reached , or (2) the DLBL statement specifies DISP=DELETE or DISP=DATE and the expiration date has been reached.

You can define entry-sequenced, key-sequenced, and relative-record files as reusable. Alternate indexes may not be defined over a reusable cluster; however, it is permissible to define reusable alternate indexes that are related to a nonreusable base cluster.

Reusable files can be multi-volumed and are restricted to a maximum of 16 physical extents per volume. REUSE must not be specified together with UNIQUE or KEYRANGES.

Abbreviations: RUS and NRUS

Default: NOREUSE

SHAREOPTIONS (*value* [b *reserved*])
specifies how a file (data or index component of a cluster) can be shared within one or more VSE systems. Files that are shared across VSE systems must reside on a shared DASD device (this excludes the 23xx series of devices - they can share files within one system only). A file cannot be deleted or reset, or its share option value altered, if it is currently open for another program, regardless of the share option value specified.

During the initial load of a file (and irrespective of the share-option value specified) VSAM treats the share-option specification as if it were SHAREOPTIONS (1). After the file is loaded and successfully closed, VSAM uses the original share-option value.

value – specifies the degree of file sharing on one or more VSE systems. If a file cannot be shared for the type of sharing you specify, your request to open it is denied. The values that can be specified are:

1
specifies that the file being defined can be opened by any number of users for input processing; once the file is opened for input it cannot be opened for output processing. Conversly once the file has been opened for output processing, no other user can open it for input or output until that (the first) output processing has been completed. This option ensures full read and write integrity. The default value is 1.

2
specifies that the file being defined can be opened by any number of users for input processing even if one user is using it for output processing. No more than one user can open the file for output at one time. This option ensures write integrity only, because the file might be modified while records are being retrieved from it. Therefore, each user must ensure his own file's read integrity.

3
specifies that the file can be opened by any number of users for both input and output processing. Except for initial load processing, VSAM does nothing to ensure read or write integrity; it only ensures that the opened file is not deleted or reset, and that its share option value is not altered.

4
specifies that the file can be opened by any number of users for input processing. At the same time the file can be opened by one or more users on a *single* system for output processing. The system that gains exclusive control of the file for output processing will be the one that first issues a request for output processing. VSAM ensures write integrity each time a record is updated or inserted as in share option 2.

You can ensure read integrity by retrieving records with the update option; if you do not use the update option, some records in control intervals being updated concurrently by more than one program might be missed or skipped by VSAM because each program might retrieve a different copy of the control interval.

Sharing option 4 is not valid for an entry-sequenced file; if you specify share option 4, VSAM treats the specification as though share option 2 had been specified.

reserved – for OS/VS compatibility only. The valid OS/VS values for this subparameter are 3 and 4. This subparameter refers to cross system sharing in OS/VS; it has no effect in VSE except that VSAM stores its value in the catalog. (This subparameter takes effect in OS/VS in those cases where the file is imported into an OS/VS system). The default value is 3.

Abbreviation: SHR

SPANNED | NONSPANNED
specifies whether or not the maximum length of a data record may be greater than the control interval size, that is, whether a data record may span control intervals. A spanned record always begins on a control interval boundary. SPANNED cannot be specified when NUMBERED is specified.

If NONSPANNED is specified and the computed control interval size is less than the maximum record size, an error message is issued and the object is not defined.

Abbreviations: SPND and NSPND

Default: NONSPANNED

SPEED (see RECOVERY)

SUBALLOCATION (see UNIQUE)

TO (see FOR)

TRACKS (*primary* [*secondary*])
specifies the number of tracks to be allocated. If the space is allocated to a cluster with the UNIQUE attribute, the space is rounded up to the nearest cylinder. If you specify a number equal to or greater than the number of tracks per cylinder, then the allocation is made in terms of cylinders. See the CYLINDERS parameter for more information and restrictions.

Abbreviation: TRK

UNIQUE | SUBALLOCATION | NOALLOCATION
specifies whether the cluster's components are allocated space of their own, whether a portion of previously-defined VSAM data space is to be used for each component, or whether no space is to be allocated to the cluster (see NOALLOCATION).

UNIQUE cannot be specified together with REUSE. You cannot define a unique file in a volume which does not already contain a VSAM data space if the unique file's catalog entry is to be in a recoverable catalog.

If SUBALLOCATION is specified: a data space must exist on the volume on which the cluster or components are to reside. The name of the data space, not of the component, appears in the VTOC.

If UNIQUE is specified:

- A cluster's data and (for a key-sequenced file) index component is allocated space of its own and the name of the component appears in the VTOC of the volume(s).
- For the data component of a key-sequenced cluster and the data component resides on a 3330 volume, the index component must reside on a 3330 volume.
- For any cluster or component, each volume specified through the VOLUMES parameter must have at least one and no more than 16 EXTENT statements. The associated logical unit must not be assigned to IGNORE or UA (unassigned).
- For CKD devices, data extents must begin on cylinder boundaries.

- A FILE parameter must be specified for each unique component individually or on the cluster level.
- For a key-sequenced cluster with KEYRANGES specified, each key range must be on a different volume and there cannot be fewer volumes specified through the VOLUMES parameter than there are key-range pairs specified through KEYRANGES.

Abbreviations: UNQ and SUBAL

Default: SUBALLOCATION

UNORDERED (see ORDERED)

UPDATEPW(*password*)
specifies an update password for the cluster or component. The update password permits read and write operations. See CONTROLPW for the definition of password.

Abbreviation: UPDPW

USECLASS (*primary* [*secondary*])
specifies the class of data space to be occupied by a nonunique (suballocated) cluster or component. For the *primary allocation* of space:

0 specifies that the cluster or component is to occupy class 0 data space. Data spaces defined under OS/VS and DOS/VS Release 34 (and prior releases) are class 0 data spaces. 0 is the default space classification.

1 specifies that the cluster or component is to occupy class 1 data space. You are strongly urged to use class 1 for fixed-head areas of direct access storage.

2–7 specifies that the cluster or component is to occupy the specified class of data space (user-defined).

For the *secondary allocation* of space:

0 specifies class 0 data space is to be used by the cluster or component.

P specifies that the primary space classification is to be used. P is the default.

Whenever the cluster or component increases in size so as to extend on to additional volumes, the first suballocation on each overflow volume is directed to the class of space specified by the primary subparameter. All subsequent allocations on each overflow volume use the class of space that is specified by the secondary subparameter. If the file is extended after you IMPORT CONNECT to OS/VS or DOS/VS Release 34 (or earlier), no USECLASS assignments are made (the file is extended into any available data space because class specifications apply to VSE/VSAM only). The file is still processable under VSE/VSAM, but the LISTCAT output may be inaccurate.

If you assign a non-zero USECLASS value to a unique cluster or component, the DEFINE will terminate.

If USECLASS is specified only at the cluster level, it is applied also to the data and index component levels. If USECLASS is specified at the data

level only, it is applied also to the index level. If you want different classifications assigned to the data and index components of a key-sequenced file you must specify (or default) different values at both the data and index levels.

See "Data Space Classification" in *VSE/VSAM Programmer's Reference* for more information.

Abbreviation: USCL

VOLUMES (*volser* [ъ *volser* . . .]) | DEFAULTVOLUMES
specifies the volume(s) to contain the cluster or component.

VOLUMES
can be specified as a parameter of CLUSTER or DATA,or of both DATA and INDEX. If more than one volume is specified with a single VOLUME parameter, they must be of the same device type except if you are establishing (defining) a default model (you can specify different device types in the model).

If you want the data and index components to reside on different volumes, you must specify the VOLUMES parameter together with DATA and INDEX.

You can specify up to 123 volumes for each component. A volume serial number, *volser*, may contain one to six alphameric, national (@, #, and $), and special characters (commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs). Single quotation marks within a volume serial number must be coded as two single quotation marks. A volume serial number must be enclosed in single quotation marks if it contains a special character. For consistency with VSE job control, only alphameric characters should be used.

DEFAULTVOLUMES
is used in conjunction with the MODEL parameter to indicate that VSAM is to select the volume(s) it needs from a list of volume(s) established in a default model instead of from volumes specified in the named model or the current define (see "Default Volumes" in *VSE/VSAM Programmer's Reference* for more information).

An error occurs if you specify DEFAULTVOLUMES and VOLUMES together at the same level. You can specify DEFAULTVOLUMES at any level and in any combination of levels; if specified at the cluster level, DEFAULTVOLUMES propagates to the data component level and, if applicable, to the index component level. If DEFAULTVOLUMES propagates to the data and index component levels where a VOLUMES parameter has been specified, the VOLUMES parameter takes precedence.

If DEFAULTVOLUMES is specified (or defaulted to) at any level with the ORDERED or UNIQUE parameters, VSAM indicates an error.

Abbreviations: VOL and DFVOL

Default: DEFAULTVOLUMES

WRITECHECK | NOWRITECHECK

specifies whether to check the data transfer of records written in the cluster. If WRITECHECK is specified, a record is written and then read, without data transfer, to test for the data check condition. If NOWRITECHECK is specified, a record is written but no checking occurs.

Abbreviations: WCK and NWCK

Default: NOWRITECHECK

DEFINE MASTERCATALOG

The DEFINE command (DEF) is used to define the master catalog. When you define a master catalog, a data space to contain the catalog is automatically created. Entries for both the master catalog itself and the volume containing the data space are placed in the master catalog. A catalog must have a master password if any VSAM files cataloged in it are to be password protected. A catalog must have an update or higher-level password if any nonVSAM files cataloged in it are to be password protected. See "Defining the Master Catalog" in "Using DEFINE: Defining Objects in a Catalog" and Example 1 in "Appendix A: Sample Job Streams" for more information.

The format of the DEFINE command is shown below. Note that the organization of the command consists of three groupings. These groupings are referred to as *levels* in this book; note that some parameters appear in more than one level.

Master Catalog Level	
DEFINE	MASTERCATALOG ({DEDICATE| CYLINDERS (*primary* [*secondary*]) | BLOCKS (*primary* [*secondary*]) | RECORDS (*primary* [*secondary*]) | TRACKS (*primary* [*secondary*]) } FILE(*dname*) NAME (*entryname*) VOLUME (*volser*) [ATTEMPTS (*number*)] [AUTHORIZATION (*entrypoint* [*string*])] [BUFFERSPACE (*size*)] [CLASS (*value*)] [CODE (*code*)] [CONTROLPW (*password*)] [FOR (*days*) | TO (*date*)] [IMBED|NOIMBED] [MASTERPW (*password*)] [ORIGIN (*tracknumber*| *blocknumber*)] [OWNER (*ownerid*)] [READPW (*password*)] [RECOVERABLE|NOTRECOVERABLE] [UPDATEPW (*password*)] [WRITECHECK|NOWRITECHECK])

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

(continued on next page)

Data Component Level	
	[DATA ([BUFFERSPACE(*size*)] [CYLINDERS(*primary* [,*secondary*]) | BLOCKS(*primary* [,*secondary*]) | RECORDS(*primary* [,*secondary*]) | TRACKS(*primary* [,*secondary*])] [RECOVERABLE|NOTRECOVERABLE] [WRITECHECK|NOWRITECHECK])]

Index Component Level	
	[INDEX ([CYLINDERS(*primary*) | BLOCKS(*primary* [,*secondary*]) | RECORDS(*primary*) | TRACKS(*primary*)] [IMBED|NOIMBED] [WRITECHECK|NOWRITECHECK])]

DEFINE MASTERCATALOG Parameters: Summary

The parameters of the DEF MCAT command can be divided into the following categories:

Name, which specifies:

- The name of the catalog being defined (NAME). **This is a required parameter.**

Data organization, which specifies:

- Whether the sequence-set records (the lowest level of the index) are to be placed with the data component of the catalog (IMBED, NOIMBED).

Allocation, which specifies:

- The volume and location on the volume where the master catalog is to reside (VOLUME, ORIGIN).
- The amount of space to be allocated (BLOCKS, CYLINDERS, DEDICATE, RECORDS, or TRACKS). **One of these parameters is required.**
- The classification of the catalog's data space (CLASS). See "Data Space Classification" in *VSE/VSAM Programmer's Reference* for more information.
- The space to be provided for buffers (BUFFERSPACE).

Protection and integrity, which specifies:

- Passwords to be associated with the catalog (MASTERPW, CONTROLPW, UPDATEPW, READPW).

- A prompting code (CODE) and number of attempts allowed to provide the correct password in response to prompting at the operator's console (ATTEMPTS).
- A user-supplied authorization verification routine (AUTHORIZATION).
- The owner of the catalog (OWNER).
- A retention period (FOR, TO).
- Whether write-check operations are to be performed as records are inserted in the catalog (WRITECHECK, NOWRITECHECK).
- Whether a catalog recovery area (CRA) is to be created on each volume owned by the catalog (RECOVERABLE, NONRECOVERABLE).

DEFINE MASTERCATALOG Parameters

ATTEMPTS (*number*)
specifies the maximum number of times (0 through 7) 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 value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Abbreviation: ATT

Default: 2

AUTHORIZATION (*entrypoint* [ƀ *string*])
specifies that in addition to passwords, you are supplying a routine to check the authority of a processing program to access the master catalog. VSAM transfers control to the verification routine only after the program trying to open the master catalog gives a correct password other than the catalog's master password. (The verification routine is always bypassed whenever a correct master password is specified.) See *VSE/VSAM Programmer's Reference* for more information.

entrypoint – specifies the entry point name of the USVR. The entry point name may contain one through eight alphameric, national (@, #, and $), or special (hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or a national character. *Entrypoint* is the phase name in the core image library.

string – specifies up to 255 characters that are to be passed to the USVR when it receives control to verify authorization. The string must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a string must be coded as two single quotation marks if the string is enclosed in single quotation marks. If the string is specified in hexadecimal, it must be preceded by X and enclosed in single quotation marks (X'string').

Abbreviation: AUTH

BLOCKS (*primary* [ƀ *secondary*])
specifies, *for FBA devices only,* the number of blocks to be allocated to the master catalog. Each block is a fixed size of 512 bytes. This parameter must be specified at the catalog level. In addition, it can be specified at the data component level, or at both the data and index component levels.

primary [*secondary*] – specifies the number of blocks to be made available for primary and secondary allocation. Primary and secondary can be specified in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Secondary allocation of space to the catalog depends on the following:

- If you do not specify a *secondary* value, VSAM itself extends the catalog's data and index components an additional 15 times (one control area at a time).
- If you specify *secondary* at the data component level, VSAM uses the specified value to extend (if needed) your catalog an additional 15 times. Note that if the value you specify does not coincide with a control area boundry, VSAM rounds it down to the nearest multiple (control area) and then uses this multiple for the secondary extensions.
- If you specify a *secondary* value at the catalog level or index component level, VSE VSAM ignores it (in this case the specified value is solely an entry in the catalog for purposes of OS/VS compatibility) and instead extends the index component by one control area at a time for an additional 15 times.

If you specify a value that does not coincide with a min-CA (track) boundary, VSAM rounds it up to a value that equals a min-CA boundary. For example, if you specify BLOCKS (40) for the 3310, VSAM rounds 40 up to 64.

A further consideration applies to the BLOCKS specification at the catalog level only. VSAM acquires the data space for the catalog concurrently with the DEFINE MASTERCATALOG command. If the beginning block number (specified in the ORIGIN parameter) does not coincide with a min-CA boundary, VSAM rounds it up to the next min-CA boundary; if the ending block (ORIGIN value plus BLOCKS value) does not coincide with a min-CA boundary, VSAM rounds it down to the previous min-CA boundary. For example, ORIGIN (40) and BLOCKS (100) for the 3310 indicates that space for the catalog begins at block number 64 (40 is rounded to 64) and ends at block number 127 (39 is rounded down to 127).

Be aware that in some instances, after VSAM rounds the values you specified, zero space is the resultant allocation. For example, if you specify ORIGIN (40) and BLOCKS (50), VSAM rounds 40 up to 64 as before. However when VSAM rounds the ending block value of 89 (40 + 50 - 1 = 89) down to the closest min-CA boundary, the value (63) is less than the beginning block value. Therefore no data space is available for allocation to the catalog. (Beginning block is 64: the ending block is 63).

TRACKS or CYLINDERS cannot be specified for FBA devices; the RECORDS or DEDICATE parameters are valid for FBA devices. For further information on min-CAs and FBA devices see "Min-CA, Max-CA" in *VSE/VSAM Programmer's Reference.* For more information on space allocation, see "How Data Space is Assigned to a Catalog."

Abbreviation: BLK or BLOCK

BUFFERSPACE (*size*)

specifies the minimum space to be provided for buffers when the master catalog is being used.

size – is the number of bytes to be provided for buffers when the master catalog is being used. The amount must be 3072 (minimum), 4096, 5120,

6144, 7168, or 8192 (maximum) bytes. It can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Abbreviation: BUFSPC or BUFSP

Default: 3072 if BUFFERSPACE is not coded; 8192 if a value greater than 8192 is specified.

CLASS (*value*)
specifies the data space classification to be assigned to the total amount of data space allocated for the catalog (even that portion of space not used, if any, by the catalog). The values you can specify (only at the master catalog level) are:

0 specifies the standard space class. Data spaces defined under OS/VS and DOS/VS Release 34 (and prior releases) are treated as class 0 data spaces. (OS/VS and DOS/VS ignore space classifications). 0 is the default value.

1 indicates that class 1 is to be assigned to the data space. You are urged to assign class 1 to fixed-head areas of direct access storage.

2–7 specifies that the defined data space is classified according to your own criteria. You can use any criteria to classify the data space; for example, you can assign particular files to the middle section of a volume if you feel that by doing this the performance of your installation will be enhanced.

See "Data Space Classification" in *VSE/VSAM Programmer's Reference* for more information on space classes.

CODE (*code*)
specifies a code name for the master catalog. If an attempt is made to access a password protected catalog without a password, the code name is used in a prompting message to the operator rather than the name of the catalog. The operator can then provide the correct password. The code may contain from one through eight EBCDIC characters.

The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. If a single quotation mark appears within the code, it must be coded as two single quotation marks. Code can also be expressed in hexadecimal (X'code') form.

If CODE is not specified and an attempt is made to access a password-protected catalog without supplying a password, the operator is prompted with the name of the master catalog.

CONTROLPW (*password*)
specifies a control password for the master catalog. The control password permits read and write operations using control interval access and all operations permitted by the update and read passwords. Even though the master catalog is a form of cluster and conventional clusters may be opened with a control password, the master catalog cannot be opened as a file unless the user supplies its master password. A master password permits such operations as REPRO to be performed on the catalog. See "Passwords to Authorize Access" in *VSE/VSAM Programmer's Reference* for additional information.

password – is a one through 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. Password can also be expressed in hexadecimal (X'password') form.

Note: Authorization checking for files cataloged in the master catalog is not performed if the master catalog is not password protected.

Abbreviation: CTLPW

CYLINDERS (*primary* [*secondary*])

specifies, *for CKD devices only*, the number of cylinders to be allocated. Either this parameter or the BLOCKS, DEDICATE, RECORDS, or TRACKS parameter **must be specified** at the master catalog level. In addition, they (except DEDICATE), can be specified at the data level, or at both the data and index levels. See "How Data Space is Assigned to a Catalog" for more information. Do not specify CYLINDERS or TRACKS for FBA devices or BLOCKS for CKD devices.

primary [*secondary*] – specifies the number of cylinders available for primary and secondary allocation. These values can be expressed in decimal (n), hexadecimal (X'n') or binary (B'n') form.

The primary suballocation for the combined data and index components must be contained in a single extent (contiguous data space).

Secondary allocation of space to the catalog depends on the following:

- If you specify *secondary* at the data component level, VSAM uses the specified value to extend (if needed) your catalog an additional 15 times. Note that if the value you specify does not coincide with a control area boundary, VSAM rounds it down to the nearest multiple (control area) and then uses this multiple for the secondary extensions.
- If you specify a *secondary* value at the catalog level or index component level, VSE VSAM ignores it (in this case the specified value is solely an entry in the catalog for purposes of OS/VS compatability) and instead extends the index component by one control area at a time for an additional 15 times.
- If you do not specify *secondary* value, VSAM itself extends the catalog's data and index components an additional 15 times (one control area at a time).

Abbreviation: CYL

DATA (*options*)

specifies attributes of the data component of the catalog. The suballocation of space for the data and index components of the master catalog is discussed in "How Data Space is Assigned to a Catalog."

If DATA allocation parameters are not coded, the space specified at the master catalog level is available to contain catalog entries only.

The total amount of space specified through DATA and INDEX parameters cannot be greater than that specified at the MASTERCATALOG level.

DEDICATE

specifies that the unowned and unallocated space (maximum of 16 extents) on VOLUME(volser) is to be owned by the master catalog. (If the catalog is recoverable, VSAM suballocates one cylinder (max-CA) of this space for the CRA.) DEDICATE is mutually exclusive with the space allocation parameters (BLOCKS, RECORDS, TRACKS, CYLINDERS) and can be specified at the catalog level only. Do not specify the ORIGIN parameter with DEDICATE. There must be a contiguous area of space on the volume large enough to contain the primary allocation.

You can specify:

- DEDICATE alone — no space parameters (BLOCKS, TRACKS, RECORDS, or CYLINDERS) are specified at the data and index component levels; in this case the data space suballocated to the catalog itself is calculated by VSAM to be the minimum size catalog and the remaining space is available for later use by VSAM.
- DEDICATE at the catalog level and a space parameter value at both the data component and index component levels; in this case the data space suballocated to the catalog itself, is the sum of the values specified at the data and index component levels. Any VSAM data space that is not suballocated at this time is available for later use by VSAM.
- DEDICATE at the catalog level and a space parameter value at the data component level; in this case VSAM calculates the space required for the index component and adds this amount to the data component specification. This sum becomes the amount of data space that is suballocated to the catalog. Any VSAM space that is not suballocated at this time is available for later use by VSAM.

Abbreviation: DED

FOR(*days*) | TO(*date*)

specifies the retention period for the master catalog.

FOR(*days*)

specifies the number of days the master catalog is to be kept. If the number specified is 0 through 1830, the catalog is retained for the number of days specified; if the number is 1831 through 9999 (maximum), the catalog is retained through the year 1999. This value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

TO(*date*)

specifies the date until which the master catalog is to be kept. The date has the form yyddd, where yy (00 through 99) is the year and ddd (001 through 366) is the day.

Default: The expiration date is set to 0, which means that the master catalog can be deleted whenever it is empty.

IMBED | NOIMBED

specifies whether the sequence set (the lowest level of the index) is to be placed with the data or index component of the catalog. If you specify IMBED (or it is defaulted to) the sequence-set record for each control area is written as many times as it will fit on the first track (min-CA) of the control area. If you specify NOIMBED, the sequence-set records are

stored together with the index set (high-level index) in the index component of the catalog. IMBED is the default parameter for the catalog level.

If you specify IMBED or NOIMBED at the index component level, it overrides the specification made (or defaulted to) at the catalog level. A catalog defined with the NOIMBED parameter cannot be processed by OS/VS VSAM or earlier releases (Release 34 and previous) of DOS/VS VSAM unless you convert the non-imbedded catalog to an imbedded catalog. See "Converting an Imbedded Catalog" in *VSE/VSAM Programmer's Reference* for conversion information.

Abbreviation: IMBD or NIMBD

INDEX (*options*)
specifies attributes of the index component of the catalog. The suballocation of space for the data and index components of the master catalog is discussed in "How Data Space is Assigned to a Catalog."

An allocation parameter (BLOCKS, CYLINDERS, RECORDS, or TRACKS) must have been specified as a parameter of DATA for an allocation parameter to be specified as a parameter of INDEX. The total amount of space specified through the DATA and INDEX subparameters cannot be greater than that specified at the MASTERCATALOG level. The WRITECHECK/NOWRITECHECK parameter defaults to whatever was specified for the master catalog.

MASTERCATALOG (*options*)
specifes that a master catalog is to be defined. MASTERCATALOG is followed by the parameters specified for the catalog as a whole; they are optionally followed by the DATA and/or INDEX parameters and their subparameters.

Abbreviations: MRCAT or MCAT

MASTERPW (*password*)
specifies a master password for the master catalog. The AUTHORIZATION, CODE, and ATTEMPTS parameters have no effect unless the catalog is master password protected. If MASTERPW is not specified and if other passwords exist, the highest-level existing password automatically becomes the master password. The master password allows all operations; it is required to open the catalog as a file. See CONTROLPW for the definition of password.

Abbreviation: MRPW

NAME (*entryname*)
specifies the name of the master catalog; it is identical to the file-ID in the IJSYSCT DLBL statement. The name may contain from 1 through 44 alphameric characters, national characters (@, #, and $), and special characters (the hyphen and the 12-0 overpunch). Names containing more than eight characters must be segmented by periods; one through eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or national character. **This parameter is required.**

NOTRECOVERABLE (see RECOVERABLE)

NOWRITECHECK (see WRITECHECK)

ORIGIN(*tracknumber*| *blocknumber*)
specifies the beginning point (track number or block number) of the master catalog's data space. (If the block number does not coincide with a min-CA boundary, VSAM rounds it up to the next min-CA boundary.) VSAM determines the ending point of the data space by adding the value you specify (*at the master catalog level*) for CYLINDERS, BLOCKS, TRACKS, or RECORDS (VSAM converts records and blocks to min-CAs) to the ORIGIN specification. See "Assigning Space to a Catalog Via the ORIGIN Parameter and a Space Allocation Parameter" for more information.

Specify ORIGIN at the catalog level only. Specify a value greater than 0 for *tracknumber* and a value greater than 1 for *blocknumber*.

You can omit the ORIGIN (and DEDICATE) parameter; in this case the beginning point of the catalog's data space is the first available area on the volume that is large enough to contain your primary allocation (if you specified CYLINDERS, the beginning boundary is a cylinder boundary).

Abbreviation: ORG

OWNER (*ownerid*)
identifies the owner of the master catalog. The *ownerid* may contain one through 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 also be expressed in hexadecimal (X'ownerid') form.

READPW (*password*)
specifies a read password for the master catalog. The read password permits read operations. See CONTROLPW for the definition of password.

Abbreviation: RDPW

RECORDS (*primary* [*secondary*])
specifies the number of records for which space is to be allocated. Each record in the catalog contains 512 bytes. See the CYLINDERS parameter for the information and restrictions that apply to CKD devices. See the BLOCKS parameter for information on FBA devices.

Abbreviation: REC

RECOVERABLE | NOTRECOVERABLE
specifies that a CRA (catalog recovery area) is to be created on the catalog's volume and on each volume owned by the catalog. A CRA occupies one cylinder (max-CA); it is allocated from space defined for the catalog itself, and from the first data space defined on each new volume.

When the CRA space becomes filled, the CRA can expand (one max-CA at a time) to include a maximum of 15 additional extents. The number of records a CRA can contain varies with the device type (for example, for a 3330 with 20 control intervals per track, a 16-extent CRA will accommodate approximately 6000 records).

RECOVERABLE cannot be specified if the catalog will contain nonVSAM files.

Abbreviations: RVBL or NRVBL

Default: NOTRECOVERABLE

TO (see FOR)

TRACKS (*primary* [*secondary*])

specifies the number of tracks to be allocated. Do not specify TRACKS for an FBA device. See the CYLINDERS parameter for more information and restrictions.

Abbreviation: TRK

UPDATEPW (*password*)

specifies an update password for the master catalog. The update password permits read and write operations. See CONTROLPW for the definition of password.

Abbreviation: UPDPW

VOLUME (*volser*)

specifies the volume that is to contain the master catalog. The volume cannot be currently owned by any other VSAM catalog. The volume serial number, *volser*, may contain one to six alphameric, national (@, #, and $), and special characters (commas, blanks, semicolons, parentheses, slashes, asterisk, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs). Single quotation marks within the volume serial number must be coded as two single quotation marks. The volume serial number must be enclosed in single quotation marks if it contains a special character. **This parameter is required.**

For consistency with VSE job control, only alphameric characters should be used.

Abbreviation: VOL

WRITECHECK | NOWRITECHECK

specifies whether to check the data transfer of records written in the master catalog. If WRITECHECK is specified, a record is written and then read, without data transfer, to test for the data check condition. If NOWRITECHECK is specified, a record is written but no checking occurs.

Abbreviations: WCK and NWCK

Default: NOWRITECHECK

DEFINE NONVSAM

The DEFINE NONVSAM command (DEF NVSAM) is used to catalog a nonVSAM file in a VSAM catalog provided the catalog is not defined as recoverable. Any already existing file can be introduced into a master or user catalog through the DEFINE command. The only result of a DEFINE NONVSAM command is that an entry is created in a master or user catalog; no space is allocated or reserved. See "Defining a NonVSAM File" and Example 4 in "Appendix A: Sample Job Stream" for examples and more information.

The format of the command is:

```
DEFINE   NONVSAM
         (
            DEVICETYPES ( devtype [ƀ devtype . . . ] )
            NAME ( entryname )
            VOLUMES ( volser [ƀ volser . . . ] )
            [FILESEQUENCENUMBERS
              ( number [ƀ number . . . ] ) ]
         )
            [CATALOG ( catname [ / password ] [ƀ dname ] ) ]
```

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

DEFINE NONVSAM Parameters

CATALOG (*catname* [/ *password*])
identifies the catalog in which the nonVSAM file is to be defined. The catalog must not be recoverable. You do not have to specify this parameter (unless it is needed for password specification) when the nonVSAM file is to be defined in the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

catname – is the name of the catalog.

password – specifies the update or higher-level password of the catalog if it is password protected.

Abbreviation: CAT

DEVICETYPES (*devtype* [ƀ *devtype* . . .])
specifies the device type(s) of the volumes containing the nonVSAM file. NonVSAM files on an FBA device cannot be defined in a VSAM catalog. If the nonVSAM file 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.

Abbreviation: DEVT

Restriction: For *devtype* substitute only supported device types: 2400T9, 2400T7, 2314, 2319, 3330, 3330-11, 3340, or 3350.

FILESEQUENCENUMBERS (*number* [ƀ *number* . . .])
specifies the file sequence number of the nonVSAM file being defined if it resides on magnetic tape. This number indicates the position of the file

with respect to other files of the tape. If the file spans volumes, the file sequence number on each volume must be specified in the same order as the volumes in the VOLUMES parameter. This value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Abbreviation: FSEQN

NAME (*entryname*)

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

NONVSAM (*options*)

specifies that a nonVSAM file is to be defined. NONVSAM is followed by the parameters that describe the nonVSAM file; they are enclosed in parentheses.

Abbreviation: NVSAM

VOLUMES (*volser* [*volser* . . .])

specifies the volume(s) that are to contain the nonVSAM file. If the file resides on magnetic tape and there is more than one tape file belonging to the file 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 in the FILESEQUENCENUMBERS parameter.

volser – may contain one through six alphameric, national (@, #, and $), and special characters (commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plussigns, hyphens, and equal signs). Single quotation marks within a volume serial number must be coded as two single quotation marks. For consistency with VSE job control, only alphameric characters should be used. A volume serial number must be enclosed in single quotation marks if it contains a special character.

Abbreviation: VOL

DEFINE PATH

The DEFINE PATH command (DEF PATH) is used to define a path. No data space is occupied by a path. A path relates an alternate index and its base cluster or, alternatively, a path may be defined directly over a base cluster. A path and its related alternate index and base cluster must all be defined in the same catalog. If the alternate index or base cluster specified in the PATHENTRY parameter is password protected, the path being defined should also be password protected to ensure the desired level of data security. Omitting password protection for the path allows access to the password protected data without the need of specifying a password. See "Defining a Path" and Example 8 in "Appendix A: Sample Job Streams" for examples and more information.

The format of the DEFINE PATH command is:

```
DEFINE   PATH
         (
            NAME ( entryname )
            PATHENTRY ( entryname [ / password ] )
            [ATTEMPTS ( number ) ]
            [AUTHORIZATION ( entrypoint [ b string ] ) ]
            [CODE ( code ) ]
            [CONTROLPW ( password ) ]
            [FILE ( dname ) ]
            [FOR ( days ) | TO ( date ) ]
            [MASTERPW ( password ) ]
            [MODEL ( entryname [ / password ]
               [ catname [ / password ] [ b dname ] ] ) ]
            [OWNER ( ownerid ) ]
            [READPW ( password ) ]
            [UPDATE | NOUPDATE]
            [UPDATEPW ( password ) ]
         )
            [CATALOG ( catname [ / password ] [ b dname ] ) ]
```

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

DEFINE PATH Parameters: Summary

The parameters of the DEF PATH command can be divided into the following categories:

Name, which specifies:

- The name of the path (NAME). **This is a required parameter.**
- The alternate index or base cluster considered as the entry to the path (PATHENTRY). **This is a required parameter.**
- The catalog that contains the alternate index or base cluster named in PATHENTRY (CATALOG).
- The path to be used as a model (MODEL). See "How to Use One Object as a Model for Another Object and Override System Defaults" in *VSE/VSAM Programmer's Reference* for more information.

Allocation, which specifies:

- Whether the cluster's upgrade set is to be opened when the path (and its cluster) is opened (UPDATE, NOUPDATE).

Protection and integrity, which specifies:

- The passwords to be associated with the path (MASTERPW, CONTROLPW, UPDATEPW, READPW).
- A prompting code (CODE) and the number of attempts allowed to provide the correct password in response to prompting at the operator's console (ATTEMPTS).
- A user-supplied authorization verification routine (AUTHORIZATION).
- The owner of the path (OWNER).
- A retention period (FOR, TO).

DEFINE PATH Parameters

ATTEMPTS (*number*)
specifies the maximum number of times (0 through 7) the operator may 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 value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Abbreviation: ATT

Default: 2

AUTHORIZATION (*entrypoint* [*string*])
specifies that, in addition to passwords, you are supplying a USVR (user security verification routine) to check the authority of a processing program to access the path. VSAM transfers control to the USVR only after the program trying to open the path gives a correct password other than the master password. (The USVR is always bypassed whenever a correct master password is specified.) See *VSE/VSAM Programmer's Reference* for more information.

entrypoint – specifies the entry point of the USVR. The entrypoint name may contain one through eight alphameric, national (@, #, and $), or special (the hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or a national character. *Entrypoint* is the phase name in the core image library.

string – specifies up to 255 characters that are to be passed to the USVR when it receives control to verify authorization. The string must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a string must be coded as two single quotation marks if the string is enclosed in single quotation marks. If the string is specified in hexadecimal, it must be coded (X'string').

Abbreviation: AUTH

CATALOG (*catname* [/ *password*])
identifies the catalog that defines the alternate index or the base cluster named in the PATHENTRY parameter.

catname – specifies the name of the catalog. You do not have to specify this parameter (unless it is needed for password specification) when the entry is to be defined in the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

password – specifies the update or higher-level password if the catalog is password protected. If no password is specified and the catalog is password protected, VSAM asks the operator for the correct password.

Abbreviation: CAT

CODE (*code*)
specifies a code name for the path, if it is password protected. If an attempt is made to access the path without a password, the code name is used in a prompting message to the operator rather than the name of the path. The operator can then provide the correct password. The code may contain one through eight EBCDIC characters.

The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. If a single quotation mark appears within the code, it must be coded as two single quotation marks. Code may also be coded in hexadecimal (X'n') form.

If CODE is not specified and an attempt is made to access a password protected path without supplying a password, the operator is prompted with the name of the path. The operator can then provide the correct password.

CONTROLPW (*password*)
specifies a control password for the path. The control password permits read and write operations using control interval access and all operations permitted by the update and read passwords. See "Passwords to Authorize Access" in *VSE/VSAM Programmer's Reference* for more information.

password – is a one through 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 may also be coded in hexadecimal (X'password') form.

Abbreviation: CTLPW

FOR (*days*) | TO (*date*)
specifies the retention period for the path being defined. If neither TO nor FOR is specified, the path can be deleted at any time. Deleting it does not affect the existence of the alternate index and the base cluster.

FOR (*days*)
specifies the number of days for which the path is to be kept. If the number specified is 0 through 1830, the path is retained for the number of days specified; if the number is 1831 through 9999 (maximum), the path is retained through the year 1999. The number of days can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

TO (*date*)
specifies the date until which the path is to be kept. The date has the form yyddd, where yy (00 through 99) is the year and ddd (001 through 366) is the day.

MASTERPW (*password*)
specifies a master password for the path. If MASTERPW is not specified and if other passwords exist, the highest-level existing password automatically becomes the master password. The master password allows all operations. See CONTROLPW for the definition of password.

Abbreviation: MRPW

MODEL (*entryname* [/ *password*]
[b *catname* [/ *password*]])
specifies that the entry of an already-defined path is to be used as a model for the entry being built. See "How to Use One Object as a Model for Another Object and Override System Defaults" in *VSE/VSAM Programmer's Reference* for more information about MODEL.

entryname – specifies the name of the path entry to be used as a model.

password – specifies a password. If the path whose entry is to be used as a model is password protected and is cataloged in a password protected catalog, either the path's password or its catalog's password is required. If both the entry password and the catalog password are specified, the catalog password is used. If the protection attributes are to be copied, substitute the master password of either the path (following *entryname*) or its catalog (following *catname*). If the model's passwords are not to be copied, any password of either the path or its catalog can be used.

catname – specifies the name of the catalog in which the path whose entry is to be used as a model is defined. This parameter is required if (1) you are going to specify the password of the catalog that defines the path instead of specifying the password of the path itself, or (2) the catalog is not the default catalog. The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.

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

NOUPDATE (see UPDATE)

OWNER (*ownerid*)
identifies the owner of the path. The *ownerid* may contain one through 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 also be coded in hexadecimal (X'ownerid') form.

PATH (*options*)
specifies that a path is to be defined. PATH is followed by other parameters.

PATHENTRY (*entryname* [/ *password*])
specifies the name *(file-ID)* of either (a) the alternate index or (b) the base cluster that is to be considered as the entry to the path.

a. If the entry to the path is an alternate index, the termination of the path is the base cluster to which the alternate index is related (via the RELATE parameter of the DEFINE ALTERNATEINDEX command).

b. If the entry to the path is a base cluster, the *name* of the path is just another name for the base cluster, used to avoid unnecessary allocation of the upgrade set (see the UPDATE | NOUPDATE parameter).

If the alternate index or the base cluster specified as *entryname* is protected, its master password is required. Alternatively, you can specify, in the CATALOG parameter, the master password of the catalog in which the alternate index or the base cluster is defined.

Abbreviation: PENT

READPW (*password*)
specifies a read password for the path. The read password permits read operations against the data records in the base cluster. See CONTROLPW for the definition of password.

Abbreviation: RDPW

TO (see FOR)

UPDATE | NOUPDATE
specifies whether the upgrade set belonging to the base cluster of the path is to be updated. If you specify NOUPDATE and the base cluster has one or more alternate indexes in its upgrade set, these alternate indexes will not be updated.

When a NOUPDATE path is opened for processing and the path entry is a base cluster, no alternate index needs to be available. No devices need to be allocated for any alternate index and no control blocks are built for them at OPEN time. When a NOUPDATE path is opened for processing and the path entry is an alternate index, only that alternate index needs to be available. No devices need to be allocated for other alternate indexes and no control blocks are built for them at OPEN time. Thus, NOUPDATE prevents unwanted allocation of the upgrade set even though the UPGRADE attribute is set for one or more of the base cluster's alternate indexes.

If you specify UPDATE, the upgrade set will be opened with the base cluster and be updated whenever the base cluster is modified, and devices have to be allocated for it.

Abbreviations: UPD and NUPD

Default: UPDATE

UPDATEPW (*password*)

specifies an update password for the path. The update password permits read and write operations against the data records in the base cluster. See CONTROLPW for the definition of password.

Abbreviation: UPDPW

DEFINE SPACE

The DEFINE SPACE command (DEF SPC) is used to define VSAM data spaces or to reserve volumes for future use by VSAM. A VSAM data space is space on a direct access volume that is owned and managed by VSAM. See "Defining a VSAM Data Space" and Example 2 in "Appendix A: Sample Job Streams" for examples and more information.

The format of the DEFINE SPACE command is:

```
DEFINE   SPACE
         (
            {CANDIDATE|
              DEDICATE|
              CYLINDERS ( primary [ secondary ] ) |
              BLOCKS ( primary [ secondary ] ) |
              RECORDS ( primary [ secondary ] ) |
              TRACKS ( primary [ secondary ] ) }
            FILE ( dname )
            VOLUMES ( volser [ volser . . . ] )
            [ORIGIN ( tracknumber | blocknumber ) ]
            [RECORDSIZE ( average maximum ) ]
            [CLASS ( value ) ]
         )
         [CATALOG ( catname [ / password ] [ dname ] ) ]
```

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

DEFINE SPACE Parameters

BLOCKS (*primary* [*secondary*])
specifies, *for FBA devices only,* the number of blocks to be allocated. Each block is a fixed size of 512 bytes.

primary [*secondary*] - specifies the number of blocks to be made available for primary and secondary allocation. Primary and secondary can be specified in decimal (n), hexadecimal (X'n'), or binary (B'n') form. VSE ignores the secondary value; however, it can be specified for OS/VS compatibility.

If you specify a value that does not coincide with a min-CA (track) boundary, VSAM rounds it up to a value that equals a min-CA boundary. For example, if you specify BLOCKS (40) for the 3310, VSAM rounds 40 up to 64.

A further consideration applies; data space is acquired concurrently with the DEFINE SPACE command via the ORIGIN and BLOCKS parameters. If the beginning block number does not coincide with a min-CA boundary, VSAM rounds it up to the next min-CA boundary; if the ending block (ORIGIN value plus BLOCKS value) does not coincide with a min-CA boundary, VSAM rounds it down to the previous min-CA boundary.

For example, ORIGIN (40) and BLOCKS (100) for the 3310, indicates that the data space begins at block number 64 (40 is rounded up to 64) and ends at block number 127 (139 is rounded down to 127.) Be aware that in some instances, after VSAM rounds the values you specified, zero space is the resultant allocation. For example, if you specify ORIGIN (40) and BLOCKS (50), VSAM rounds 40 up to 64 as before. However when VSAM rounds the ending block value of 89 (40 + 50 -1 = 89) down to the closest min-CA boundary, the value (63) is less than the beginning block

value. Therefore no data space is available for allocation. (Beginnng block is 64: the ending block is 63.)

TRACKS or CYLINDERS cannot be specified *for FBA devices*; the RECORDS or DEDICATE parameters are valid *for FBA devices*. For further information on min-CAs and FBA devices see "Min-CA, Max-CA" in *VSE/VSAM Programmer's Reference.*

The minimum primary space allocation for the first data space on a volume belonging to a recoverable catalog is one max-CA (cylinder) (one max-CA is required by the CRA).

Abbreviations: BLK or BLOCK

CANDIDATE

specifies that the volume(s) listed in the VOLUMES parameter are reserved (candidates) for future use by VSAM. No space is allocated on the volumes(s). The volumes will be owned either by the catalog specified in the CATALOG parameter or, alternatively by the job or master catalog. Either the CANDIDATE parameter or the BLOCKS, TRACKS, CYLINDERS, DEDICATE, or RECORDS parameter must be specified.

CANDIDATE cannot be specified if (1) a volume is to be owned by a recoverable catalog (space for a catalog recovery area must be allocated at the time of defining a volume owned by a recoverable catalog), (2) a VSAM data space already exists on the volume(s) named in the VOLUMES parameter, (3) you specified the ORIGIN parameter.

Abbreviation: CAN

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

identifies the catalog in which the data space(s) are to be defined or in which candidate volume(s) are to be indicated. You do not have to specify this parameter (unless it is needed for password specification) when the space is to be owned by the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

catname – specifies the name of the catalog.

password – specifies the update or higher-level password if the catalog is password protected.

Abbreviation: CAT

CLASS (*value*)

specifies the data space classification to be assigned to the total data space being defined. A data space can be assigned to one space classification only. You cannot specify CLASS if CANDIDATE is specified. The values you can specify are:

- 0 Specifies the standard space class. Data spaces defined under OS/VS and DOS/VS Release 34 (and prior releases) are treated as class 0 data spaces (OS/VS and DOS/VS ignore space classifications). 0 is the default value.
- 1 Indicates that class 1 is to be assigned to the data space. You are urged, for consistency, to use class 1 for fixed-head areas of direct access storage.
- 2–7 Specifies that the defined data space is classified according to your own criteria. You can use any criteria to classify the data space; for example, you can assign particular files to the middle section of a volume if you feel that performance will be enhanced.

See "Data Space Classification" in *VSE/VSAM Programmer's Reference* for more information on space classes.

CYLINDERS (*primary* [*secondary*])
specifies the number of cylinders to be allocated. The value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form. Either this parameter or the CANDIDATE, BLOCKS, RECORDS, DEDICATE, or TRACKS parameter must be specified. Do not specify CYLINDERS or TRACKS *for FBA devices* or BLOCKS *for CKD devices.*

primary – specifies the number of cylinders available for primary allocation. You can specify the ORIGIN parameter to indicate the beginning point of your data space.

The minimum primary space allocation for the first data space on a volume that belongs to a recoverable catalog is one cylinder, all of which will be given to the CRA.

secondary – the secondary value is not used in VSE, but it may be specified for the compatibility of the functional command with OS/VS.

Abbreviation: CYL

DEDICATE
specifies that the unowned and unallocated data space (maximum of 16 extents) on VOLUMES (volser) is to be allocated to VSAM. DEDICATE is mutually exclusive with the space allocation parameters (BLOCKS, CANDIDATE, RECORDS, TRACKS, CYLINDERS). Do not specify the ORIGIN parameter with DEDICATE.

A maximum of 16 extents can be acquired by VSAM for each individual volume; a maximum of 255 extents can be acquired by VSAM for a given DEFINE SPACE command.

Abbreviation: DED

ORIGIN(*tracknumber* | *blocknumber*)
specifies the beginning point (track number or block number) of the data space. (If the block number does not coincide with a min-CA boundary, VSAM rounds it up to the next min-CA boundary.) VSAM determines the ending point of the data space by adding the value you specify for CYLINDERS, BLOCKS, TRACKS, or RECORDS (VSAM converts records and blocks to min-CAs) to the ORIGIN specification.

Specify a value greater than 0 for *tracknumber* and a value greater than 1 for *blocknumber*.

If you specify ORIGIN, do not specify the CANDIDATE parameter. You can omit both the DEDICATE and ORIGIN parameters; in this case the beginning point of the data space is the first available area on the volume that is large enough to contain your primary allocation (if you specified CYLINDERS, the beginning boundary is a cylinder boundary).

Abbreviation: ORG

RECORDS (*primary* [*secondary*])
specifies the number of records for which space is to be allocated. The RECORDSIZE parameter gives the size used (average record size) for calculating the amount of space. See the CYLINDERS parameter for the information and restrictions that apply to CKD devices. See the BLOCKS parameter for information on FBA devices.

RECORDSIZE must be specified if this parameter is specified.

Abbreviation: REC

RECORDSIZE (*average* ƀ *maximum*)
specifies the average and maximum record size, in bytes, to be used to calculate the amount of space to be allocated in terms of number of records. These values can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

RECORDSIZE must be specified if RECORDS is specified. The minimum size that can be specified is 1; the maximum size is 32,761 bytes.

Abbreviation RECSZ

SPACE (*options*)
specifies that a data space is to be defined.

Abbreviation: SPC

TRACKS (*primary* [ƀ *secondary*])
specifies the number of tracks to be allocated. See the CYLINDERS parameter for more information and restrictions.

Abbreviation: TRK

VOLUMES (*volser* [ƀ *volser* . . .])
specifies the volume(s) on which data space(s) are to be defined or which are to be candidates owned by the catalog indicated in the CATALOG parameter. You can specify up to 123 volumes; they must all be of the same device type. Volumes with duplicate volume serial numbers cannot be owned by the same catalog.

Either (a) the amount specified in the space allocation parameter (BLOCKS, etc.) is allocated on each respective volume, or (b) if DEDICATE is specified, the unowned and unallocated data space (maximum of 16 extents) on each respective volume is allocated to VSAM, or (c) if CANDIDATE is specified, the volume is merely marked as owned by the catalog.

A volume serial number, *volser,* may contain one through six alphameric, national (@, #, and $), and special characters (commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs). Single quotation marks within a volume serial number must be coded as two single quotation marks. A volume serial number must be enclosed in single quotation marks if it contains a special character. For consistency with VSE job control, only alphameric characters should be used.

Abbreviation: VOL

DEFINE USERCATALOG

The DEFINE USERCATALOG command (DEF UCAT) is used to define a user catalog.When you define a user catalog, a data space to contain the catalog is automatically created. An entry for the volume containing the data space is placed in the user catalog. A pointer to the user catalog is placed in the master catalog and entries describing the user catalog and its components (as a VSAM file) are placed in the user catalog itself. A catalog must have a master password if any VSAM files cataloged in it are to be password protected. A catalog must have an update or higher-level password if any nonVSAM files cataloged in it are to be password protected. See "Defining a User Catalog" in "Using DEFINE: Defining Objects in a Catalog" and Examples 1 and 2 in "Appendix A: Sample Job Streams" for examples and more information.

The format of the command is shown below. Note that the organization of the command consists of three groupings. These groupings are referred to as *levels* in this book—note that some parameters appear in more than one level. (The optional CATALOG parameter stands alone; it does not belong to any level.)

User Catalog Level	
DEFINE	USERCATALOG ({DEDICATE| CYLINDERS (*primary* [*secondary*]) | BLOCKS (*primary* [*secondary*]) | RECORDS (*primary* [*secondary*]) | TRACKS (*primary* [*secondary*]) } FILE (*dname*) NAME (*entryname*) VOLUME (*volser*) [ATTEMPTS (*number*)] [AUTHORIZATION (*entrypoint* [*string*])] [BUFFERSPACE (*size*)] [CLASS (*value*)] [CODE (*code*)] [CONTROLPW (*password*)] [FOR (*days*) | TO (*date*)] [IMBED|NOIMBED] [MASTERPW (*password*)] [MODEL (*entryname* [/ *password*] [*catname* [/ *password*] [*dname*]])] [ORIGIN (*tracknumber*| *blocknumber*)] [OWNER (*ownerid*)] [READPW (*password*)] [RECOVERABLE|NOTRECOVERABLE] [UPDATEPW (*password*)] [WRITECHECK|NOWRITECHECK])

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

Data Component Level	
	[DATA ([BUFFERSPACE(*size*)] [CYLINDERS(*primary* [,*secondary*]) | BLOCKS(*primary* [,*secondary*]) | RECORDS(*primary* [,*secondary*]) | TRACKS(*primary* [,*secondary*])] [RECOVERABLE | NOTRECOVERABLE] [WRITECHECK | NOWRITECHECK])]

Index Component Level	
	[INDEX ([CYLINDERS(*primary*) | BLOCKS(*primary*) | RECORDS(*primary*) | TRACKS(*primary*)] [IMBED | NOIMBED] [WRITECHECK | NOWRITECHECK])]

	[CATALOG(*mastercatname* [/ *password*])]

DEFINE USERCATALOG Parameters: Summary

The parameters of the DEF UCAT command can be divided into the following categories:

Name, which specifies:

- The name of the catalog being defined (NAME). **This is a required parameter.**
- The name of the master catalog that points to the user catalog (CATALOG).
- The name of the master or user catalog that is to be used as a model for this user catalog (MODEL). See "How to Use One Object as a Model for Another Object and Override System Defaults"in *VSE/VSAM Programmer's Reference* for more information.

Data organization, which specifies:

- Whether the sequence-set records are to be placed with the data component of the catalog (IMBED, NOIMBED).

Allocation, which specifies:

- The volume and location on the volume where the catalog is to reside (ORIGIN, VOLUME).
- The amount of space to be allocated (BLOCKS, CYLINDERS, DEDICATE, RECORDS, TRACKS). **One of these parameters is required.**
- The classification of the catalog's data space (CLASS). See "Data Space Classification" in *VSE/VSAM Programmer's Reference* for more information.

- The space to be provided for buffers (BUFFERSPACE).

Protection and integrity, which specifies:

- Passwords to be associated with the catalog or its components (MASTERPW, CONTROLPW, UPDATEPW, READPW).
- A prompting code (CODE) and number of attempts allowed to provide the correct password in response to prompting at the operator's console (ATTEMPTS).
- A user-supplied authorization security verification routine (AUTHORIZATION).
- The owner of the catalog (OWNER).
- A retention period (FOR, TO).
- Whether write-check operations are to be performed as records are inserted in the catalog (WRITECHECK, NOWRITECHECK).
- Whether a catalog recovery area (CRA) is to be created on each volume owned by the catalog (RECOVERABLE, NOTRECOVERABLE).

DEFINE USERCATALOG Parameters

ATTEMPTS (*number*)
specifies the maximum number of times (0 through 7) 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 value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Abbreviation: ATT

Default: 2

AUTHORIZATION (*entrypoint* [*string*])
specifies that, in addition to passwords, you are supplying a routine to check the authority of a processing program to access the user catalog. VSAM transfers control to the verification routine only after the program trying to open the catalog gives a correct password other than the catalog's master password. (The verification routine is always bypassed whenever a correct master password is specified,) See *VSE/VSAM Programmer's Reference* for more information.

entrypoint – specifies the entry point name of the user's security verification routine. The entry point name may contain one through eight alphameric, national (@, #, and $), or special (the hyphen and 12-0 overpunch) characters. The first character must be an alphabetic or a national character.

string – specifies up to 255 characters that are to be passed to the user's security verification routine when it receives controlto verify authorization. The string must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. A single quotation mark within a string must be coded as two single quotation marks if the string is enclosed in single quotation marks. If string is specified in hexadecimal, it must be preceded by X and enclosed in single quotation marks (X'string').

Abbreviation: AUTH

BLOCKS (*primary* [*secondary*])

specifies, ***for FBA devices only,*** the number of blocks to be suballocated to the user catalog. Each block is a fixed size of 512 bytes. This parameter must be specified at the catalog level. In addition, it can be specified at the data component level, or at both the data and index component levels.

primary [*secondary*] - specifies the number of blocks to be made available for primary and secondary suballocation. Primary and secondary can be specified in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

Secondary suballocation of space to the catalog depends on the following:

- If you do not specify a *secondary* value, VSAM itself extends the catalog's data and index components an additional 15 times (one control area at a time).
- If you specify *secondary* at the data component level, VSAM uses the specified value to extend (if needed) your catalog an additional 15 times. Note that if the value you specify does not coincide with a control area boundary, VSAM rounds it down to the nearest multiple (control area) and then uses this multiple for the secondary extensions.
- If you specify a *secondary* value at the catalog level or index component level, VSE/VSAM ignores it (in this case the specified value is solely an entry in the catalog for purposes of OS/VS compatibility) and instead extends the index component by one control area at a time for an additional 15 times.

If you specify a value that does not coincide with a min-CA (track) boundary, VSAM rounds it up to a value that equals a min-CA boundary. For example, if you specify BLOCKS (40) for the 3310, VSAM rounds 40 up to 64.

A further consideration applies to the BLOCKS specification at the catalog level only. VSAM acquires the data space for the catalog concurrently with the DEFINE USERCATALOG command. If the beginning block number (specified in the ORIGIN parameter) does not coincide with a min-CA boundary, VSAM rounds it up to the next min-CA boundary; if the ending block (ORIGIN value plus BLOCKS value) does not coincide with a min-CA boundary, VSAM rounds it down to the previous min-CA boundary. For example, ORIGIN (40) and BLOCKS (100) for the 3310 indicates that space for the catalog begins at block number 64 (40 is rounded to 64) and ends at block number 127 (139 is rounded down to 127).

Be aware that in some instances, after VSAM rounds the values you specified, zero space is the resultant allocation. For example, if you specify ORIGIN (40) and BLOCKS (50), VSAM rounds 40 up to 64 as before. However when VSAM rounds the ending block value of 89 (40 + 50 -1 = 89) down to the closest min-CA boundary, the value (63) is less than the beginning block value. Therefore no data space is available for allocation to the catalog. (Beginning block is 64: the ending block is 63.)

TRACKS or CYLINDERS cannot be specified ***for FBA devices***; the RECORDS or DEDICATE parameters are valid ***for FBA devices.*** For further information on min-CAs and FBA devices see "Min-CA, Max-CA" in ***VSE/VSAM Programmer's Reference.*** For more information on space allocation, see "How Data Space is Assigned to a Catalog."

Abbreviations: BLK or BLOCK

BUFFERSPACE (*size*)
specifies the minimum space to be provided for buffers when the user catalog is being used.

size – is the number of bytes to be provided for buffers when the user catalog is being used. The amount must be 3072 (minimum), 4096, 5120, 6144, 7168, or 8192 (maximum) bytes. This value can be expressed in decimal (n), hexadecimal (X'n'),or binary(B'n') form.

Abbreviations: BUFSPC or BUFSP

Default: 3072 if BUFFERSPACE is not coded; 8192 if a value greater than 8192 is specified.

CATALOG (*mastercatname* [/ *password*])
specifies the name and update or higher-level password of the master catalog, if the master catalog is password protected. The password must be provided in this parameter or in response to prompting.

Abbreviation: CAT

CLASS (*value*)
specifies the data space classification to be assigned to the total amount of data space allocated for the catalog (even that portion of space not used, if any, by the catalog). The values you can specify (only at the catalog level) are:

- 0 Specifies the standard space class. Data spaces defined under OS/VS and DOS/VS Release 34 (and prior releases) are treated as class 0 data spaces (OS/VS and DOS/VS ignore space classifications). 0 is the default value.
- 1 Indicates that class 1 is to be assigned to the data space. You are urged, for consistency, to use class 1 for fixed-head areas of direct access storage.
- 2–7 Specifies that the defined data space is classified according to your own criteria. You can use any criteria to classify the data space; for example, you can assign particular files to the middle section of a volume if you feel that performance will be enhanced.

See "Data Space Classification" in *VSE/VSAM Programmer's Reference* for more information on space classes.

CODE (*code*)
specifies a code name for the user catalog. If an attempt is made to access a password protected catalog without a password, the code name is used in a prompting message to the operator rather than the name of the catalog. The operator can then provide the correct password.

The code may contain from one through eight EBCDIC characters. The code must be enclosed in single quotation marks if it contains commas, semicolons, blanks, parentheses, or slashes. If a single quotation mark appears within the code, it must be coded as two single quotation marks when the code is enclosed in single quotation marks. Code can also be expressed in hexadecimal (X'code') form.

If CODE is not specified and an attempt is made to access a password-protected catalog without supplying a password, the operator is prompted with the name of the user catalog.

CONTROLPW (*password*)

specifies a control password for the user catalog. The control password permits read and write operations using control interval access and all operations permitted by the update and read passwords. Because the master password is required to open the catalog as a file, the CONTROLPW has little meaning for a catalog. See "Passwords to Authorize Access" in *VSE/VSAM Programmer's Reference* for more information.

password – is a one through 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 also be expressed in hexadecimal (X'password') form.

Note: Authorization checking for files cataloged in the user catalog is not performed if the user catalog is not password protected.

Abbreviation: CTLPW

CYLINDERS (*primary* [ƀ *secondary*])

specifies, *for CKD devices only*, the number of cylinders to be allocated. Either this parameter or the BLOCKS, DEDICATE, RECORDS, or TRACKS parameter must be specified at the user catalog level. In addition, they (except DEDICATE) can be specified at the data level, or at both the data and index levels. Do not specify CYLINDERS or TRACKS *for FBA devices*, or BLOCKS *for CKD devices*. See "How Data Space is Assigned to a Catalog" under "Using Access Method Services" for more information.

primary [ƀ *secondary*] – specifies the number of cylinders available for primary and secondary allocation. These values can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

The primary allocation for the combined data and index components must be contained in a single extent (contiguous data space).

Secondary suballocation of space to the catalog depends on the following:

- If you specify *secondary* at the data component level, VSAM uses the specified value to extend (if needed) your catalog an additional 15 times. Note that if the value you specify does not coincide with a control area boundary, VSAM rounds it down to the nearest multiple (control area) and then uses this multiple for the secondary extensions.
- If you specify a *secondary* value at the catalog level or index component level, VSE VSAM ignores it (in this case the specified value is solely an entry in the catalog for purposes of OS/VS compatibility) and instead extends the index component by one control area at a time for an additional 15 times.
- If you do not specify a *secondary* value, VSAM itself extends the catalog's data and index components an additional 15 times (one control area at a time).

Abbreviation: CYL

DATA (*options*)

specifies attributes of the data component of the catalog. The allocation of space for the data and index components of the user catalog is discussed in

"How Data Space is Assigned to a Catalog." Buffer space, write-checking, and the recoverable attribute may also be specified for the data component.

If DATA allocation parameters are not coded, the space specified at the user catalog level is available to contain catalog entries only. The total amount of space specified through DATA and INDEX parameters cannot be greater than that specified at the USERCATALOG level.

DEDICATE
specifies that the unowned and unallocated space (maximum of 16 extents) on VOLUME(volser) is to be owned by the user catalog. (If the catalog is recoverable, VSAM suballocates one cylinder (max-CA) of this space for the CRA.) DEDICATE is mutually exclusive with the space allocation parameters (BLOCKS, RECORDS, TRACKS, CYLINDERS) and can be specified at the catalog level only. Do not specify the ORIGIN parameter with DEDICATE. There must be a contiguous area of space on the volume large enough to contain the primary allocation.

You can specify:

- DEDICATE alone — no space parameters (BLOCKS, TRACKS, RECORDS, or CYLINDERS) are specified at the data and index component levels; in this case the data space suballocated to the catalog itself is calculated by VSAM to be the minimum size catalog and the remaining space is available for later use by VSAM.
- DEDICATE at the catalog level and a space parameter value at both the data component and index component levels; in this case the data space suballocated to the catalog itself, is the sum of the values specified at the data and index component levels. Any VSAM data space that is not suballocated at this time is available for later use by VSAM.
- DEDICATE at the catalog level and a space parameter value at the data component level only; in this case VSAM calculates the space required for the index component and adds this amount to the data component specification. This sum becomes the amount of data space that is suballocated to the catalog. Any VSAM data space that is not suballocated at this time is available for later use by VSAM.

Abbreviation: DED

FOR(*days*) | TO(*date*)
specifies the retention period for the user catalog.

FOR(*days*)
specifies the number of days the user catalog is to be kept. If the number specified is 0 through 1830, the catalog is retained for the number of days specified; if the number is 1831 through 9999 (maximum), the catalog is retained through the year 1999. This value can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

TO(*date*)
specifies the date until which the user catalog is to be kept. The date has the form yyddd, where yy (00 through 99) is the year and ddd (001 through 366) is the day.

Default: The expiration data is set to 0, which means that the user catalog can be deleted whenever it is empty.

IMBED | NOIMBED

specifies whether the sequence set (the lowest level of the index) is to be placed with the data or index component of the catalog. If you specify IMBED (or it is defaulted to) the sequence-set record for each control area is written as many times as it will fit on the first track of the control area. If you specify NOIMBED, the sequence-set records are stored together with the index set (high-level index) in the index component of the catalog. IMBED is the default parameter for the catalog level.

If you specify IMBED or NOIMBED at the index component level, it overrides the specification made (or defaulted to) at the catalog level. A catalog defined with the NOIMBED parameter cannot be processed by OS/VS VSAM or Release 34 (and earlier releases) of DOS/VS VSAM unless you convert the non-imbedded catalog to an imbedded catalog. See "Converting an Imbedded Catalog" in *VSE/VSAM Programmer's Reference* for conversion information.

Abbreviations: IMBD or NIMBD

INDEX (*options*)

specifies attributes of the index component of the catalog. The allocation of space for the data and index components of the user catalog is discussed in "How Data Space is Assigned to a Catalog."

An allocation parameter (BLOCKS, CYLINDERS, RECORDS, or TRACKS) must have been specified as a parameter of DATA for an allocation parameter to be specified as a parameter of INDEX. The total amount of space specified through DATA and INDEX subparameters cannot be greater than that specified at the USERCATALOG level.

The WRITECHECK/NOWRITECHECK parameter defaults to whatever was specified for the user catalog. If the allocation parameters are not specified, VSAM calculates the amount of space for the index component.

Abbreviation: IX

MASTERPW (*password*)

specifies a master password for the user catalog. If MASTERPW is not specified and if other passwords exist, the highest level existing password automatically becomes the master password. The master password allows all operations; it is required to open the catalog as a file. See CONTROLPW for the definition of password.

Abbreviation: MRPW

MODEL (*entryname* [/ *password*]
[ↄ *catname* [/ *password*]])

specifies that an existing user or master catalog is to be used as a model for the user catalog being defined. 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 you specify the MODEL parameter you must specify the following parameters in the DEFINE command even if no attributes of the model are to be changed or added:

- Space allocation parameter (BLOCKS, CYLINDERS, DEDICATE, RECORDS, or TRACKS) at the user catalog level.

- NAME(entryname)
- ORIGIN(tracknumber|blocknumber) - specified if DEDICATE not specified or you do not want the "default ORIGIN".
- VOLUME(volser)

Note that the space suballocated to the catalog itself is controlled by:

- The space allocation attributes previously established by the model catalog, or, by
- The explicit specification of space parameters (in the current DEFINE USERCATALOG command) at the data or index component level (an explicit specification overrides the model attributes).

For more information about MODEL, see "How to Use One Object as a Model for Another Object and Overriding System Defaults" in *VSE/VSAM Programmer's Reference.*

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

password – specifies a password of the catalog to be used as a model if it is password protected. If both the entry password and the catalog password are specified, the catalog password is used. If the protection attributes are to be copied, you must specify the master password. If the protection attributes are not to be copied, you may specify any password.

catname – specifies the name of the catalog to be used as a model. This parameter is required if (1) you are going to specify the password of the catalog that contains the entry instead of specifying the password of the entry itself, or (2) the catalog is not the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

The VOLUME parameter cannot be modeled and must be specified to uniquely identify the volume.

Your job may be terminated because of allocation problems if you use the MODEL parameter and (1) specify a device type different from that specified for the model through the VOLUME parameter, or (2) change the buffer space through the BUFFERSPACE parameter.

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

NOTRECOVERABLE (see RECOVERABLE)

NOWRITECHECK (see WRITECHECK)

ORIGIN(*tracknumber*| *blocknumber*)
specifies the beginning point (track number or block number) of the user catalog's data space. (If the block number does not coincide with the min-CA boundary, VSAM rounds it up to the next min-CA boundary.) VSAM determines the ending point of the data space by adding the value you specify (*at the user catalog level*) for CYLINDERS, BLOCKS, TRACKS, or RECORDS (VSAM converts records and blocks to min-CAs) to the ORIGIN specification. See "Assigning Space to a Catalog Via

the ORIGIN Parameter and a Space Allocation Parameter" for more information.

Specify ORIGIN at the catalog level only. Specify a value greater than 0 for *tracknumber* and a value greater than 1 for *blocknumber*. You can omit the ORIGIN (and DEDICATE) parameter; in this case the beginning point of the catalog's data space is the first available area on the volume that is large enough to contain your primary allocation (if you specified CYLINDERS, the beginning boundary is a cylinder boundary).

Abbreviation: ORG

OWNER (*ownerid*)

identifies the owner of the catalog. The *ownerid* may contain one through 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 also be expressed in hexadecimal (X'ownerid') form.

READPW (*password*)

specifies a read password for the catalog. The read password permits read operations. See CONTROLPW for the definition of password.

Abbreviation: RDPW

RECORDS (*primary* [*secondary*])

specifies the number of records for which space is to be allocated. Each record in the catalog contains 512 bytes. See the CYLINDERS parameter for the information and restrictions that apply to CKD devices. See the BLOCKS parameter for information on FBA devices.

Abbreviation: REC

RECOVERABLE | NOTRECOVERABLE

specifies that a CRA (catalog recovery area) is to be created on the catalog's volume and on each volume owned by the catalog. A CRA occupies one cylinder; it is allocated from space defined for the catalog itself, and from the first data space defined on each new volume.

When the CRA space is filled, the CRA can expand, one cylinder at a time, to include a maximum of 15 additional extents. The number of records a CRA can contain varies with the device type (for example, for a 3330 with 20 control intervals per track, a 16 extent CRA will accommodate approximately 6000 records).

Do not specify RECOVERABLE if the user catalog will contain nonVSAM files.

Abbreviations: RVBL or NRVBL

Default: NOTRECOVERABLE

TO (see FOR)

TRACKS (*primary* [*secondary*])

specifies the number of tracks to be allocated. See the CYLINDERS parameter for more information and restrictions.

Abbreviation: TRK

UPDATEPW (*password*)
specifies an update password for the user catalog. The update password permits read and write operations. See CONTROLPW for the definition of password.

Abbreviation: UPDPW

USERCATALOG (*options*)
specifies that a user catalog is to be defined. USERCATALOG is followed by the parameters specified for the catalog as a whole; they are optionally followed by the DATA and/or INDEX parameters and their subparameters.

Abbreviation: UCAT

VOLUME (*volser*)
specifies the volume that is to contain the catalog. The volume cannot be currently owned by any other VSAM catalog. The volume serial number, *volser*, may contain one through six alphameric, national (@, #, and $), and special characters (commas, blanks, semicolons, parentheses, slashes, asterisks, periods, quotation marks, ampersands, plus signs, hyphens, and equal signs). Single quotation marks within the volume serial number must be coded as two single quotation marks. The volume serial number must be enclosed in single quotation marks if it contains a special character.

For consistency with VSE job control, only alphameric characters should be used.

Abbreviation: VOL

WRITECHECK | NOWRITECHECK
specifies whether to check the data transfer of records written in the user catalog. If WRITECHECK is specified, a record is written and then read, without data transfer, to test for the data check condition. If NOWRITECHECK is specified, a record is written but no checking occurs.

Abbreviations: WCK and NWCK

Default: NOWRITECHECK

DELETE

The DELETE command (DEL) is used to delete entries (for any previously-defined objects) from a catalog. This, in effect, causes the objects to cease to exist. When the entry represents an object that occupies space in a VSAM data space, the object's space is made available for other VSAM objects. See "Using DELETE: Deleting Catalog Entries" and Examples 22 and 23 in "Appendix A: Sample Job Streams" for examples and more information.

The format of the DELETE command is:

DELETE	(*entryname* [/ *password*] [ↄ *entryname* [/ *password*] . . .] | *volser*) [ALTERNATEINDEX | CLUSTER | MASTERCATALOG | NONVSAM | PATH | SPACE | USERCATALOG] [CATALOG (*catname* [/ *password*] [ↄ *dname*])] [ERASE | NOERASE] [FILE (*dname*)] [FORCE | NOFORCE] [PURGE | NOPURGE] [SCRATCH | NOSCRATCH]

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

DELETE Parameters: Summary

The parameters for the DELETE command can be divided into the following categories:

Name, which specifies:

- The entry to be deleted (entryname). **This parameter is required.**
- The type of entry to be deleted (ALTERNATEINDEX, CLUSTER, MASTERCATALOG, NONVSAM, PATH, SPACE, USERCATALOG).
- The name of the catalog containing the entry to be deleted (CATALOG).

Protection and integrity, which specifies:

- That VSAM overwrite with binary zeros, the data component of the alternate index or cluster to be deleted (ERASE, NOERASE).
- That VSAM scratch nonempty VSAM data spaces from the VTOC and give up VSAM volume ownership for the volumes to be deleted (FORCE, NOFORCE).
- That the object is to be deleted even if its retention period has not expired (PURGE, NOPURGE).
- Whether a nonVSAM file is to have its label removed from the VTOC on the volume on which it resides (SCRATCH, NOSCRATCH).

(*entryname* [/ *password*] [ъ *entryname* [/ *password*] . . .] | *volser*)
names the objects to be deleted (and their catalog entries to be removed), or the volume serial number of the volume that contains data spaces to be deleted.

entryname – is the name of the object (other than a data space) to be deleted. This parameter must be the first parameter following DELETE. Entry names for more than one alternate index, cluster, path, or nonVSAM file can be specified (a list of entry names must be enclosed in parentheses); only one catalog name or one volume serial number can be specified at a time. That is, more than one alternate index, cluster, path, or nonVSAM file can be deleted at a time, but only one catalog or the data spaces on only one volume can be deleted at a time.

Data and index components cannot be named for deletion. If a component is to be deleted, the name of the cluster or alternate index to which it belongs must be specified. When a cluster is deleted, its components, alternate indexes, and paths are also deleted. When an alternate index is deleted, its components and paths are also deleted, but its base cluster is not deleted.

password – specifies the master password for a password-protected object. The master password may be specified for each entry name or the catalog's master password may be specified through the CATALOG parameter for the catalog that contains the entries to be deleted. If you are deleting a catalog, the master password must be specified with the entry name.

volser – is the volume serial number of the volume that contains data space(s) to be deleted. Deletion of data space(s) is done by naming the volume on which the data space(s) are defined. If a volume is indicated that contains more than one data space, all empty data spaces are deleted. When all spaces are deleted from a volume, that volume is no longer available to receive VSAM files. To make the volume available once again, you must define space on the volume.

If the catalog that defines the data space(s) is password-protected, specify the update or higher-level password in the CATALOG parameter.

ALTERNATEINDEX | CLUSTER | MASTERCATALOG | NONVSAM | PATH | SPACE | USERCATALOG
specifies the type of object to be deleted. Only objects of one type can be deleted in one DELETE if one of these parameters is specified. If one of these parameters is coded and the object named in *entryname* is not of the specified type, the command is terminated.

ALTERNATEINDEX
specifies that the object to be deleted is an alternate index. It can be deleted only if none of its components are in use by any other partition or system. Deletion of an alternate index causes its data and index components and its related paths to be deleted (its base cluster is unaffected.)

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. When the alternate index's components reside in unique data spaces, the volumes containing these components must be mounted.

When a unique alternate index is deleted, the alternate index entry and the data and index entries are deleted from the catalog and the data space they occupied is also deleted; however, the volume entries will not be automatically deleted from the catalog. If you want to delete the volume entries, specify SPACE in a separate DELETE command.

Abbreviation: AIX

CLUSTER

specifies that the object to be deleted is a cluster. It can be deleted only if none of its components are in use by any other partition or system. Deletion of a cluster causes its data and index components and its related paths and alternate indexes to be deleted. If you want to overwrite a related alternate index with binary zeros, you must delete it individually with the ERASE parameter before you delete the base cluster.

When ERASE is coded, or when the cluster has the ERASE attribute, the volume containing the cluster's data component must be mounted. When the cluster's components reside in unique data spaces, the volumes containing these components must be mounted.

When a unique file is deleted, the cluster entry and the data and index entries are deleted from the catalog and the data space they occupied is also deleted; however, the volume entries will not be automatically deleted from the catalog. If you want to delete the volume entries, specify SPACE in a separate DELETE command.

Abbreviation: CL

MASTERCATALOG

specifies that the object to be deleted is the master catalog. This parameter must be specified to delete the master catalog. It can only be deleted if it is not in use by any other partition (or system) and it is empty, that is, all other objects (except data space entries for the catalog volume) defined in the master catalog (including all user catalogs and the objects defined in them) must have already been deleted.

Abbreviations: MCAT, MRCAT

NONVSAM

specifies that the object to be deleted from the catalog is a nonVSAM file. If the file is on a direct access storage device, its entry will be scratched from the VTOC of the volume(s) on which the file resides unless you specify NOSCRATCH.

Abbreviation: NVSAM

PATH

specifies that the object to be deleted is a path. Deletion of a path does not cause its associated alternate index or base cluster to be deleted.

SPACE

specifies that all empty VSAM data spaces on the volume specified in the *volser* parameter are to be deleted. (This includes data spaces that contain nothing other than catalog recovery area space.) Space must be specified in a separate DELETE command; it cannot be placed in a list that contains an entry type other than itself.

If all the data spaces on the volume have been deleted and the volume is not in the candidate list of any VSAM files, that volume is no longer available to receive VSAM files. That is, the volume record for the volume is removed from the catalog and the VSAM ownership flag in

the Format 4 record of the VTOC is turned off. To make the volume available once again, you must define space on the volume.

If there are nonempty data spaces on the volume or the volume is in a VSAM file's candidate list, VSAM ownership of the volume is retained (the volume remains available).

See the FORCE parameter if you want to delete nonempty data spaces or cause VSAM ownership of the volume to be removed even though the volume is a candidate volume for VSAM files.

Abbreviation: SPC

USERCATALOG
specifies that the object to be deleted is a user catalog entry. This parameter must be specified if you want to delete a user catalog. A user catalog can only be deleted if it is not in use by any other partition (or system) and is empty, that is, all other objects (except data space entries for the catalog volume) defined in the user catalog have already been deleted.

Abbreviation: UCAT

CATALOG (*catname* [/ *password*])
specifies the name of the catalog that contains the entries to be deleted. CATALOG cannot be specified when a user or master catalog that is password-protected is to be deleted; instead, specify the name of the user or master catalog in the *entryname* parameter, together with its master password. CATALOG must be specified when non-empty VSAM data spaces are to be deleted (DELETE SPACE FORCE) and the catalog owning the volume is password-protected.

You do not have to specify CATALOG (unless it is needed for password specification) when the entry to be deleted exists in the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

catname – identifies the catalog that contains the entry to be deleted.

password – specifies the password of the catalog. The master password must be specified if (1) objects (other than data spaces and catalogs) which are password-protected, are to be deleted and their master passwords were not provided with the *entryname* parameter, or (2) data spaces owned by a password-protected catalog are to be deleted with the FORCE parameter. The update or higher-level password must be specified if data spaces owned by a password-protected catalog are to be deleted without the FORCE parameter. Otherwise, the password parameter is not required.

Abbreviation: CAT

CLUSTER (see ALTERNATEINDEX)

ERASE | NOERASE
specifies whether the data component of the alternate index or cluster to be deleted is to be erased, that is, overwritten with binary zeros. This parameter overrides whatever was coded when the alternate index or cluster was defined or last altered.

This parameter cannot be specified for data spaces, catalogs, nonVSAM files, or paths.

ERASE
specifies that the data component is to be overwritten with binary zeros when the alternate index or cluster 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 alternate index or cluster is deleted.

Abbreviations: ERAS and NERAS

FORCE | NOFORCE
specifies (1) whether nonempty VSAM data spaces are to be scratched (their labels removed) from the VTOC, and VSAM volume ownership given up and (2) whether VSAM volume ownership should be given up if the volume is in the candidate list of any VSAM files.

FORCE
specifies that nonempty VSAM data spaces are to be scratched from the VTOC and that VSAM volume ownership removal is to be forced even if the volume is in the candidate list of any VSAM files. If there are any VSAM files which have space on the volume, their catalog entries are marked unusable (because owned space has been deleted). If any VSAM files list the volume as a candidate volume, the volume is removed from the candidate list and, if the volume is (a) the recovery volume, or (b) the only volume owned by an unallocated (NOALLOCATION parameter) component, then the catalog entries are marked unusable.

Note: A flagged unusable file can be opened, but a warning error code X'60' results. A flagged unusable file cannot be successfully exported, but it can be retrieved via the PRINT and REPRO commands.

If any of the files having space on the volume or having the volume in its candidate list are open, no data spaces are scratched, no files are marked unusable, and the command is rejected.

If you specify FORCE for a volume that contains a VSAM catalog, the command is rejected. FORCE can only be specified for VSAM data spaces. The master password of the catalog owning the volume must be specified in the CATALOG parameter if the catalog is password protected.

NOFORCE
specifies that nonempty VSAM data spaces are not to be scratched from the VTOC and that VSAM volume ownership is not to be removed if the volume is in the candidate list of any VSAM files.

Abbreviations: FRC and NFRC

Default: NOFORCE

MASTERCATALOG (see ALTERNATEINDEX)

NOERASE (see ERASE)

NOFORCE (see FORCE)

NONVSAM (see ALTERNATEINDEX)

NOPURGE (see PURGE)

NOSCRATCH (see SCRATCH)

PATH (see ALTERNATEINDEX)

PURGE | NOPURGE
specifies whether the alternate index, cluster, catalog, nonVSAM file, or path is to be deleted regardless of its retention period.

PURGE
specifies that the object is to be deleted even if its retention period, defined in the TO or FOR parameter, has not expired.

NOPURGE
specifies that the object is not to be deleted if its retention period has not expired.

Abbreviations: PRG and NPRG

Default: NOPURGE

SCRATCH | NOSCRATCH
specifies whether a nonVSAM DASD file is to be scratched (its label removed from the VTOC) from the volume on which it resides.

SCRATCH
specifies that the nonVSAM DASD file being deleted is to have its label(s) removed from the VTOC of the volume(s) on which it resides; that is, both the catalog entry and the file are to be deleted.

NOSCRATCH
specifies that the file being deleted is to keep its label in the VTOC.

Abbreviations: SCR and NSCR

Default: SCRATCH

SPACE (see ALTERNATEINDEX)

USERCATALOG (see ALTERNATEINDEX)

EXPORT

The EXPORT command (EXP) can be used to move clusters, alternate indexes, and user catalogs from one system (or set of systems in a DASD Sharing environment) to another (with the help of IMPORT), 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 a master catalog, a nonVSAM file, a data space, or a path, or to provide a backup copy of a catalog. However, all paths are automatically exported along with their related base cluster or alternate index. You cannot export an empty cluster or alternate index.

See "Using EXPORT/IMPORT: Transporting or Backing Up Files" for more information and Examples 10-12 in "Appendix A: Sample Job Streams."

The format of the EXPORT command when it is used to move or disconnect a user catalog is:

```
EXPORT    entryname [ / password ]
          DISCONNECT
```

The format of the EXPORT command when it is used to move a cluster or alternate index is:

```
EXPORT    entryname [ / password ]
          INFILE( dname )
          OUTFILE
             ( dname
             [ŏENVIRONMENT
               ([ŏBLOCKSIZE( size )]
               [ŏNOLABEL|STDLABEL]
               [ŏNOREWIND|REWIND|UNLOAD]
               [ŏPRIMEDATADEVICE( devtype )])]
             )
          [CIMODE|RECORDMODE]
          [ERASE|NOERASE]
          [INHIBITSOURCE|NOINHIBITSOURCE]
          [INHIBITTARGET|NOINHIBITTARGET]
          [PURGE|NOPURGE]
          [TEMPORARY|PERMANENT]
```

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

EXPORT Parameters

entryname [/ *password*]

entryname– names the cluster, alternate index, or user catalog to be exported. **This parameter must be the first parameter following EXPORT.** Data and index components cannot be individually exported; the name of their associated cluster or alternate index must be specified. A path name cannot be used to define a cluster or alternate index.

Entryname (if not a user catalog name) must exist in the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

password– if you are exporting a password protected cluster or alternate index, you must supply the master password of the cluster or alternate index. If any password protected paths are defined over the cluster or alternate index being exported, you must supply the master password of the master catalog in order to export the protection attribute of all the paths.

If you are exporting a user catalog, you must supply the update or higher-level password of the master catalog, if the master catalog is password protected.

CIMODE | RECORDMODE
specifies whether the data component of a VSAM file is to be copied to a portable file in control-interval mode or logical-record mode. You must specify RECORDMODE if you are exporting the portable copy to either an OS/VS system or a DOS/VS Release 34, or earlier, system.

CIMODE
specifies that VSAM control interval access is to be used. CIMODE enhances export performance and is the preferred option.

RECORDMODE
specifies that VSAM logical record access is to be used. This option should only be used when exporting to either an OS/VS system or a DOS/VS Release 34, or earlier, system.

Abbreviations: CIM or RECM

DISCONNECT
must be specified if a user catalog is to be exported. The entry for the user catalog will be deleted from this system's master catalog. If other VSE systems are sharing this master catalog, the user catalog is disconnected from those systems also. The volume that contains the user catalog must be physically moved to the system (or systems) to which the catalog will be imported. To make a user catalog available in other systems *and* in the original system, don't export, it but code the IMPORT CONNECT command to import it to each system in which it is to be available.

A user catalog cannot be disconnected if it is currently open (the catalog or one of its files is being processed by another program in this system or a DASD Sharing system).

Abbreviation: DCON

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 cluster or alternate index was defined or last altered. ERASE or NOERASE can be specified only if the file is to be permanently exported (that is, deleted from the catalog). Note that if you permanently export a cluster, associated alternate indexes (which are implicity deleted) are not erased even if you specify ERASE (or ERASE was specified when the alternate indexes were defined or last altered). One of the ways to cause alternate indexes to be erased is to permanently export (or explicitly delete) them before permanently exporting the base cluster.

Abbreviations: ERAS and NERAS

INHIBITSOURCE | NOINHIBITSOURCE
specifies whether the original cluster or alternate index can be updated. This specification can later be altered with the ALTER command (see INHIBIT, UNINHIBIT).

INHIBITSOURCE
specifies that the original cluster or alternate index can only be read.

The data and index entries in the catalog are modified to indicate update inhibited. When the file is re-imported, the inhibit update flags are reset (unless the copy to be imported was created with the INHIBITTARGET specification).

If INHIBITSOURCE is specified, TEMPORARY must be specified.

NOINHIBITSOURCE
specifies that the original cluster or alternate index can be accessed for any kind of operation and is not to be modified to "inhibit update." The catalog inhibit indicators are not reset.

Abbreviations: INHS and NINHS

Default: NOINHIBITSOURCE

INHIBITTARGET | NOINHIBITTARGET
specifies whether the exported copy of the file can be updated after it has been imported. This specification can later be altered with the ALTER command (see INHIBIT, UNINHIBIT).

INHIBITTARGET
specifies that the exported cluster or alternate index is to be read-only (cannot be updated) after it has been imported into the receiving system. The imported catalog's data and index entries are flagged inhibit update. This imported version of the file will remain flagged "inhibit update" until its components are reset using the UNINHIBIT parameter of the ALTER command.

NOINHIBITTARGET
specifies that the exported copy of the file can be accessed for any type of operation after it has been imported and is not to be modified to "inhibit update."

Abbreviations: INHT and NINHT

Default: NOINHIBITTARGET

NOERASE (see ERASE)

NOINHIBITSOURCE (see INHIBITSOURCE)

NOINHIBITTARGET (see INHIBITARGET)

NOPURGE (see PURGE)

OUTFILE(*dname* [ðENVIRONMENT(*subparameters*)])
dname specifies the *filename* of the DLBL or TLBL job control statement that identifies the output file (portable file) to be created as a result of the EXPORT command. **This is a required parameter for exporting files.** If an unlabeled tape is used (NOLABEL option), a TLBL statement is not required. However, since ***dname*** is a required positional parameter, a dummy (ficticious) ***filename*** of your choosing must be specified.

Abbreviation: OFILE

ENVIRONMENT (*subparameters*)
describes the parameters of the portable copy.

Abbreviation: ENV

BLOCKSIZE (*size*)
specifies the block size of the portable copy. For CIMODE, it is recommended that this value be at least 8 bytes larger than the file's data component control interval size. (For enhanced performance.)

Abbreviation: BLKSZ

Default: 2048 bytes per block

NOLABEL | STDLABEL
specifies the type of tape (unlabeled or EBCDIC standard-labeled) which contains the portable file, if PRIMEDATADEVICE specifies 2400.

NOLABEL
indicates an unlabeled tape is to be processed. You must properly position the tape to the file you want to process. For an output file, Access Method Services does not write a tapemark as the first record. Instead, a data record is written immediately, wherever the tape is positioned. If an output file is to begin somewhere in the middle of the reel, it is your responsibility to position the tape immediately past the ending tapemark of the preceeding file. Use the MTC job control statement or command (refer to *VSE Advanced Functions*) to properly position the tape or to write a tapemark.

If VSE/Access Control is installed, REWIND must be specified for unlabeled tapes.

Abbreviation: NLBL

Restriction: If a portable-copy tape file is unlabeled, it must be contained on a single volume, because IMPORT cannot handle a multivolume unlabeled file.

STDLABEL
indicates EBCDIC standard-labeled tapes are to be processed.

Abbreviation: SLBL

Default: STDLABEL

Restriction: Access Method Services does not support ASCII files or nonstandard labels.

NOREWIND | REWIND | UNLOAD
specifies the tape positioning action for an OPEN, CLOSE, and EOV (end-of-volume) condition, if PRIMEDATADEVICE specifies 2400.

Note: Be aware, when using the REWIND or UNLOAD options (especially with multi-file volumes), that any OPEN causes the tape to be positioned at load point. You must be sure that the file you want to process is really the first one (at load point) on the reel. Otherwise, specify the NOREWIND option and use the MTC job control statement (see *VSE/Advanced Functions*), for example, to properly position the tape.

NOREWIND
specifies that rewind is never to be performed on OPEN, CLOSE, and EOV.

End-of-file considerations: For EBCDIC standard-labeled tapes, the tape is positioned between the two trailing tapemarks. For unlabeled tapes, the tape is positioned following the single trailing tapemark.

Abbreviation: NREW

Default: NOREWIND

REWIND
specifies that tapes are to be rewound to load point but not unloaded on an OPEN, CLOSE, or EOV condition.

Abbreviation: REW

UNLOAD
specifies that tapes are rewound on an OPEN, and rewound and unloaded on an EOV and CLOSE condition.

Abbreviation: UNLD

PRIMEDATADEVICE(*devtype*)
Specify this parameter only if the portable copy resides on a tape device (in which case you specify 2400). The 2400 identifies all tape devices that DTFMT supports. You must specify SYS005 in the ASSGN statement for a tape output file. See "Magnetic Tape Considerations for Access Method Services" in *VSE/VSAM Programmer's Reference* for more information on tape.

Abbreviation: PDEV

PERMANENT (see TEMPORARY)

PURGE | NOPURGE
specifies whether the cluster or alternate index (not a user catalog) to be permanently exported is to be deleted from the original system(s) regardless of the retention period specified in the TO or FOR parameter when the cluster or alternate index was defined. PURGE or NOPURGE can be specified only if the file is to be permanently exported (that is, deleted from the original system(s)).

PURGE
specifies that the cluster or alternate index is to be deleted even if the retention period has not expired.

NOPURGE
specifies that the cluster or alternate index is not to be deleted if the retention period has not expired.

Abbreviations: PRG and NPRG

Default: NOPURGE

TEMPORARY | PERMANENT
specifies whether the cluster or alternate index to be exported is to be deleted from the original system(s).

TEMPORARY
specifies that the cluster or alternate index is not to be deleted from the original system(s). The cluster or alternate index in the original system(s) is marked as temporary to indicate that another copy exists and that the original copy can be replaced (via IMPORT).

PERMANENT
specifies that the cluster or alternate index is to be deleted from the original system(s). The catalog entry is deleted and the storage space used by the cluster or alternate index is freed. If a cluster or alternate index is to be deleted from the original system(s) and its retention period has not expired, PURGE must be coded. When a base cluster is exported, its components, alternate indexes, and paths are also deleted. When an alternate index is exported, its components and path are also deleted, but its base cluster is not deleted.

Abbreviations: TEMP and PERM

Default: PERMANENT

EXPORTRA

The EXPORTRA command (XPRA) is used to recover VSAM catalog entries and data using catalog recovery areas (CRAs) to open files rather than the catalog itself. For files cataloged in a recoverable catalog, critical catalog information is recorded in CRAs on each owned volume. If a file (alternate index or cluster) is not addressable by way of the catalog, the EXPORTRA command can be used to gain access to the file through the CRA to create a copy of the catalog entries and data which can be introduced back into a new catalog by the IMPORTRA command. In the case of a user catalog that is not addressable by way of its owning catalog, its pointer is recovered from the CRA residing on the owning catalog's volume. For information about the contents of CRAs, see "Catalog Recovery Area Contents" in *VSE/VSAM Programmer's Reference.*

The EXPORTRA command can be used to recover all of the catalog entries and files on one or more volumes, or selected catalog entries and files on one or more volumes. To do a selective recovery, use the LISTCRA command to determine catalog entries and file names and their associated volumes. See "Using EXPORTRA/IMPORTRA Recovering Catalog Entries and Data" for examples and more information and Example 20 in "Appendix A: Sample Job Streams."

The format of the EXPORTRA command is:

```
EXPORTRA   CRA( ( dname1 [ðALL[ðINFILE( dname2 )]|
              ðNONE|
              ðENTRIES( ( entryname [ð dname3 ] ) ... ) }
           CRAVOLUMES( ( volser{ALL|NONE|
              ENTRIES( entryname [ð...] ) }) ... )
           OUTFILE
              ( dname
              [ðENVIRONMENT
                ( [ðBLOCKSIZE( size )]
                [ðNOLABEL|STDLABEL]
                [ðNOREWIND|REWIND|UNLOAD]
                [ðPRIMEDATADEVICE( devtype ) ] ) ]
              )
           [CIMODE|RECORDMODE]
           [FORCE|NOFORCE]
           [MASTERPW( password ) ]
```

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

EXPORTRA Parameters

CRAVOLUMES((*volser* {ALL|NONE|
ENTRIES(*entryname* [ð...]) }) ...)

specifies the volume serial numbers of the volumes that contain the CRAs and the catalog entries and files to be recovered. The parameter sublist is specified for each CRA to be accessed, whether or not any catalog entries or files are to be recovered from it.

The maximum repetition for CRAVOLUMES subparameters is 20.

volser – specifies the volume serial number of the volume that contains the CRA from which catalog entries and files are to be recovered.

All volumes specified must be owned by the same catalog (it can be any user catalog or the master catalog). Volumes identified by *volser* need not be premounted (VSAM will issue a mount request).

Abbreviation: CRAVOL

ALL
indicates that all of the catalog entries (and their associated file data) recorded in the CRA on the volume indicated by *volser* are to be recovered. This is the default.

NONE
indicates that none of the files recorded in this volume's CRA are to be recovered. The CRAs for all volumes in all multivolume files that are to be recovered must be specified as a *volser* parameter. See "Using EXPORTRA For Entries on Multiple Volumes" for further information and an example of the NONE parameter.

Be aware that the CRA for an alternate index is always on the same volume as its base cluster even though the alternate index can be on a different volume.

ENTRIES(*entryname* [ƀ...]...)
specifies which entries are to be recovered from this CRA. The maximum repetition for ENTRIES subparameters is 100.

entryname – specifies the name of the alternate index, cluster, user catalog entry, or nonVSAM entry for which catalog records and (if applicable) data are to be recovered, or the name of the user catalog whose catalog records are to be recovered. See "Using EXPORTRA For Selected Entries" for further information and an example of the ENTRIES subparameter. Note that empty clusters and alternate indexes (including model objects) can be exported.

Abbreviation: ENT

CIMODE | RECORDMODE
specifies whether the data component of a VSAM file is to be copied to a portable file in control-interval mode or logical-record mode. You must specify RECORDMODE if you are exporting the portable copy to either an OS/VS system or a DOS/VS Release 34, or earlier, system.

CIMODE
specifies that VSAM control interval access is to be used. CIMODE enhances export performance and is the preferred option.

RECORDMODE
specifies that VSAM logical record access is to be used. This option should only be used when exporting to either an OS/VS system or a DOS/VS Release 34, or earlier, system.

Abbreviation: CIM or RECM

FORCE | NOFORCE
specifies what action is to be taken if an out-of-synchronization condition is detected in recovering an entry. (Out-of-synchronization is when two or more volumes of a multivolume file have inconsistent extent information.)

An out-of-synchronization situation can occur in VSE only under the following conditions:

1. A VSAM file that was out-of-synchronization on an OS/VS1 or OS/VS2 system is introduced into a VSE system.
2. A VSAM file created on an OS/VS1 or OS/VS2 system is introduced into a VSE system and extended.

Under either of these two situations, FORCE must be specified to cause EXPORTRA to process the entry.

FORCE
specifies that recovery should be attempted. *Warning:* The CRA extent information is needed to recover data; if this information is incorrect, the result may be I/O errors or wrong data.

NOFORCE
specifies that recovery is not to be attempted. EXPORTRA bypasses the out-of-synchronization file.

Abbreviations: FRC and NFRC

Default: NOFORCE

MASTERPW (*password*)
specifies the master password of the master catalog; it must be specified if the master catalog is password protected.

Abbreviation: MRPW

OUTFILE (*dname* [ƀENVIRONMENT (*subparameters*)])
dname specifies the *filename* of the DLBL or TLBL statement that identifies the output file (portable copy) in which the recovered files and/or associated CRA catalog entries are to be written. **This is a required parameter.**

If an unlabeled tape is used (NOLABEL option), a TLBL statement is not required. However, since *dname* is a required positional parameter, a dummy (ficticious) *filename* of your choosing must be specified.

Abbreviation: OFILE

ENVIRONMENT (*subparameters*)
describes the parameters of the portable copy.

Abbreviation: ENV

BLOCKSIZE (*size*)
specifies the block size of the portable copy. If you specify CI-MODE, it is recommended (for enhanced performance) that this value be at least 12 bytes larger than the largest data component control interval size being exported.

Abbreviation: BLKSZ

Default: 2048 bytes per block

NOLABEL | STDLABEL

specifies the type of tape (unlabeled or EBCDIC standard-labeled) which contains the portable file, if PRIMEDATADEVICE specifies 2400.

NOLABEL

indicates an unlabeled tape is to be processed. You must properly position the tape to the file you want to process. For an output file, Access Method Services does not write a tapemark as the first record. Instead, a data record is written immediately, wherever the tape is positioned. If an output file is to begin somewhere in the middle of the reel, it is your responsibility to position the tape immediately past the ending tapemark of the preceeding file. Use the MTC job control statement or command (refer to *VSE/Advanced Functions*) to properly position the tape or to write a tapemark.

If VSE/Access Control is installed, REWIND must be specified for unlabeled tapes.

Abbreviation: NLBL

Restriction: If a portable-copy tape file is unlabeled, it must be contained on a single volume, because IMPORTRA cannot handle a multivolume unlabeled file.

STDLABEL

indicates EBCDIC standard-labeled tapes are to be processed.

Abbreviation: SLBL

Default: STDLABEL

Restriction: Access Method Services does not support ASCII files or nonstandard labels.

NOREWIND | REWIND | UNLOAD

specifies the tape positioning action for an OPEN, CLOSE, and EOV (end-of-volume) condition if PRIMEDATEDEVICE specifies 2400. (Prior to DOS/VS Release 34, IMPORTRA and EXPORTRA were the only tape processing commands that rewound a tape on an OPEN, CLOSE, or EOV condition. Starting with DOS/VS Release 34, IMPORTRA and EXPORTRA are consistent with the other tape processing commands, that is, NOREWIND is the default for the IMPORTRA and EXPORTRA commands.)

Note: Be aware, when using the REWIND or UNLOAD options (especially with multi-file volumes), that any OPEN causes the tape to be positioned at load point. You must be sure that the file you want to process is really the first one (at load point) on the reel. Otherwise, specify the NOREWIND option and use the MTC job control statement (see *VSE/Advanced Functions*), for example, to properly position the tape.

NOREWIND

specifies that rewind is never to be performed on OPEN, CLOSE, and EOV.

End-of-file considerations: For EBCDIC standard-labeled tapes, the tape is positioned between the two trailing tapemarks. For unlabeled tapes, the tape is positioned following the single trailing tapemark.

Abbreviation: NREW

Default: NOREWIND

REWIND
specifies that tapes are to be rewound to load point but not unloaded on an OPEN, CLOSE, or EOV condition.

Abbreviation: REW

UNLOAD
specifies that tapes are to be rewound on an OPEN, and rewound and unloaded on an EOV and CLOSE condition.

Abbreviation: UNLD

PRIMEDATADEVICE(*devtype*)
Specify this parameter only if the portable copy resides on a tape device (in which case you specify 2400). The 2400 identifies all tape devices that DTFMT supports. You must specify SYS005 in the ASSGN statement for tape output files. See "Magnetic Tape Considerations for Access Method Services" in *VSE/VSAM Programmer's Reference* for more information on tape.

Abbreviation: PDEV

IMPORT

The IMPORT command (IMP) is used in conjunction with the EXPORT command to move clusters, alternate indexes and user catalogs from one system (or set of systems in a DASD Sharing environment) to another, to restore backup copies of clusters and alternate indexes, and to connect user catalogs to the master catalog. In the process of moving an entry, you may choose to change some of its attributes.

The IMPORT command cannot be used to move a master catalog, a nonVSAM file, a data space or a path. However, all paths exported with their related object are automatically redefined.

See "Using EXPORT/IMPORT: Transporting or Backing Up Files" for more information and Examples 13-16 in "Appendix A: Sample Job Streams."

The format of the IMPORT command, when it is used to move or connect a user catalog is:

```
IMPORT   CONNECT
         OBJECTS ( ( entryname ŏDEVICETYPE ( devtype )
            ŏVOLUMES ( volser ) ) . . . )
         [CATALOG ( catname [ /password ] ) ]
```

The format of the IMPORT command, when it is used to move or restore a cluster or alternate index, is:

```
IMPORT   INFILE
            ( dname
            [ŏENVIRONMENT
              ( [ŏBLOCKSIZE ( size ) ]
              [ŏNOLABEL | STDLABEL]
              [ŏNOREWIND | REWIND | UNLOAD]
              [ŏPRIMEDATADEVICE ( devtype ) ] ) ]
            )
         OUTFILE ( dname [ / password ] )
         [CATALOG ( catname [ / password ] ) ]
         [ERASE | NOERASE]
         [OBJECTS ( ( entryname
            [ŏFILE ( dname ) ]
            [ŏKEYRANGES ( ( lowkeyŏ highkey )
              [ŏ ( lowkeyŏ highkey ) . . . ] ) ]
            [ŏNEWNAME ( newname ) ]
            [ŏORDERED | UNORDERED]
            [ŏUSECLASS ( primary [ŏ secondary ] ) ]
            [ŏVOLUMES ( volser [ŏ volser ] . . . ) |
              DEFAULTVOLUMES] ) . . . ) ]
         [OUTPW(password)]
         [PURGE | NOPURGE]
```

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

CATALOG (*catname* [/ *password*])
specifies the name and the password of the catalog in which the imported cluster or alternate index is to be defined or to which the imported user catalogs are to be connected. This parameter is required only when the catalog is password protected or when you want to direct the imported object to a catalog other than the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

catname – is the name of the catalog.

password – specifies the update or higher-level password of the catalog. If the object being imported is an alternate index over a password-protected base cluster, you should specify the catalog's master password to avoid being prompted for the base cluster's master password.

Abbreviation: CAT

CONNECT
specifies that one or more user catalogs are to be imported. User catalogs are connected to the master catalog in the receiving system(s).

CONNECT must be specified when user catalogs are to be imported. OBJECTS must also be specified to name the catalogs and to identify the volume serial number and device type of the volume that contains them.

In a DASD Sharing environment where the master catalog is not shared, use IMPORT CONNECT to connect newly-defined user catalogs to the master catalogs in the other systems which are to share them.

Abbreviation: CON

ERASE | NOERASE
specifies whether the data component of a previously temporarily exported cluster or alternate index is to be erased (that is, overwritten with binary zeros) when replaced by the cluster or alternate index to be imported. The object to be imported has the same name as the object it is replacing. The erase operation is actually performed by VSAM. This parameter overrides whatever was specified when the cluster or alternate index was defined or last altered.

ERASE or NOERASE is specified only if the cluster or alternate index is being imported back into the catalog from which it was temporarily exported.

Note that if you import a cluster causing the old temporarily exported cluster to be deleted, any alternate indexes associated with the old cluster will be implicitly deleted (but not erased). One of the ways to erase the alternate indexes is to explicitly erase them with DELETE before importing the base cluster.

Abbreviations: ERAS and NERAS

INFILE (*dname* [ENVIRONMENT (*subparameters*)])
specifies the *filename* of the DLBL or TLBL statement that identifies and describes the portable copy of the cluster or alternate index to be imported. This copy is used to create the new version of the cluster or alternate index in the receiving system. **This is a required parameter for importing files.** If

an unlabeled tape is used (NOLABEL option), a TLBL statement is not required. However, since *dname* is a required positional parameter, a dummy (ficticious) *filename* of your choosing must be specified.

Abbreviation: IFILE

ENVIRONMENT(*subparameters*)
describes the parameters of the portable copy.

Abbreviation: ENV

BLOCKSIZE(*size*)
specifies the block size of the portable copy created by EXPORT.

Abbreviation: BLKSZ

Default: 2048 bytes per block.

NOLABEL | STDLABEL
specifies the type of tape (unlabeled or EBCDIC standard-labeled) to contain the portable file, if PRIMEDATADEVICE specifies 2400.

NOLABEL
indicates an unlabeled tape is to be processed. Since the first record of an unlabeled tape may or may not be preceded by a tapemark (user's prerogative), you must take care to properly position the tape to the file you want to process. If a tapemark precedes the file, position the tape either immediately past or immediately before the tapemark. If no tapemark is present, position the tape immediately before the first data record. If the unlabeled tape was created by VSE EXPORT there is no tapemark preceding the first record of the file.

If VSE/Access Control is installed, REWIND must be specified for unlabeled tapes.

Abbreviation: NLBL

Restriction: If a portable-copy tape file is unlabeled, it must be contained on a single volume, because IMPORT cannot handle a multivolume unlabeled file.

STDLABEL
indicates EBCDIC standard-labeled tapes are to be processed.

Abbreviation: SLBL

Default: STDLABEL

Restriction: Access Method Services does not support nonstandard labels.

NOREWIND | REWIND | UNLOAD
specifies the tape positioning action for an OPEN, CLOSE, and EOV (end-of-volume) condition if PRIMEDATADEVICE specifies 2400.

Note: Be aware, when using the REWIND or UNLOAD options (especially with multi-file volumes), that any OPEN causes the tape to be positioned at load point. You must be sure that the file you want to process is really the first one (at load point) on the reel. Otherwise, specify the NOREWIND option and use the MTC job control statement (see *VSE/Advanced Functions*), for example, to properly position the tape.

NOREWIND
specifies that rewind is never to be performed on OPEN, CLOSE, and EOV.

End-of-file considerations: For EBCDIC standard-labeled tapes, the tape is positioned between the two trailing tapemarks. For unlabeled tapes, the tape is positioned following the single trailing tapemark.

Abbreviation: NREW

Default: NOREWIND

REWIND
specifies that tapes are to be rewound to load point but not unloaded on an OPEN, CLOSE, or EOV condition.

Abbreviation: REW

UNLOAD
specifies that tapes are to be rewound on an OPEN, and rewound and unloaded on an EOV and CLOSE condition.

Abbreviation: UNLD

PRIMEDATADEVICE(*devtype*)
specify this parameter only if the file resides on a tape device (in which case you specify 2400). The 2400 identifies all tape devices that DTFMT supports. You must specify SYS004 in the ASSGN statement for a tape input file. See "Magnetic Tape Considerations for Access Method Services" in *VSE/VSAM Programmer's Reference* for more information on tape.

Abbreviation: PDEV

NOERASE (see ERASE)

NOPURGE (see PURGE)

- For a user catalog:

```
OBJECTS ( ( entryname
  DEVICETYPE ( devtype )
  VOLUMES ( volser ) ) . . . )
```

- For a cluster or alternate index:

```
OBJECTS ( ( entryname
  [ЪFILE ( dname ) ]
  [ЪKEYRANGES ( ( lowkeyЪ highkey )
    [Ъ ( lowkeyЪ highkey ) . . . ] ) ]
  [ЪNEWNAME ( newname ) ]
  [ЪORDERED | UNORDERED]
  [ЪUSECLASS ( primary [Ъ secondary ] ) ]
  [ЪVOLUMES ( volser [Ъ volser . . . ] ) |
    DEFAULTVOLUMES ] ) . . . )
```

OBJECTS (*entryname*)
specifies attributes for the cluster, alternate index, their data and index components, or the path or user catalog to be imported. Attributes may be specified for multiple objects by repeating the parameter list for each object.

By repeating the OBJECTS parameter set (maximum of 20 times) 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 device type different from that originally containing the cluster, alternate index, or user catalog, the job may be terminated because of allocation problems due to different cylinder and track capacities. See "Using EXPORT/IMPORT: Transporting or Backing Up Files" for techniques to ensure device independence.

entryname – specifies the name of the data component, index component, cluster, alternate index, path, or user catalog whose attributes are being specified.

Abbreviation: OBJ

DEVICETYPE(*devtype*)
specifies the device type (2314, 2319, 3330, 3330-11, 3340, 3350, or FBA) of the volume that contains a user catalog that is to be imported.

Abbreviation: DEVT

FILE(*dname*)
specifies the *filename* of the DLBL statement that, together with associated EXTENT statement(s), identifies the volume(s) and extents allocated to the data and index components of an alternate index or cluster that is to be imported.

This parameter is required only when the data or index components (of an alternate index or cluster) are defined with the UNIQUE attribute. (If only one component is unique you code one FILE parameter; if both components are unique you code two FILE parameters)

The symbolic unit can be omitted from the EXTENT statement.

Do not specify the FILE parameter when importing into a predefined empty file.

Note: This parameter explanation applies to the job control simplification introduced in Release 2. If you are using Release 1 job control, refer to the appropriate explanation in Appendix J.

KEYRANGES((*lowkey*b*highkey*) [b(*lowkey*b*highkey*) . . .])
specifies portions of key-sequenced data to be placed on separate volumes. The data is divided by key among the volumes specified in VOLUMES. If a volume serial number was duplicated in VOLUMES, multiple key ranges (of a suballocated object) 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 low-key/high-key pairs is 20. Key ranges may not overlap and must be in ascending order; gaps may exist within a specified set of ranges, but records cannot be inserted within a gap.

Each subparameter can contain 1 to 64 characters. All EBCDIC characters are allowed. Subparameters (keys) 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 coded (X'key'). See *VSE/VSAM Programmer's Reference* for more information.

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.

Abbreviation: KRNG

Restrictions: May be specified only with key-sequenced clusters or alternate indexes or their data components. If KEYRANGES is specified for a unique object, each key range must reside on a separate volume.

NEWNAME (*newname*)

specifies the new name of an imported cluster or alternate index or its components, or of a path. You cannot change the name of an imported catalog. The new name may contain 1 to 44 alphameric, national (@, #, and $), and special (hyphen and 12-0 overpunch) characters. Names that contain more than eight characters must be segmented by periods; one through eight characters may be specified between periods. The first character of the name or name segment must be either an alphabetic character or a national character.

If you are importing a cluster or alternate index into a predefined empty file with a different file-ID, you must use the NEWNAME parameter to rename the file-ID of the cluster or alternate index to match the file-ID of the predefined empty file.

If you are importing an object back into the catalog from which it was exported with the TEMPORARY option, you must rename the object and each of its components and paths unless (a) you specify PURGE to delete the old copy, or (b) the old copy has reached its expiration date or was defined without an expiration date.

Abbreviation: NEWNM

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, and so forth. If it is impossible to allocate volumes in the given order and ORDERED is specified, the command is terminated.

Abbreviations: ORD and UNORD

USECLASS (*primary* [ƀ *secondary*])

specifies that the class of an imported object (cluster, alternate index, and their data and index components) is to be modified. The USECLASS specifications for a cluster and alternate index are modified in the following order; cluster or alternate index; data component; and index component. For example, if you specify a USECLASS value for both the data component and cluster (in either order) of an imported object, (1) the value specified for the cluster level will also be propagated to the data component level and (if present) index component level, (2) the value specified for the data component will be inserted into the data component, superceding the previous data component value derived from the cluster specification, (3) the index component (if present) would retain the propagated cluster USECLASS value.

If you specify only a primary USECLASS value, the default secondary value ('P' - same as primary) is applied to the imported object. This effectively overrides the secondary value for that imported object in the portable file.

If you specify primary USECLASS values other than 0 for a unique cluster or alternate index an error occurs.

Abbreviation: USCL

VOLUMES (*volser* [ƀ *volser* . . .]) | DEFAULTVOLUMES

specifies the volumes on which the cluster, alternate index, data component or index component is to reside or on which the user catalog resides.

VOLUMES

If the original volume(s) from which the cluster or alternate index was exported is to be the receiving volume(s), VOLUMES is not required. If VOLUMES is specified at the cluster or alternate index level, it will propagate to the data and index levels. A data or index level specification will override a cluster or alternate-index level specification.

volser – contains one through 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. For consistency with VSE job control, only alphameric characters should be used.

VOLUMES must be specified when a user catalog is to be imported (only one volume). VOLUMES or DEFAULTVOLUMES must be specified when the object was permanently exported from DOS/VS Release 30 or earlier, from OS/VS1 Release 3.1 or earlier, and OS/VS2 Release 3 or earlier.

All volumes for a component of a cluster or alternate index must be of the same device type.

DEFAULTVOLUMES

indicates that (for the specified cluster, alternate index, data or index component) VSAM is to disregard the exported volume list(s) and, instead, is to obtain a new list of volume(s). VSAM obtains the new volume(s) list from the data and index (if present) components of the applicable default models in the receiving catalog. The default model used corresponds to the type of file being imported (alternate index, key-sequenced file, etc.).

An error message is issued if you specify DEFAULTVOLUMES and VOLUMES together in the same *entryname* list. You can specify DEFAULTVOLUMES at any level and in any combination of levels; if specified at the alternate index or cluster level, DEFAULTVOLUMES propagates to the data component level and (if applicable) the index component level. (If DEFAULTVOLUMES is propagated to the data or index component level where the VOLUMES parameter has been specified, the VOLUMES parameter takes precedence.)

DEFAULTVOLUMES cannot be specified for components which are defined with the ORDERED or UNIQUE parameters.

Abbreviations: DFVOL and VOL

Default: If neither VOLUMES nor DEFAULTVOLUMES is specified, the original volume(s) from which the object was exported is used.

OUTPW (*password*)

specifies the password of the file to receive the portable copy.

password— is required only if importing into a predefined empty file that has passwords that are different from the old passwords of the file being imported. You must specify the file's update or higher-level password.

Abbreviation: OPW

PURGE | NOPURGE

specifies whether the entry for the original cluster or alternate index is to be deleted and replaced by a new entry with the same name regardless of the retention time specified in the TO or FOR parameter in the original definition.

PURGE

specifies that the entry for the original cluster or alternate index is to be deleted even if the retention period has not expired. This parameter can be used only when you are importing the cluster or alternate index into the original catalog from which it was exported with the TEMPORARY parameter.

NOPURGE

specifies that the entry for the original cluster or alternate index is not to be deleted if the retention period has not expired. If you specify NOPURGE and the retention period has not expired, you get a message that tells you that the cluster or alternate index has not been deleted and the command is terminated. The portable copy of the object that was to have been imported is still usable.

Abbreviations: PRG and NPRG

Default: NOPURGE

IMPORTRA

The IMPORTRA command (MPRA) is used to reestablish in a catalog those objects that have become portable by means of the EXPORTRA command. Clusters and their associated data and index components and any paths over them, and alternate indexes and their associated data and index components and any paths over them, are automatically defined in the catalog by means of the IMPORTRA command. If user catalog pointers are found on the portable file, those user catalogs are connected to the master catalog by IMPORTRA. If the target (or receiving) catalog is not a recoverable catalog, any nonVSAM objects on the portable file are also defined in the catalog. (Note that nonVSAM objects can exist on the portable file only if they were defined into a recoverable catalog on an operating system other than DOS/VS or DOS/VSE.) Should IMPORTRA find an entry for the object in the catalog, it will attempt to delete it. If this succeeds, it will then redefine the object using the information on the portable file. If the object is a user catalog, it will be disconnected rather than deleted, and then reconnected. See "Using EXPORTRA/IMPORTRA: Recovering Catalog Entries and Data" for more information and Example 21 in "Appendix A: Sample Job Streams."

The format of the IMPORTRA command is:

```
IMPORTRA   INFILE( dname
              [ðENVIRONMENT
                 ( [ðBLOCKSIZE( size ) ]
                 [ðNOLABEL|STDLABEL]
                 [ðNOREWIND|REWIND|UNLOAD]
                 [ðPRIMEDATADEVICE( devtype ) ] ) ]
              )
           OUTFILE( dname )
           [CATALOG( catname [ / password ] ) ]
           [OBJECTS( ( entryname [ðFILE( dname ) ]
                 [ðUSECLASS( primary [ð secondary ] ) ]
                 [ðVOLUMES( volser . . . ) |
                   DEFAULTVOLUMES]
                 [ðDEVICETYPE( devtype ) ] ) . . . ) ]
```

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

IMPORTRA Parameters

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

specifies the name and the password of the catalog in which the objects being imported are to be defined. This parameter is required only when the catalog is password protected or when you want to direct the imported object to a catalog other than the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

catname – is the name of the catalog.

password – specifies the update or higher-level password of the catalog. If any of the files being imported are alternate indexes over a password-protected base cluster, specify the catalog's master password to avoid being prompted for the base cluster's master password.

Abbreviation: CAT

INFILE(*dname* [ƀENVIRONMENT(*subparameters*)])

dnamƀ – specifies the *filename* of the DLBL or TLBL statement that identifies the portable file. **This is a required parameter.** If an unlabeled tape is used (NOLABEL option), a TLBL statement is not required. However, since ***dname*** is a required positional parameter, a dummy (ficticious) ***filename*** of your choosing must be specified.

ENVIRONMENT(*subparameters*)

describes the parameters of the portable file.

Abbreviation: ENV

BLOCKSIZE(*size*)

specifies the block size of the portable file.

Abbreviation: BLKSZ

Default: 2048 bytes per block.

NOLABEL | STDLABEL

specifies the type of tape (unlabeled or EBCDIC standard-labeled) which contains the portable file, if PRIMEDATADEVICE specifies 2400.

NOLABEL

indicates an unlabeled tape is to be processed. Since the first record of an unlabeled tape may or may not be preceded by a tapemark (user's prerogative), you must take care to properly position the tape to the file you want to process. If a tapemark precedes the file, postion the tape either immediately past or immediately before the tapemark. If no tapemark is present, position the tape immediately before the first data record. If the unlabeled tape was created by VSE EXPORTRA, there is no tapemark preceding the first record of the file.

If VSE/Access Control is installed, REWIND must be specified for unlabeled tapes.

Abbreviation: NLBL

Restriction: If a portable-copy tape file is unlabeled, it must be contained on a single volume, because IMPORTRA cannot handle a multivolume unlabeled file.

STDLABEL

indicates EBCDIC standard-labeled tapes are to be processed.

Abbreviation: SLBL

Default: STDLABEL

Restriction: Access Method Services does not support ASCII files or nonstandard labels.

NOREWIND | REWIND | UNLOAD

specifies the tape positioning action for an OPEN, CLOSE, and EOV (end-of-volume) condition if PRIMEDATADEVICE specifies 2400. (Prior to DOS/VS Release 34, IMPORTRA and EXPORTRA were the only tape processing commands that rewound a tape on an OPEN, CLOSE, or EOV condition. Starting with DOS/VS Release 34, IMPORTRA and EXPORTRA are consistent with the other tape

processing commands, that is, NOREWIND is the default for the IMPORTRA and EXPORTRA commands.

Note: Be aware, when using the REWIND or UNLOAD options (especially with multi-file volumes), that any OPEN causes the tape to be positioned at load point. You must be sure that the file you want to process is really the first one (at load point) on the reel. Otherwise, specify the NOREWIND option and use the MTC job control statement (see *VSE/Advanced Functions*), for example, to properly position the tape.

NOREWIND
specifies that rewind is never to be performed on OPEN, CLOSE, and EOV.

End-of-file considerations: For EBCDIC standard-labeled tapes, the tape is positioned between the two trailing tapemarks. For unlabeled tapes, the tape is positioned following the single trailing tapemark.

Abbreviation: NREW

Default: NOREWIND

REWIND
specifies that tapes are to be rewound to load point but not unloaded on an OPEN, CLOSE, or EOV condition.

Abbreviation: REW

UNLOAD
specifies that tapes are to be rewound on an OPEN, and rewound and unloaded on an EOV and CLOSE condition.

Abbreviation: UNLD

PRIMEDATADEVICE(*devtype*)
specify this parameter only if the file resides on a tape device (in which case you specify 2400). The 2400 identifies all tape devices that DTFMT supports. SYS004 must be used (in the ASSGN statement) for a tape input file. See "Magnetic Tape Considerations for Access Method Services" in *VSE/VSAM Programmer's Reference* for more information on tape.

Abbreviation: PDEV

OBJECTS((*entryname*
[ƀFILE(*(dname)*)]
[ƀDEVICETYPE(*devtype*)]
[ƀUSECLASS(primary[ƀsecondary])]
[ƀVOLUMES(*volser* . . .) |
DEFAULTVOLUMES]) . . .)
specifies attributes for some object on the portable file.

entryname – specifies the name of the data component, index component, cluster, alternate index, user catalog, or nonVSAM object whose attributes are being specified. The name and attribute parameters can be repeated for up to 255 objects.

Abbreviation: OBJ

FILE(*dname*)
specifies the *filename* of the DLBL statement that, together with associated EXTENT statement(s), identifies the volume(s) and extents allocated to the data and index components of an alternate index or cluster

that is to be imported. This parameter is required only when one or both components are defined with the UNIQUE attribute. (If only one component is unique you code one FILE parameter; if both components are unique you code two FILE parameters.)

The *entryname* parameter preceding the FILE parameter cannot be a cluster or alternate index name; it must be a data or index component name.

The *file-ID* in the DLBL statement pointed to by *dname* can be omitted but, if included, it should be the same as the component name (that is, the *entryname* parameter preceding the FILE parameter). The symbolic unit can be omitted from the EXTENT statement.

Note: This parameter explanation applies to the job control simplification introduced in Release 2. If you are using Release 1 job control, refer to the appropriate explanation in Appendix J.

DEVICETYPE(*devtype*)

specifies the device type (2314, 2319, 3330, 3330-11, 3340, 3350, FBA, 2400T7, or 2400T9) of a user catalog or nonVSAM file to be imported. If you do not specify DEVICETYPE, the original exxported device type is used.

Abbreviation: DEVT

USECLASS(*primary* [ŏ *secondary*])

specifies that the class of an imported object (cluster, alternate index, and their data and index components) is to be modified. The USECLASS specifications for a cluster and alternate index are modified in the following order: cluster or alternate index; data component; and index component. For example, if you specify a USECLASS value for both the data component and cluster (in either order) of an imported object, (1) the value specified for the cluster level will also be propagated to the data component level and (if present) index component level, (2) the value specified for the data component will be inserted into the data component, superceding the previous data component value derived from the cluster specification, (3) the index component (if present) would retain the propagated cluster USECLASS value.

If you specify only a primary USECLASS value, the default secondary value ('P' - same as primary) is applied to the imported object. This effectively overrides the secondary value for that imported object in the portable file.

If you specify primary USECLASS values other than 0 for a unique cluster or alternate index, an error occurs.

Abbreviation: USCL

VOLUMES (*volser* . . .) | DEFAULTVOLUMES

specifies the volumes to be used in defining a cluster, alternate index, data component, index component, user catalog, or nonVSAM file.

VOLUMES

If you do not specify the VOLUMES (or DEFAULTVOLUMES) subparameter, the original volumes list (contained in the catalog entries for the object) that was exported will be used. If VOLUMES is specified at the cluster or alternate index level, it always propagates to the data and index levels. A data or index level specification overrides the original exported list of volumes and a cluster or alternate-index level

specification. If you want to change VOLUMES for only one component, specify VOLUMES only for that component.

volser – contains one through 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. For consistency with VSE job control, only alphameric characters should be used.

DEFAULTVOLUMES
indicates that (for the specified cluster, alternate index, data or index component) VSAM is to disregard the exported volume list(s) and, instead, is to obtain a new list of volume(s). VSAM obtains the new volume(s) list from the data and index (if present) components of the applicable default models in the receiving catalog. The default model used corresponds to the type of file being imported (alternate index, key-sequenced file, etc.).

An error message is issued if you specify DEFAULTVOLUMES and VOLUMES together in the same *objectname* list. You can specify DEFAULTVOLUMES at any level and in any combination of levels; if specified at the alternate index or cluster level, DEFAULTVOLUMES propagates to the data component level and (if applicable) the index component level. (If DEFAULTVOLUMES is propagated to the data or index component level where the VOLUMES parameter has been specified, the VOLUMES parameter takes precedence.)

Abbreviation: VOL and DFVOL

LISTCAT

The LISTCAT command (LISTC) is used to list entries contained in a catalog. The ENTRIES parameter is used to specify the name(s) of individual entries to be listed. The AIX, CL, DATA, INDEX, NONVSAM, PATH, SPACE, and UCAT parameters are used to specify the type(s) of entries to be listed. The NAME, VOLUME, ALLOCATION, and ALL parameters specify the fields to be listed for each catalog entry.

Figure 3-2 shows the kinds of listings that result from coding various combinations of *entryname* and type of object.

See "Using LISTCAT: Listing Catalog Entries" for more information, "Appendix B: Interpreting LISTCAT Output" for an explanation of the output produced by the LISTCAT command, and Examples 3, 4, and 7 in "Appendix A: Sample Job Streams."

Requested type is:	**CL**	**AIX**	**DATA**	**IX**	**PATH**	**SPC**	**NVSAM**	**UCAT**	**None**
Entryname type is:									
CL	L*	E	A*	A*	A*	E	E	E	LA*
AIX	E	L*	A*	A*	A*	E	E	E	LA*
DATA	E	E	L*	E	E	E	E	E	L*
IX	E	E	E	L*	E	E	E	E	L*
PATH	E	E	E	E	L*	E	E	E	L*
SPC	U	U	U	U	U	L*	U	U	U
NVSAM	E	E	E	E	E	E	L*	E	L*
UCAT	MU	E	E	E	E	E	E	L*	UM

where

- A = list the desired association
- E = error message: "not a requested type"
- L = list the entry
- LA = list the entry and associated entries
- MU = if listing from master catalog, "E"
  if listing from user catalog, "L"
- U = unpredictable results
- UM = if listing from master catalog, "L"
  if listing from user catalog, "LA"

and

- AIX = alternate index
- CL = cluster
- IX = index
- NVSAM = nonVSAM
- SPC = space
- UCAT = user catalog
- * = combination producing a useful listing

Note: If any of the above *entryname* types are not in the catalog, Access Method Services generates an appropriate information message.

Figure 3-2. Listing Contents Depending Upon Parameters Requested

The format of the LISTCAT command is:

```
LISTCAT   [ALTERNATEINDEX]
             [ƀCLUSTER]
             [ƀDATA]
             [ƀINDEX]
             [ƀNONVSAM]
             [ƀPATH]
             [ƀSPACE]
             [ƀUSERCATALOG]
          [CATALOG( catname [/ password ] [ƀ dname ] )]
          [ENTRIES( entryname [/ password ]
             [ƀ entryname [/ password ] [ƀ...]|
             volser [ƀ volser ...])]
          [NAME|VOLUME|ALLOCATION|ALL]
          [NOTUSABLE]
```

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

LISTCAT Parameters

ALTERNATEINDEX CLUSTER DATA INDEX
NONVSAM PATH SPACE USERCATALOG

specifies the types of catalog entries to be listed. The catalog specified in CATALOG (or the default) contains the entries. Only those entries whose type is specified are listed (that is, when you specify CLUSTER but not INDEX, DATA, or PATH, the cluster's entry is listed and its associated data, index, and path 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 you want. If you want to completely list a catalog, do not specify any entry type. When either DATA or INDEX is specified, the entry for the data or index component of the catalog itself is not listed.

ALTERNATEINDEX

specifies that alternate index entries are to be listed. If ALTERNATEINDEX is specified but DATA, INDEX, and PATH are not specified, entries for the alternate index's data and index components and path(s) are not listed.

Abbreviation: AIX

CLUSTER

specifies that cluster entries are to be listed. If CLUSTER is specified but DATA, INDEX, and PATH are not specified, entries for the cluster's data and index components and path are not listed.

Abbreviation: CL

DATA

specifies that entries for data components, excluding the data component of the catalog, are to be listed. If a cluster or alternate index's name is specified under ENTRIES and DATA is coded, only the data component entry is listed.

INDEX
specifies that entries for index components, excluding the index component of the catalog, are to be listed. If a cluster or alternate index's name is specified under ENTRIES and INDEX is coded, only the index component entry is listed.

Abbreviation: IX

NONVSAM
specifies that entries for nonVSAM files are to be listed.

Abbreviation: NVSAM

PATH
specifies that entries for paths are to be listed.

SPACE
specifies that entries for volumes containing data spaces defined in the catalog are to be listed. Candidate volumes are included. If entries are identified by *entryname*, SPACE can be coded only when no other entry type is coded. You must specify SPACE if ENTRIES specifies a volser (volume serial number).

Abbreviation: SPC

USERCATALOG
specifies that user catalog pointers in the master catalog are to be listed. USERCATALOG is applicable only when the catalog to be listed is the master catalog.

Abbreviation: UCAT

CATALOG (*catname* [/ *password*])
specifies the catalog that contains the entries that are to be listed. This parameter is required only when the catalog is password protected or the entry to be listed is in a catalog other than the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

catname – is the name of the catalog.

password – specifies the read or higher-level password of the catalog if it is password protected. If the entries to be listed contain password protection information and this information is to be listed, then the master password must be supplied either through this parameter or through the ENTRIES parameter.

Abbreviation: CAT

ENTRIES (*entryname* [/ *password*] [ƀ *entryname* [/ *password*] [ƀ . . .]] | *volser* ([ƀ *volser* . . .])
specifies the entries to be listed. If ENTRIES is not specified and no entry-types are specified, the entire catalog is listed. You can also specify the fields you want listed for each entry (ALL, NAME, and so forth). If you want information about a user catalog, the catalog's volume must be physically mounted. You then specify the catalog's name as the *entryname*. (You cannot specify the catalog's components by name.)

If you want information about the data spaces on particular volumes, specify SPACE and the volumes' serial numbers as *volser*.

password – specifies a password when the entry to be listed is password protected and a password was not specified through the CATALOG

parameter. The password must be the entry's read or higher-level password. If protection attributes are to be listed, you must supply the entry's master password; if no password is supplied, the operator is prompted for each entry's password.

Abbreviation: ENT

NAME | VOLUME | ALLOCATION | ALL
specifies the fields to be listed for each catalog entry.

NAME
specifies that the name and type of the objects defined by each entry are to be listed. NAME is the default.

VOLUME
specifies that the name of the object defined by each entry, its history information (ownerid, creation and expiration dates, the VSAM release ID when it was created), its volume serial number, and device type allocated to it are to be listed; additionally, if present, the catalog recovery volume's serial number, control interval number, and device type. Volume information (volser and devtype) is only listed for data and index component entries, user catalogs, and nonVSAM file entries.

Abbreviation: VOL

ALLOCATION
specifies that volume information and detailed information about the allocation are to be listed. This information is listed only for data and index component entries. Types of entries should only specify DATA or INDEX. ENTRIES can specify cluster, alternate index, data or index component names.

Abbreviation: ALLOC

ALL
specifies that all the fields in an entry are to be listed, including protected attributes, if the entry's or the catalog's master password is specified.

NOTUSABLE
specifies that only those entries for data and index components that are flagged as unusable are to be listed. They are so flagged because data space has been deleted from the volume by either the FORCE option of DELETE SPACE or the IGNORE option of RESETCAT, or because ALTER REMOVEVOLUMES has removed the recovery volume or the last volume of a NOALLOCATION file.

Abbreviation: NUS

LISTCRA

The LISTCRA command (LISTR) is used to list the contents of one or more volume's catalog recovery area (CRA). LISTCRA either dumps all CRA records or lists each defined object's type, name, and volume(s). The entries in the listing can be limited to those entries in the CRA that do not match the catalog. In this case the listing includes a *MISCOMPARES message which identifies whether name, volume, key ranges, extents, high-used RBA, or statistics do not match. The term "OTHER" in a *MISCOMPARES message refers to mismatches other than those specified that are less severe and do not affect data. The corresponding data from the catalog is also included and the mismatched fields are underlined if the DUMP option has been specified.

See *VSE/VSAM Programmer's Reference* for the contents of CRAs and how catalog entries are placed in CRAs.

The output listing can be used to select objects for the EXPORTRA function or, in the case of a mismatch in the high-used RBA, to select an object for the VERIFY function. See "Using EXPORTRA/IMPORTRA: Recovering Catalog Entries and Data" for more information, Example 19 in "Appendix A: Sample Job Streams," and "Appendix D: Interpreting LISTCRA Output" for an explanation of the output produced by the LISTCRA command.

The format of the LISTCRA command is:

LISTCRA	INFILE(*dname* [b . . .]) INVOLUMES (*volser* . . .) [CATALOG (*catname* [/ *password*] b *dname*)] [COMPARE | NOCOMPARE] [DUMP | NAME | SEQUENTIALDUMP] [MASTERPW (*password*)]

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

LISTCRA Parameters

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

specifies the name of the catalog that owns the volumes containing the CRAs. If COMPARE is specified, the CATALOG parameter must be specified.

catname – is the name of the catalog whose entries are to be compared with the entries in the CRAs.

password – specifies the master password of the catalog identified by *catname* (must be specified if the catalog is password protected.)

Abbreviation: CAT

COMPARE | NOCOMPARE

specifies whether the CRA entries are to be compared to their corresponding catalog entries. If COMPARE is specified, the printed listing is limited to mismatches. All entries except the entries that describe the catalog itself are compared. If NOCOMPARE is specified, all entries are printed.

When COMPARE is specified, the CATALOG parameter is required. COMPARE cannot be specified with SEQUENTIALDUMP.

Abbreviations: CMPR and NCMPR

Default: NOCOMPARE

DUMP | NAME | SEQUENTIALDUMP
specifies the type of list. DUMP indicates a dump of all of each entry's records in hexadecimal and character format in alphabetical order by entry name and grouped by type of object. NAME indicates a list of each defined object's type, name, and volume(s). Only the names of clusters, alternate indexes, and nonVSAM objects are listed in alphabetical order. Associated information is also given where relevant.

SEQUENTIALDUMP indicates a dump of all records in hexadecimal and character format in the physical sequential order they occur in the CRA with no grouping or logical ordering (first-in, first-out). SEQUENTIAL-DUMP cannot be specified with COMPARE.

Abbreviation: SDUMP

Default: NAME

INVOLUMES (*volser. . .*)
indicates the volumes of the CRAs to be listed. You can list more than one CRA by supplying multiple *volser* subparameters. The CRA volumes need not be premounted. They can be serially mounted as each volume is processed.

Abbreviation: IVOL

MASTERPW (*password*)
specifies the master password of the master catalog. It must be specified if the master catalog is password protected.

Abbreviation: MRPW

PRINT

The PRINT command is used to list part or all of an indexed-sequential, sequential, or VSAM file. A base cluster is listed in alternate key sequence via a path. See "Sample Output from PRINT," following this command, for sample listings produced by the PRINT command. See "Using PRINT: Printing Data Records" for more information and Examples 5, 6, and 8 in "Appendix A: Sample Job Streams."

The format of the PRINT command is:

```
PRINT   INFILE
           ( dname [ / password ]
           [ðENVIRONMENT                                          †
             (ðBLOCKSIZE( size )                                  †
             [ðHINDEXDEVICE( devtype ) ]
             [ðNOLABEL | STDLABEL]
             [ðNOREWIND | REWIND | UNLOAD]
             [ðPRIMEDATADEVICE( devtype ) ]
               ðRECORDFORMAT( format )                            †
             [ðRECORDSIZE( size ) ] ) ]
           )
        [CHARACTER | DUMP | HEX]
        [FROMADDRESS( address ) | FROMKEY( key ) |
           FROMNUMBER( number ) | SKIP( count ) ]
        [TOADDRESS( address ) | TOKEY( key ) |
           TONUMBER( number ) | COUNT( count ) ]
```

† This parameter is required under certain conditions. See the following parameter explanations or Appendix E for more information.

PRINT Parameters

INFILE(*dname* [/ *password*] [ðENVIRONMENT (*subparameters*)])

describes the input file. If the file is defined in a user catalog which is not the job catalog, you must explicitly name the user catalog by specifying the CAT parameter in the file's DLBL job control statement. (The user catalog's DLBL statement does not require an associated EXTENT statement.)

dname – specifies the *filename* of the DLBL (an associated EXTENT statment is required only if the input file is nonVSAM) or TLBL statement that identifies the input file to be listed. **This is a required parameter.** You can list a base cluster in alternate key sequence by specifying a path name as the *file-ID*, in the DLBL statement. If an unlabeled tape is used (NOLABEL option), a TLBL statement is not required. However, since ***dname*** is a required positional parameter, a dummy (ficticious) ***filename*** of your choosing must be specified.

password – specifies the read or higher-level password of a password protected VSAM file or path. For printing a data or index component,

supply the password of the cluster or alternate index to which the component belongs; for a path, it is the password of the path.

Abbreviation: IFILE

Note: This parameter explanation applies to the job control simplification introduced in Release 2. If you are using Release 1 job control, refer to the appropriate explanation in Appendix J.

ENVIRONMENT(*subparameters*)
describes the input file if that file is a nonVSAM file. When you are printing a nonVSAM file, this parameter and the BLOCKSIZE and RECORDFORMAT subparameters are required. The subparameters are described in the following paragraphs.

Abbreviation: ENV

BLOCKSIZE(*size*)
specifies the block size for a nonVSAM file. This parameter is required. For FIXBLK files, substitute the logical record length times the number of records per block. For indexed-sequential FIXUNB files, substitute the logical record length plus the key length.

Abbreviation: BLKSZ

HINDEXDEVICE(*devtype*)
specifies, for an indexed-sequential file, the device type of the volume on which the highest index resides. The possible device types are 2314, 2319, 3330, and 3340.

Abbreviation: HDEV

Default: The device type of the highest index defaults to the device type of the volume on which the prime data records reside (see PRIMEDATADEVICE).

NOLABEL | STDLABEL
specifies the type of tape (unlabeled or EBCDIC standard-labeled) to be processed if PRIMEDATADEVICE specifies 2400.

NOLABEL
indicates an unlabeled tape is to be processed. Since the first record of an unlabeled tape may or may not be preceded by a tapemark (user's prerogative), you must take care to properly position the tape to the file you want to process. If a tapemark precedes the file, position the tape either immediately past or immediately before the tapemark. If no tapemark is present, position the tape immediately before the first data record. If the unlabeled tape was created by VSE Access Method Services, there is no tapemark preceding the first record of the file.

If VSE/Access Control is installed, REWIND must be specified for unlabeled tapes.

Abbreviation: NLBL

Restriction: PRINT cannot process multivolume unlabeled files.

STDLABEL
indicates EBCDIC standard-labeled tapes are to be processed.

Abbreviation: SLBL

Default: STDLABEL

Restriction: Access Method Services does not support ASCII files or nonstandard labels.

NOREWIND | REWIND | UNLOAD

specifies the tape positioning action for an OPEN, CLOSE, and EOV (end-of-volume) condition if PRIMEDATADEVICE specifies 2400.

Note: Be aware, when using the REWIND or UNLOAD options (especially with multi-file volumes), that any OPEN causes the tape to be positioned at load point. You must be sure that the file you want to process is really the first one (at load point) on the reel. Otherwise, specify the NOREWIND option and use the MTC job control statement (see *VSE/Advanced Functions*), for example, to properly position the tape.

NOREWIND

specifies that rewind is never to be performed on OPEN, CLOSE, and EOV.

End-of-file considerations: For EBCDIC standard-labeled tapes, the tape is positioned between the two trailing tapemarks. For unlabeled tapes, the tape is positioned following the single trailing tapemark.

Abbreviation: NREW

Default: NOREWIND

REWIND

specifies that tapes are to be rewound to load point but not unloaded on an OPEN, CLOSE, or EOV condition.

Abbreviation: REW

UNLOAD

specifies that tapes are to be rewound on an OPEN, and rewound and unloaded on an EOV and CLOSE condition.

Abbreviation: UNLD

PRIMEDATADEVICE(*devtype*)

specifies the device type of the volume on which the prime data records reside. The possible device types that apply to an ISAM file are 2314, 2319, 3330, and 3340. You do not have to specify this parameter for a sequential file unless the file resides on tape (in which case you specify 2400). The 2400 identifies all tape devices that DTFMT supports. You must specify SYS004 in the ASSGN statement for a tape input file.

Abbreviation: PDEV

Default: 2314

RECORDFORMAT(*format*)

specifies the format of the records in the nonVSAM file. This parameter is required. For *format,* substitute one of the following values:

Value	Meaning
FIXUNB|F	Fixed, unblocked
FIXBLK|FB	Fixed, blocked
VARUNB|V	Variable, unblocked
VARBLK|VB	Variable, blocked
SPNUNB|S	Variable spanned, unblocked
SPNBLK|SB	Variable spanned, blocked
UNDEF|U	Undefined

Abbreviation: RECFM

RECORDSIZE (*size*)
specifies the length of a logical record for FIXBLK or FIXUNB nonVSAM files. For indexed-sequential FIXUNB files, this parameter is the logical record length plus the key length. For UNDEF, SPNBLK, or SPNUNB nonVSAM files, this parameter is the maximum record size. This parameter is not necessary for VARUNB or VARBLK.

Abbreviation: RECSZ

Default: If RECORDSIZE is not specified, blocking is one record per block, therefore, RECORDSIZE is equal to block size for files that are FIXUNB, FIXBLK, or UNDEF; RECORDSIZE is equal to block size minus four for files that are VARUNB, VARBLK, SPNUNB, or SPNBLK.

CHARACTER | DUMP | HEX
specifies the format of the listing.

CHARACTER
specifies that each byte in each 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 3-3).

Abbreviation: CHAR

DUMP
specifies that each byte in each 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 3-3).

HEX
specifies that each byte in each record is to be printed as two hexadecimal digits. Key fields are listed in hexadecimal format (see Figure 3-3).

Default: DUMP

FROMADDRESS (*address*) | FROMKEY (*key*) |
FROMNUMBER (*number*) | SKIP (*count*)
specifies the first record to print in the file.

FROMADDRESS (*address*)
specifies the RBA of the first record. It must be the beginning of a logical record. If you specify this parameter for a key-sequenced file, the listing will be sequenced by RBA rather than by key. The address may be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

FROMKEY(*key*)
specifies the key of the first record you want listed. It may be a full key or a generic key — that is, the high-order portion of a key. If you specify a generic key, listing begins at the first record whose key matches the portion of the key you specified. (You must not specify a key longer than that defined for the file. If you do, the listing is not performed.) If the specified key is not found, listing begins with the next higher key. 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 if the key is enclosed in single quotation marks; it may contain up to 255 bytes, that is, 510 hexadecimal or 255 EBCDIC characters. If it is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks.

FROMNUMBER(*number*)
specifies the relative-record number of the first record.

SKIP(*count*)
specifies the number of records you want to skip before the listing of records begins. For example, if you want the listing to begin with the 500th record, you specify SKIP(499). The count can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B, respectively, enclosed in single quotation marks, and no longer than one fullword (4 bytes or 32 bits).

Abbreviations: FADDR, FKEY, FNUM

Default: If nothing is specified, the listing begins with the first record.

Restrictions:

Parameter	Can be used with	Cannot be used in combination with	Path processing
FROMADDRESS	Key-sequenced and entry-sequenced files	TOKEY TONUMBER	Not permitted
FROMKEY	Key-sequenced and indexed-sequential files	TOADDRESS TONUMBER	Permitted
FROMNUMBER	Relative-record files	TOADDRESS TOKEY	Not permitted
SKIP	Indexed-sequential, sequential and VSAM files		Unpredictable results

TOADDRESS(*address*) | TOKEY(*key*) | TONUMBER(*number*) | COUNT(*count*)
specifies the last record to print in the file. The last record must follow the first record.

TOADDRESS(*address*)
specifies the RBA of the last record. Unlike FROMADDRESS, the RBA 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 file, the listing will be sequenced by RBA rather than by key. The address can be expressed in decimal, hexadecimal, or binary. If it is specified in hexadecimal or binary, it must be preceded by X or B, respectively, and enclosed in single quotation marks.

TOKEY(*key*)
specifies the key of the last record. It may be a full key or a generic key — that is, the high-order portion of a key. If you specify a generic key, listing stops after the last record is listed whose key matches the portion of the key you specified. (You must not specify a key longer than that defined for the file.If you do, the listing is not performed.) If the specified key is not found, listing ends with the next lower key. 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 if the key is enclosed in single quotation marks; it may contain up to 255 bytes, that is, 510 hexadecimal or 255 EBCDIC characters. If it is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks.

TONUMBER(*number*)
specifies the relative-record number of the last record.

COUNT(*count*)
specifies the number of records to be listed. Processing is terminated when the number of records printed exceeds the value supplied by COUNT. The count can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n' form.

Abbreviations: TADDR, TNUM

Default: If nothing is specified, the listing ends with the last record.

Restrictions:

Parameter	Can be used with	Cannot be used in combination with	Path processing
COUNT	Indexed sequential, sequential and VSAM files		Unpredictable results
TOADDRESS	Key-sequenced and entry-sequenced files	FROMKEY FROMNUMBER	Not permitted
TOKEY	Key-sequenced and indexed-sequential files	FROMADDRESS FROMNUMBER	Permitted
TONUMBER	Relative-record files	FROMADDRESS FROMKEY	Not permitted

Dump Format

```
IDCAMS  SYSTEM SERVICES            TIME:  19:25:14   03/08/73  PAGE  2

LISTING OF DATA SET -EXAMPLE.LISTC

KEY OF RECORD - F0F0F0F0F0F0F0F0F0F0F0F1
0000   C1C2C3C4 F0F0F0F0 F0F0F0F0 F0F0F0F1   C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7   *ABCD000000000001ABCDEFGHIJKLMNOP*
0020   D8D9E2E3 E4E5E6E7 E8E9F0F1 F2F3F4F5   F6F7F8F9 C1C2C3C4 C5C6C7C8 C9D1D2D3   *QRSTUVWXYZ0123456789ABCDEFGHIJKL*
0040   D4D5D6D7 D8D9E2E3 E4E5E6E7 F8F9F0F1   F2F3F4F5 F6F7F8F9 C1C2C3C4 C5C6C7C8   *MNOPQRSTUVWXYZ0123456789ABCDEFGH*
0060   C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7   E8E9F0F1 F2F3F4F5 F6F7F8F9 C1C2C3C4   *IJKLMNOPQRSTUVWXYZ0123456789ABCD*
0080   C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3   E4E5E6E7 E8E9F0F1 F2F3F4F5 F6F7F8F9   *EFGHIJKLMNOPQRSTUVWXYZ0123456789*
00A0   C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7   D8D9E2E3 E4E5E6E7 E8E9F0F1 F2F3F4F5   *ABCDEFGHIJKLMNOPQRSTUVWXYZ012345*
00C0   F6F7F8F9 C1C2C3C4 C5C6C7C8 C9D1D2D3   D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E9F0F1   *6789ABCDEFGHIJKLMNOPQRSTUVWXYZ01*
00E0   F2F3F4F5 F6F7F8F9 C1C2C3C4 C5C6C7C8   C9D1D2D3 D4D5D6D7 D8D9E2E3 E4E5E6E7   *23456789ABCDEFGHIJKLMNOPQRSTUVWX*
0100   E8E9F0F1 F2F3F4F5 F6F7F8F9 C1C2C3C4   C5C6C7C8 C9D1D2D3 D4D5D6D7 D8D9E2E3   *YZ0123456789ABCDEFGHIJKLMNOPQRST*
0120   E4E5E6E7 E8E9F0F1 F2F3F4F5 F6F7F8F9   C1C2C3C4 C5C6C7C8 C9D1D2D3 D4D5D6D7   *UVWXYZ0123456789ABCDEFGHIJKLMNOP*
0140   D8D9E2E3 E4E5E6E7 E8E9F0F1 F2F3F4F5   F6F7F8F9 C1C2C3C4 C5C6C7C8 C9D1D2D3   *QRSTUVWXYZ0123456789ABCDEFGHIJKL*
0160   D4D5D6D7 D8D9E2E3 E4E5E6E7 E8E9F0F1   F2F3F4F5 F6F7F8F9 C1C2C3C4 C5C6C7C8   *MNOPQRSTUVWXYZ0123456789ABCDEFGH*
0180   C9D1D2D3 D4D5D6D7 D8D9E2E3 00000000                     *IJKLMNOPQRST....       *
```

Hex Format

```
IDCAMS   SYSTEM SERVICES      TIME:  19:26:13   03/08/73  PAGE  4

LISTING OF DATA SET -EXAMPLE.LISTC.DATA

RBA OF RECORD - 0
C1C2C3C4F0F0F0F0F0F0F0F0F0F0F1C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8
C9D1D2D3D4D5D6D7D8D9E2D3D4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5
F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3
E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8
C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F0F1F2F3F4F5
F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3E4E5E6E7E8E9F-F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E3
E4E5E6E7E8E9F0F1F2F3F4F5f6F7F8F9C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6D7D8D9E2E300000000
```

Character Format

```
IDCAMS   SYSTEM SERVICES            TIME:  19:26:13   03/08/73  PAGE  6

LISTING OF DATA SET -EXAMPLE.LISTC.DATA

RBA OF RECORD - 0
ABCD000000000001ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ012345
6789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGH
IJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789ABCDEFGHIJKLMNOPQRST
UVWXYZ0123456789ABCDEFGHIJKLMNOPQRST....
```

Figure 3-3. Sample Output From PRINT

REPRO

The REPRO command is used to do any of the following:

- Copy a VSAM file into another VSAM file.
- Copy a sequential file on tape or disk into another sequential file on tape or disk.
- Convert a sequential or indexed-sequential file into a VSAM file.
- Convert a VSAM or ISAM file into a sequential file.
- Reorganize the records and space in a key-sequenced file into another key-sequenced file. See "Reorganizing a File" to find out what happens when records from the input file are added to an empty or nonempty output file.
- Merge two VSAM files.
- Copy a catalog to a sequential nonVSAM file or to a VSAM key-sequenced file or entry-sequenced file and vice versa. (Do not copy the master catalog to a VSAM file.) See Examples 17 and 18 in "Appendix A: Sample Job Streams" for more information.

See also "Loading Records into a File" in "Defining a VSAM file" for more information and Examples 5 and 6 in "Appendix A: Sample Job Streams."

The format of the REPRO command is:

REPRO	INFILE (*dname* [/ *password*] [ƀENVIRONMENT (BLOCKSIZE(*size*) [ƀHINDEXDEVICE(*devtype*)] [ƀNOLABEL|STDLABEL] [ƀNOREWIND|REWIND|UNLOAD] [ƀPRIMEDATADEVICE(*devtype*)] ƀRECORDFORMAT(*format*) [ƀRECORDSIZE(*size*)])]) OUTFILE (*dname* [/ *password*] [ƀENVIRONMENT (BLOCKSIZE(*size*) [ƀNOLABEL|STDLABEL] [ƀNOREWIND|REWIND|UNLOAD] [ƀPRIMEDATADEVICE(*devtype*)] ƀRECORDFORMAT(*format*) [ƀRECORDSIZE(*size*)])]) [FROMADDRESS(*address*)|FROMKEY(*key*)| FROMNUMBER(*number*)|SKIP(*count*)] [TOADDRESS(*address*)|TOKEY(*key*)| TONUMBER(*number*)|COUNT(*count*)] [REPLACE|NOREPLACE] [REUSE|NOREUSE]

INFILE(*dname*[/ *password*]
[ŏENVIRONMENT(*subparameters*)])

describes the input file. If the file is defined in a user catalog which is not the job catalog, you must explicitly name the user catalog by specifying the CAT parameter in the file's DLBL job control statement. (The user catalog's DLBL statement does not require an associated EXTENT statement.)

dname – specifies the *filename* of the DLBL (an associated EXTENT statement is required only if the input file is a nonVSAM file) or TLBL job control statement that identifies the file to be copied. You can process a base cluster in alternate key sequence by specifying a path name as *file-ID* in the DLBL statement. **This is a required parameter.**

If an unlabeled tape is used (NOLABEL option), a TLBL statement is not required. However, since ***dname*** is a required positional parameter, a dummy (ficticious) ***filename*** of your choosing must be specified.

password – the read or higher-level password of the password-protected file to be copied. If a catalog is to be copied, the catalog's master password is required. For a path, it is the read or higher-level password of the path.

Abbreviation: IFILE

Note: This parameter explanation applies to the job control simplification introduced in Release 2. If you are using Release 1 job control, refer to the appropriate explanation in Appendix J.

ENVIRONMENT(*subparameters*)

describes the input file if that file is a nonVSAM file. This parameter, and the BLOCKSIZE and RECORDFORMAT parameters are required when you are copying a nonVSAM file, or reloading a backup copy of a VSAM catalog from a nonVSAM file. The ENVIRONMENT subparameters are described in the following paragraphs.

Abbreviation: ENV

BLOCKSIZE(*size*)

specifies the block size for a nonVSAM file. This parameter is required. For FIXBLK files, substitute the logical record length times the number of records per block. For indexed-sequential FIXUNB files, substitute the logical record length plus the key length. If you are reloading a backup copy of a VSAM catalog, block size must be a multiple of 516 plus 4.

Abbreviation: BLKSZ

HINDEXDEVICE(*devtype*)

specifies, for an indexed-sequential file, the device type of the volume on which the highest index resides. The possible device types are 2314, 2319, 3330, and 3340.

Abbreviation: HDEV

Default: The device type of the highest index defaults to the device type of the volume on which the prime data records reside (see PRIMEDATADEVICE).

NOLABEL | STDLABEL

specifies the type of tape (unlabeled or EBCDIC standard-labeled) to be processed if PRIMEDATADEVICE specifies 2400.

NOLABEL
indicates an unlabeled tape is to be processed. Since the first record of an unlabeled tape may or may not be preceded by a tapemark (user's prerogative), you must take care to properly position the tape to the file you want to process. If a tapemark precedes the file, position the tape either immediately past or immediately before the tapemark. If no tapemark is present, position the tape immediately before the first data record. If the unlabeled tape was created by VSE Access Method Services, there is no tapemark preceding the first record of the file.

If VSE/Access Control is installed, REWIND must be specified for unlabeled tapes.

Abbreviation: NLBL

Restriction: REPRO cannot process multivolume unlabeled files.

STDLABEL
indicates EBCDIC standard-labeled tapes are to be processed.

Abbreviation: SLBL

Default: STDLABEL

Restriction: Access Method Services does not support ASCII files or nonstandard labels.

NOREWIND | REWIND | UNLOAD
specifies the tape positioning action for an OPEN, CLOSE, and EOV (end-of-volume) condition if PRIMEDATADEVICE specifies 2400.

Note: Be aware, when using the REWIND or UNLOAD options (especially with multi-file volumes), that any OPEN causes the tape to be positioned at load point. You must be sure that the file you want to process is really the first one (at load point) on the reel. Otherwise, specify the NOREWIND option and use the MTC job control statement (see *VSE/Advanced Functions*), for example, to properly position the tape.

NOREWIND
specifies that rewind is never to be performed on OPEN, CLOSE, and EOV.

End-of-file considerations: For EBCDIC standard-labeled tapes, the tape is positioned between the two trailing tapemarks. For unlabeled tapes, the tape is positioned following the single trailing tapemark.

Abbreviation: NREW

Default: NOREWIND

REWIND
specifies that tapes are to be rewound to load point but not unloaded on an OPEN, CLOSE, or EOV condition.

Abbreviation: REW

UNLOAD
specifies that tapes are to be rewound on an OPEN, and rewound and unloaded on an EOV and CLOSE condition.

Abbreviation: UNLD

PRIMEDATADEVICE(*devtype*)
specifies the device type of the volume on which the prime data records reside. The possible device types that apply to an ISAM file are 2314, 2319, 3330, and 3340. You do not have to specify this parameter for a sequential file, unless the file resides on tape (in which case you specify 2400). The 2400 identifies all tape devices that DTFMT supports. You must specify SYS004 in the ASSGN statement for a tape input file.

Abbreviation: PDEV

Default: 2314

RECORDFORMAT(*format*)
specifies the format of the records in a nonVSAM file. This parameter is required. For *format,* substitute one of the following values:

Value	Meaning
FIXUNB|F	Fixed, unblocked
FIXBLK|FB	Fixed, blocked
VARUNB|V	Variable, unblocked
VARBLK|VB	Variable, blocked
SPNUNB|S	Variable spanned, unblocked
SPNBLK|SB	Variable spanned, blocked
UNDEF|U	Undefined

VARBLK must be specified if you are reloading a backup copy of a VSAM catalog from a nonVSAM file.

Abbreviation: RECFM

RECORDSIZE(*size*)
specifies the length of a logical record for FIXBLK or FIXUNB nonVSAM files. For indexed-sequential FIXUNB files, this parameter is the logical record length plus the key length. For UNDEF, SPNBLK, or SPNUNB nonVSAM files, this parameter is the maximum record size. This parameter is not necessary for VARUNB. If you are reloading a backup copy of a VSAM catalog, record size must be 516.

Abbreviation: RECSZ

Default: If RECORDSIZE is not specified, blocking is one record per block, therefore, RECORDSIZE is equal to block size for files that are FIXUNB, FIXBLK, or UNDEF; RECORDSIZE is equal to block size minus four for files that are VARUNB, VARBLK, SPNUNB, or SPNBLK.

OUTFILE(*dname*[/*password*] [ENVIRONMENT(*subparameters*)])
dname – specifies the *filename* of the DLBL (an associated EXTENT statement is required only if the output file is a sequential file) or TLBL job control statement that identifies the output file. ISAM files cannot be specified as output files. **This is a required parameter**. For VSAM files, the *file-ID* may be that of a path. A master catalog must be copied to a sequential file.

If an unlabeled tape is used (NOLABEL option), a TLBL statement is not required. However, since ***dname*** is a required positional parameter, a dummy (ficticious) ***filename*** of your choosing must be specified.

If the file is defined in a user catalog which is not the job catalog, you must explicitly name the user catalog by specifying the CAT parameter in the

file's DLBL job control statement. (The user catalog's DLBL statement does not require an associated EXTENT statement.)

password – the update or higher-level password of the VSAM file being used for output if it is password protected. If a catalog is to be reloaded, the catalog's master password is required; for a path, it is the update or higher-level password of the path.

Abbreviation: OFILE

Note: This parameter explanation applies to the job control simplification introduced in Release 2. If you are using Release 1 job control, refer to the appropriate explanation in Appendix J.

ENVIRONMENT(*subparameters*)

describes the output file if that file is a nonVSAM file. This parameter, and the BLOCKSIZE and RECORDFORMAT parameters are required when a nonVSAM file is produced as output or a VSAM catalog is copied (unloaded) into a sequential (SAM) file. Its subparameters are described in the following paragraphs.

Abbreviation: ENV

BLOCKSIZE(*size*)

specifies the block size for a nonVSAM file. This parameter is required. For fixed-length records, substitute the logical record length times the number of records per block. Block size must be a multiple of 516 plus 4 if a catalog is to be unloaded into a sequential file.

For variable-length records, the block size must be the length of the largest block possible (each block includes a 4-byte block length field, and each logical record in the block includes a 4-byte record length field). If the input file (INFILE) is a VSAM file or contains fixed or undefined records, Access Method Services prefixes each logical record with a 4-byte record length field which you must account for when you specify BLOCKSIZE.

Abbreviation: BLKSZ

NOLABEL | STDLABEL

specifies the type of tape (unlabeled or EBCDIC standard-labeled) to be processed if PRIMEDATADEVICE specifies 2400.

NOLABEL

indicates an unlabeled tape is to be processed. You must properly position the tape to the file you want to process. For an output file, Access Method Services does not write a tapemark as the first record. Instead, a data record is written immediately, wherever the tape is positioned. If an output file is to begin somewhere in the middle of the reel, it is your responsibility to position the tape immediately past the ending tapemark of the preceeding file. Use the MTC job control statement or command (refer to *VSE/Advanced Functions* to properly position the tape or to write a tapemark.

If VSE/Access Control is installed, REWIND must be specified for unlabeled tapes.

Abbreviation: NLBL

Restriction: Although REPRO can create multivolume unlabeled output files, other Access Method Services commands (including REPRO INFILE) can only process single volume unlabeled files.

STDLABEL
indicates EBCDIC standard-labeled tapes are to be processed.

Abbreviation: SLBL

Default: STDLABEL

Restriction: Access Method Services does not support ASCII files or nonstandard labels.

NOREWIND | REWIND | UNLOAD
specifies the tape positioning action for an OPEN, CLOSE, and EOV (end-of-volume) condition if PRIMEDATADEVICE specifies 2400.

Note: Be aware, when using the REWIND or UNLOAD options (especially with multi-file volumes), that any OPEN causes the tape to be positioned at load point. You must be sure that the file you want to process is really the first one (at load point) on the reel. Otherwise, specify the NOREWIND option and use the MTC job control statement (see *VSE/Advanced Functions*), for example, to properly position the tape.

NOREWIND
specifies that rewind is never to be performed on OPEN, CLOSE, and EOV.

End-of-file considerations: For EBCDIC standard-labeled tapes, the tape is positioned between the two trailing tapemarks. For unlabeled tapes, the tape is positioned following the single trailing tapemark.

Abbreviation: NREW

Default: NOREWIND

REWIND
specifies that tapes are to be rewound to load point but not unloaded on an OPEN, CLOSE, or EOV condition.

Abbreviation: REW

UNLOAD
specifies that tapes are to be rewound on an OPEN, and rewound and unloaded on an EOV and CLOSE condition.

Abbreviation: UNLD

PRIMEDATADEVICE(*devtype*)
specify this parameter only if the file is on tape (in which case you specify 2400). The 2400 identifies all tape devices that DTFMT supports. You must specify SYS005 in the ASSGN statement for a tape output file. See "Magnetic Tape Considerations for Access Method Services" in *VSE/VSAM Programmer's Reference* for more information on tape.

Abbreviation: PDEV

RECORDFORMAT(*format*)
specifies the format of the records in a nonVSAM file. This parameter is required. For *format*, substitute one of the following values:

Value	Meaning
FIXUNB|F	Fixed, unblocked
FIXBLK|FB	Fixed, blocked
VARUNB|V	Variable, unblocked
VARBLK|VB	Variable, blocked
SPNUNB|S	Variable spanned, unblocked
SPNBLK|SB	Variable spanned, blocked
UNDEF|U	Undefined

VARBLK must be specified if a catalog is to be unloaded into a sequential file.

Abbreviation: RECFM

RECORDSIZE(*size*)
specifies the length of a logical record for FIXBLK or FIXUNB nonVSAM files. For UNDEF, SPNBLK, or SPNUNB nonVSAM files, this parameter is the maximum record size. This parameter is not necessary for VARUNB. Record size must be 516 if a catalog is to be unloaded into a sequential file.

Abbreviation: RECSZ

Default: If RECORDSIZE is not specified, blocking is one record per block, therefore, RECORDSIZE is equal to block size for files that are FIXUNB, FIXBLK, or UNDEF; RECORDSIZE is equal to block size minus four for files that are VARUNB, VARBLK, SPNUNB, or SPNBLK.

FROMADDRESS(*address*) | FROMKEY(*key*) |
FROMNUMBER(*number*) | SKIP(*count*)
specifies the first record to copy from the input file.

FROMADDRESS(*address*)
specifies the RBA of the first record. It must be the beginning of a logical record. If you specify this parameter for a key-sequenced file, the records will be copied in physical sequential order instead of in key order. The address may be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

FROMKEY(*key*)
specifies the key of the first record. It may be a full key or a generic key--that is, the high-order portion of a key. If you specify a generic key, copying begins at the first record whose key matches the portion of the key you specified. (You must not specify a key longer than that defined for the file. If you do, the file is not copied.) If the specified key is not found, listing begins with the next higher key. 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 if the key is enclosed in single quotation marks; it may contain up to 255 bytes, that is, 510 hexadecimal or 255 EBCDIC characters. If it is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks.

FROMNUMBER(*number*)
specifies the relative-record number of the first record.

SKIP(*count*)
specifies the number of records you want to skip before beginning to copy records. For example, if you want copying to begin with the 500th

record, you specify SKIP(499). The count can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form and be no longer than one fullword (4 bytes or 32 bits).

Abbreviations: FADDR, FKEY, FNUM

Default: If nothing is specified, the copying begins with the first record.

Restrictions: Cannot be used when input file is a catalog.

Parameter	Can be used with	Cannot be used in combination with	Path processing
FROMADDRESS	Key-sequenced and entry-sequenced files	TOKEY TONUMBER	Not permitted
FROMKEY	Key-sequenced and indexed-sequential files	TOADDRESS TONUMBER	Permitted
FROMNUMBER	Relative-record files	TOADDRESS TOKEY	Not permitted
SKIP	Indexed-sequential, sequential and VSAM files		Unpredictable results

TOADDRESS (*address*) | TOKEY (*key*) | TONUMBER (*number*) | COUNT (*count*)

specifies the last record to copy in the file. The last record must follow the first record.

TOADDRESS (*address*)

specifies the RBA of the last record. Unlike FROMADDRESS, the RBA 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 file, the copying will be in physical sequential order instead of in key order. The address can be expressed in decimal (n), hexadecimal (X'n'), or binary (B'n') form.

TOKEY (*key*)

specifies the key of the last record. It may be a full key or a generic key—that is, the high-order portion of a key. If you specify a generic key, copying stops after the last record is copied whose key matches the portion of the key you specified. (You must not specify a key longer than that defined for the file. If you do, the copying is not performed.) If the specified key is not found, copying ends with the next lower key. 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 if the key is enclosed in single quotation marks; the key may contain up to 255 bytes, that is, 510 hexadecimal or 255 EBCDIC characters. If it is specified in hexadecimal, it must be preceded by X and be enclosed in single quotation marks.

TONUMBER (*number*)

specifies the relative-record number of the last record.

COUNT (*count*)

specifies the number of records to be copied. Processing is terminated when the number of records copied exceeds the value supplied by COUNT. The count can be expressed in decimal (n), hexadecimal

(X'n'), or binary form and be no longer than one fullword (4 bytes or 32 bits).

Abbreviations: TADDR, TNUM

Default: If nothing is specified, the copying ends with the last record.

Restrictions: Cannot be used when input file is a catalog.

Parameter	Can be used with	Cannot be used in combination with	Path processing
COUNT	Indexed sequential, sequential and VSAM files		Unpredictable results
TOADDRESS	Key-sequenced and entry-sequenced files	FROMKEY FROMNUMBER	Not permitted
TOKEY	Key-sequenced and indexed-sequential files	FROMADDRESS FROMNUMBER	Permitted
TONUMBER	Relative-record files	FROMADDRESS FROMKEY	Not permitted

REPLACE | NOREPLACE

specifies whether an input record which has a key (key-sequenced file) or record number (relative-record file) identical to the key or record number of a record in the output file is to replace that record.

If you specify NOREPLACE, VSAM issues a warning message that identifies the key or record number of the duplicate record; processing continues with the next record. Do not specify REPLACE if the output file is:

- An entry-sequenced or sequential file.
- A path over an alternate index.
- A base cluster with a unique-key alternate index in the upgrade set.

Abbreviations: REP, NREP

Default: NOREPLACE

REUSE | NOREUSE

specifies whether the output file will be opened with the reset option, that is, with its "high-used RBA" set to 0. If REUSE is specified and the file is not empty and is defined as nonreusable (NOREUSE), REPRO will terminate.

Abbreviations: RUS, NRUS

Default: NOREUSE

RESETCAT

The RESETCAT command is used with recoverable catalogs to regain access to data. This is done without moving and reorganizing data. The catalog recovery area (CRA) on each volume owned by the recoverable catalog is central to the 'reset' process. When the RESETCAT command is invoked the CRA (s) of the volumes indicated in the command parameters are informationally compared with the catalog entries. If there is a mismatch, the information in the catalog is replaced with that from the CRA(s). This can be used advantageously when a volume must be restored or the catalog reloaded. See "Using RESETCAT: Resetting Catalog Entries" for examples and more information.

The format of the RESETCAT command is:

RESETCAT	CATALOG (*catname* [/ *password*] ƀ *dname*) CRAFILES ((*dname* [ƀALL | NONE]) [(*dname* { ALL | NONE }) . . .]) CRAVOLUMES ((*volser* { ƀALL | NONE }) [. . .]) WORKCAT (*catname* [/ *password*] ƀ *dname*) [WORKFILE (*dname* [/ *password*])] [WORKVOLUMES (*volser* [/ *password*] [ƀ *volser* . . .])] [MASTERPW (*password*)] [IGNORE | NOIGNORE]

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

RESETCAT Parameters

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

identifies the catalog to be reset; it is a required parameter. The catalog specified by *catname* must have been defined with the RECOVERABLE attribute. Exclusive control of the catalog is required throughout the duration of the command; also, no VSAM files cataloged in the catalog being recovered may be open. The master catalog may be reset while it is in use as a master catalog, or it may be connected to the system as a user catalog.

password – specifies the master password of the catalog if it is password protected.

Abbreviation: CAT

CRAVOLUMES ((*volser* { ƀALL \| NONE }) [(*volser* { ƀALL \| NONE }) . . .])

specifies a list of volumes to be used to reset the catalog. You must specify the *volser* subparameter for each CRA to be accessed, whether or not any catalog records are to be reset in the catalog (see the NONE subparameter for more information).

See *VSE/VSAM Programmer's Reference* for further information about the contents of CRAs and how catalog entries are placed in CRAs.

volser
specifies the volumes containing CRAs to be accessed. They need not be premounted (VSAM will issue mount requests for unmounted volumes).

ALL
specifies that the catalog records in the CRA on the volume specified by *volser* are to be reset in the catalog. This is the default.

NONE
specifies that the catalog records in the CRA of the volume specified by *volser* are not to be reset in the catalog.

If a volume being recovered contains a multivolume file, other volumes of that multivolume file may be needed during the reset operation; they must be specified in the CRAVOLUMES volume list and NONE specified after the CRAVOLUMES *volser* parameter for the volume if you do not wish the entries in the CRA's of the additional volumes to be reset.

Abbreviation: CRAVOL

WORKCAT (*catname* [/ *password*])
catname – specifies the name of the catalog where the work file is to be defined. **This is a required parameter**. This catalog must not be the catalog being reset.

password – specifies the update or higher level password if the catalog is password protected.

Abbreviation: WCAT

WORKVOLUMES (*volser* [/ *password*]) [*volser* ...]
volser – specifies the volumes to be used by RESETCAT for the work file. The work file will be defined by RESETCAT and used as temporary storage while processing the command; it will be deleted at the end of the command. This file must not already be defined. If WORKVOLUMES is not specified, the work catalog is searched for a relative record file default model. If a model is found, the work file will be built on the volumes specified in the model's volume list, otherwise processing terminates. See "Work File Space Requirements" in "Using RESETCAT: Reseting Catalog Entries" for work file space requirements.

password – If desired, *password* may be specified to protect the workfile. Since the workfile will contain a copy of the catalog records, it may contain security-sensitive information. The *password* specified becomes the workfile's master password.

Abbreviation: WVOL

MASTERPW (*password*)
specifies the master catalog's master password. The master catalog's master password is required when the master catalog is password protected; a password prompt will occur if the master password is not specified.

Abbreviation: MRPW

IGNORE | NOIGNORE
specifies whether RESETCAT should continue after certain errors and force the reset of the catalog.

IGNORE
specifies that resetting continues even if errors occur. For example, if IGNORE is specified and an I/O error occurs while processing a record for a CRA, that record would be ignored. In the process, files that could not be reset from a particular volume would be marked unusable on that volume, and the space that was assigned to them would be freed. The errors that are ignored are:

- I/O error in the catalog (any VSAM request macro error code)
- I/O error in the CRA (any VSAM request macro error code)
- Error in the volume record in the CRA

NOIGNORE
specifies that all errors result in termination.

Abbreviation: IGN and NIGN

Default: NOIGNORE

VERIFY

The VERIFY command (VFY) is used to compare the end-of-file information as it is stored in a VSAM catalog with the end-of-file indicator(s) in the file itself. If the information in the catalog does not agree with the end-of-file indicator(s) in the file, the catalog information is corrected. The VERIFY command can be used following a system failure that prevented a component opened for output processing from being closed or that caused it to be improperly closed.

Clusters, alternate indexes, their components, paths defined over base clusters, and catalogs can be verified; paths defined over an alternate index cannot be verified. To avoid possible errors, do not specify the file (key-sequenced or alternate index) to be verified at the data component or index component level; always specify it at the cluster level.

VERIFY does not correct statistics that may be wrong as a result of a system failure. If the VERIFY command is to be used to recover from a failure during initial loading of a file, the RECOVERY parameter (see DEFINE CLUSTER) must have been specified or defaulted to when the file was defined.

If the SPEED parameter was specified when the file was defined, and VERIFY is executed after a failure during initial load, no attempt is made to compare and recover the end-of-file indicators. No message is issued to describe this condition (SPEED mode and no end-of file recovery).

See "Verifying a File's Accessibility" for examples and more information.

The format of the VERIFY command is:

VERIFY	DATASET (*entryname* [/ *password*]) FILE (*dname* [/ *password*])

Note: Ignore the shaded parameters if you are using the job control simplification introduced in Release 2. The shaded parameters are included for the convenience of users who are using the Release 1 method of job control (see Appendix J for their explanation).

VERIFY Parameters

DATASET (*entryname* [/ *password*])

entryname – specifies the *entryname* of the object to be verified.

The object to be verified must be in (or be) the default catalog. (The default catalog is (1) the job catalog if one is being used, or (2) the master catalog.)

password – is the control or master password of a password protected cluster, alternate index, or component, or the master password of a password protected catalog.

Abbreviation: DS

Modal Command Format

The modal commands are:

- IF-THEN-ELSE command sequence, which is used to control functional command execution by testing condition codes.
- SET command, which is used to set or reset condition codes.
- PARM command, which is used to specify options for diagnostic aids, printed output, and input record margins.

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. Informational messages may have been issued.
- 4, which indicates that a problem was met in executing the function, but it was possible to continue and complete the function. The results might not be exactly what you wanted, but no permanent harm has been done. For example, with LISTCAT, the system might have been unable to locate an entry to be isted. A warning message was issued.
- 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.
- 12, which indicates that the requested function could not be performed This condition code might be set because Access Method Services was unable to open a user file in REPRO or PRINT because you specified FROMKEY or TOKEY when the file was neither indexed sequential nor key sequenced, or because you named an unavailable catalog in the CATALOG parameter of DEFINE, ALTER, etc.
- 16, which indicates that a severe error occurred and the remainder of the command stream was flushed. This condition code might be set because a system output file could not be opened, an unrecoverable error occurred in a system file, or Access Method Services encountered improper command sequences.

IF-THEN-ELSE

The IF-THEN-ELSE command sequence is used to control command execution.

The format of the IF-THEN-ELSE command sequence is:

IF	{LASTCC|MAXCC} *comparand*␣*number* THEN[␣*command*| DO *commandset* END] [ELSE[␣*command*| DO *commandset* END]]

where

IF
specifies that one or more functional commands are to be executed based on a test of a condition code. The condition code is set by a SET command or is set by Access Method Services to reflect the completion status of a previous functional command.

LASTCC
specifies that the condition code that resulted from the immediately preceding function command is to be compared as indicated by *comparand*␣*number* to determine whether the THEN action is to be performed. LASTCC is initialized to zero upon entry to Access Method Services. See "Condition Codes" for the meaning of condition codes.

MAXCC
specifies that the maximum condition code that resulted from previous functional commands or SET commands is to be compared as indicated by *comparand*␣*number* to determine whether the THEN action is to be performed. MAXCC is initialized to zero upon entry to Access Method Services. See "Condition Codes" for the meaning of condition codes.

comparand – specifies the comparison to be made between the condition code specified by LASTCC or MAXCC and the following number. This can be any of six possible comparisons:

Equal, specified as "=" or "EQ"
Not equal, specified as "NE"
Greater than, specified as ">" or "GT"
Less than, specified as "<" or "LT"
Greater than or equal, specified as "GE"
Less than or equal, specified as "LE"

number – specifies the decimal integer to which LASTCC or MAXCC is to be compared. The number can be up to ten digits long. Values greater than 16 are reduced to 16. See "Condition Codes" 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 IF statement is true. THEN can be followed by another IF statement.

DO–END (see "DO-END Command Sequence")

ELSE

specifies that a single command or a group of commands (introduced by DO) are to be executed if the IF statement is false. ELSE can be followed by another IF statement. See "Nested IF Statements" for a discussion of IF following THEN or ELSE.

Command Example:

If you want to copy the file identified by MYDATA01 only if the last condition code was zero, specify:

```
IF LASTCC = 0 -
THEN REPRO INFILE(MYDATA01) OUTFILE(MYOUT02)
```

Null Clauses

A null clause is a THEN or ELSE clause that is not followed by a command or a continuation character. If no THEN action is required, the null THEN must be specified; a null ELSE need be specified only if required to balance THENs and ELSEs in nested IF statements.

If you want to indicate a null ELSE clause, specify:

```
ELSE
```

A null ELSE clause indicates that no action is to be taken if the IF statement is false.

If you want to indicate a null THEN clause, specify:

```
IF...THEN
ELSE...
```

A null THEN clause indicates that no action is to be taken if the IF statement is true.

Command Examples:

This example illustrates a null ELSE clause.

```
IF LASTCC = 4 THEN PRINT ...
ELSE
```

No action is taken if LASTCC doesn't equal 4.

The next example illustrates a null THEN clause.

```
IF LASTCC = 4 THEN
ELSE PRINT...
```

No action is taken if LASTCC does equal 4.

The examples above illustrate THEN and ELSE clauses that have been made null simply by the absence of a following command or continuing hyphen. Alternatively, a semicolon may be used, sInce a semicolon also terminates a command.

Nested IF Statements

When an IF statement appears in a THEN or ELSE clause, it is called a nested IF statement. Nested IF statements are coded exactly like unnested IF statements. The maximum number of IF statements in a nest is 10, including the unNested IF.

Within a nest of IF statements, each ELSE is matched with the nearest preceding unmatched THEN. If an IF statement does not require an ELSE clauSe, code a null ELSE clause after it, unless no other ELSE clause follows.

(If an ELSE clause follows, it would be taken for a missing preceding one — the second example below has a null ELSE clause to avoid this.)

Command Examples:

```
IF LASTCC > 4 -
  THEN IF MAXCC < 12 -
    THEN REPRO ...
    ELSE DELETE ...
  ELSE IF LASTCC = 4 -
    THEN
    ELSE PRINT ...
```

The example above specifies that if LASTCC is greater than 4, then MAXCC is to be tested. If MAXCC is less than 12, the REPRO command is executed, while if MAXCC is 12 or greater, the DELETE command is executed instead. However, if 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.

The next example is similar to the first, except for the use of a null ELSE clause.

```
IF LASTCC > 4 -
  THEN IF MAXCC < 12 -
    THEN REPRO ...
    ELSE
  ELSE IF LASTCC = 4 -
    THEN PRINT ...
```

If LASTCC is greater than 4, but MAXCC is 12 or greater, no functional command is executed. The null ELSE clause is employed here to cause the next ELSE to correspond to the first THEN.

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 test in an IF command. The set of commands is terminated by END. The first command following a DO must begin on a new line.

END specifies the end of a set of commands initiated by the nearest preceding unended DO. END must be on a line by itself.

If THEN or ELSE is not followed by DO, by a continuation character, or by a command in the same record, the THEN or ELSE is null--it results in no action.

Command Examples:

If you want to list a catalog and print a file if the maximum condition code is zero, specify:

```
IF  MAXCC = 0  THEN  DO
  LISTCAT  CATALOG (AMASTCAT/MST27) ENT (MN01.0005)
  PRINT INFILE (AJK006)
  END
ELSE ....
```

If you want to list a catalog and print a file if the last condition code is zero, but otherwise list its catalog entry before and after a VERIFY command, specify:

```
IF  LASTCC = 0 THEN    DO
    LISTCAT ...
    PRINT INFILE (AJK006)
    END
  ELSE DO
    LISTCAT ENTRY (AJK006) ALL
    VERIFY DATASET (FILE001)
    LISTCAT ENTRY (AJK006) ALL
              END
```

DO groups and nested IF statements may be combined. In the next example, a file is printed if LASTCC is equal to zero and is reproduced if MAXCC is equal to zero. Otherwise, the file's catalog entry is listed.

```
IF LASTCC = 0 -
  THEN DO;
    PRINT INFILE (AJK006)
    IF MAXCC=0 -
      THEN REPRO INFILE (AJK006) OUTFILE (AJK007)
    END;
  ELSE LISTCAT ENTRY (FILE006)
```

Notice that a null ELSE clause is not required preceding the END command; it is assumed because of the presence of END.

PARM

The PARM command specifies processing options for diagnostic aids and printed output to be used during execution. These options remain in effect until changed by another PARM command.

The format of the PARM command is:

PARM	[TEST({[TRACE] [AREAS(*areaid* [*areaid* . . .])] [FULL((*dumpid* [*count1* [*count2*]]) [(*dumpid* . . .)])] | [OFF] })] [GRAPHICS(CHAIN(*chain*) |TABLE(*mname*))] [MARGINS(*leftmargin* *rightmargin*)]

where:

TEST({[TRACE]
[AREAS(*areaid* [*areaid* . . .])]
[FULL((*dumpid* [*count1* [*count2*]])
[(*dumpid* . . .)])] |
[OFF] })

specifies the diagnostic aids to be used. Once the TEST option has been established, it remains in effect until end of job step or 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* [*areaid* . . .]

identifies modules that are to have selected variables dumped at their dump points. Each *areaid* is a two-character area-identifier defined within the implementation. See *VSE/VSAM Access Method Services Logic* for more information.

FULL((*dumpid* [*count1* [*count2*]]) [(*dumpid* . . .)])

specifies that a partition 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 *VSE/VSAM Access Method Services Logic* for more information.

Count1 is a decimal integer that specifies the number of times the program is to encounter the dump point before beginning the dump listing.

Count2 is a decimal integer that specifies the number of encounters at a dump point for which dumps are to be listed.

Default: The default for *count1* and *count2* is 1.

Restrictions: *Count1* and *count2* must range between a minimum of 1 and a maximum of 32,767.

OFF

specifies that use of diagnostic aids is to stop.

GRAPHICS(CHAIN(*chain*) | TABLE(*mname*))

specifies the print chain/train graphic character set or a special graphics table to be used in producing the output.

CHAIN({AN|HN|PN|QN|RN|SN|TN})

specifies the graphic character set of the print chain or train. PN is the default.

TABLE(*mname*)

specifies the name of a module (accessible through VSE CDLOAD) in the core image library that contains a 256-byte user-provided translate table. This table defines the graphic characters for each of the possible 256 bit patterns. 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).

Restriction: If you specify the TABLE parameter, issue the PARM command early in the Access Method Services command stream to avoid having CDLOAD fail. (Access Method Services loads the user-provided translate table by issuing a CDLOAD macro. VSE issues message IDC4999I UABORT CODE 52, if more than 50 CDLOADS are issued.)

Abbreviation: GRPH

MARGINS (*leftmargin* ↄ *rightmargin*)

specifies that the margins of input records on which command statements and comments are written are to be changed. The standard left and right margins are 2 and 72, respectively. If MARGINS is coded, all subsequent input records are scanned in accordance with the new margins. MARGINS may be used to cause commands coded like comments (between /* and */) to be active or inactive: respecification of margins could be used to cause the /* and */ characters to be omitted from the scan or included.

leftmargin – specifies the left margin.

rightmargin – specifies the right margin.

Abbreviation: MAR

Restriction: The right margin value must be greater than the left margin value.

SET

The SET command is used to change or reset a previously defined condition code. See "Condition Codes" for the meaning of condition codes.

The format of the SET command is:

SET	{MAXCC | LASTCC}= *number*

where

SET
specifies that a condition code is to be set. A SET command that follows a THEN or ELSE clause that is not executed does not change or reset a condition code.

MAXCC
specifies that the maximum condition code set by a previous functional command is to be reset. Setting MAXCC does not affect the value of LASTCC even if LASTCC was the maximum condition code set previously.

LASTCC
specifies that the condition code set by the immediately previous functional command is to be reset.

The symbol EQ may be used instead of the = sign.

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 LASTCC. All processing terminates when MAXCC or LASTCC is set to 16.

Command Examples:

If you want to set the last condition code established to 12, specify:

```
SET LASTCC=12
```

If you want to replace the highest condition code established so far in processing with 8, specify:

```
SET MAXCC=8
```

Chapter 4: Introduction to the VSAM Macros

This chapter identifies and briefly describes the macro instructions that are used to open and close a file, manage VSAM control blocks, and issue data processing requests. Format descriptions and examples of each macro are in the following chapter.

Assembler Programming Considerations

VSAM macro instructions are coded in an assembler language processing program to gain access to the data. There are macros for:

- Specifying parameters that relate the program and the data. These identify the file and describe the kind of processing to be done.
- Connecting and disconnecting a processing program and a file. These prepare a bridge for VSAM between the program and the data.
- Manipulating the information relating the program and the data. These are used to specify changes in processing.
- Requesting access to a file. These initiate the transfer of data between direct access storage and virtual storage.
- Sharing resources among files. These are used (a) to build and delete a VSAM resource pool for sharing I/O buffers and control blocks, and (b) to write out buffers.
- Displaying catalog information about VSAM files (mainly needed for the proper definition of the resource pool).

Specifying Parameters that Relate the Program and the Data

To open a file for processing, you must identify the file and the types of processing to be done. The ACB macro specifies the file to be processed and the types of access you want to use, the EXLST macro specifies a list of user-supplied exit routines, and the RPL macro specifies information for a request to access a particular record in the file. These are the declarative macros of VSAM. The GENCB macro can be used in place of the ACB, EXLST, or RPL macros to generate processing specifications during the execution of a processing program rather than while assembling or compiling the program.

ACB: Specifying the Access-Method Control Block

The ACB macro specifies an Access-Method Control Block (ACB). Each VSAM file has an ACB; it identifies the file and contains information about it. The filename of the DLBL job control statement that describes the file is included, so that the Open routine can connect the program to the data.

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

- The address of a list of exit-routine names that you supply (EXLST paramenter). You use the EXLST macro, described next, to construct the list.
- The amount of space for I/O buffers (BUFSP parameter) and the number of I/O buffers (BUFND and BUFNI parameters) that VSAM will use to process data and index records. The minimum number of buffers allowed depends on how much buffer space is allocated, the number of concurrent requests to be allowed, and whether processing will be direct or sequential.

- The password, if required, for the level of authorization to access the file (read, read and update, etc.)(PASSWD parameter).
- The processing options to be used (MACRF parameter): keyed or addressed access, or both; sequential, direct, or skip sequential processing, or a combination; retrieval, storage, or update (including deletion), or a combination; whether to use the shared resource pool and to defer the writing of updated records.
- For processing concurrent requests (STRNO parameter), the number of requests that are defined for processing the file (see the discussion of the RPL macro following EXLST).
- Address and length of an area for error messages from OPEN, CLOSE, or TCLOSE (MAREA and MLEN parameters).

EXLST: Specifying the Exit List

The EXLST macro specifies the addresses of optional exit routines that you can supply for analyzing physical and logical errors, for end-of-file processing, for overlapping I/O operations, and for writing a journal. Any number of ACBs in a program can indicate the same exit list, and an exit routine can be used for more than one file.

Analyzing Physical Errors (SYNAD): When VSAM encounters an error in an I/O operation that the DOS/VSE error recovery routines cannot correct, it exits to the physical-error analysis (SYNAD) routine. VSAM sets a code in the RPL to indicate whether the I/O error occurred during reading or writing the data or the index.

Analyzing Logical Errors (LERAD): Errors not directly associated with an I/O operation, such as an invalid request, cause VSAM to exit to the logical-error analysis (LERAD) routine. VSAM sets a code in the RPL that indicates the type of logical error.

End-of-File Processing (EODAD): When your program requests a record beyond the last record in the file during sequential access, your end-of-file (EODAD) routine is given control. The last record is the highest-addressed record for addressed or control-interval access or the highest-keyed record for keyed access. If an EODAD exit routine is not available, control is given to the LERAD exit routine.

Overlapping I/O Operations (EXCPAD): When VSAM starts an I/O operation caused by a request macro, the execute-channel-program (EXCPAD) exit routine is given control. The EXCPAD routine must return control to VSAM, which continues your mainline routine at the instruction following the request macro. The EXCPAD exit is intended for use by programmers of utilities and systems.

Writing a Journal (JRNAD): You can use the JRNAD routine to journalize the transactions made against your file and to keep track of RBA changes.

For recording transactions, VSAM exits to the JRNAD routine each time your processing program issues a GET, PUT, or ERASE. For keeping track of RBA changes, VSAM takes the JRNAD exit each time data is shifted within a control interval or moved to another control interval. To process a key-sequenced file with addressed access, you need to know whether any RBAs have changed during keyed processing. VSAM takes the JRNAD exit before transmitting to direct-access storage the contents of a control interval in which there was an RBA change.

RPL: Specifying the Request Parameter List

The RPL macro produces a Request Parameter List (RPL) which contains all the information needed by a request macro to access a record in the file. The request macros are GET, PUT, POINT, ERASE, and WRTBFR. The RPL identifies the file to which the request is directed by naming the ACB of the file.

You can use a single RPL to define parameters that apply to several requests. With the MODCB macro (described below) you can modify some of the parameters to change the type of processing, such as from direct to sequential or from update to non-update.

For concurrent requests, which require VSAM to keep track of more than one position in a file, any number of RPL macros may be used asynchronously by a processing program or its subtasks to process a file. The requests can be sequential or direct or both, and they can be for records in the same part or different parts of the file. You need specify only the RPL parameters appropriate to a given request, as follows:

Processing Options for a Request (OPTCD): A request is for keyed, addressed, or control-interval access. The processing can be sequential, skip sequential (keyed access only), or direct. For keyed and addressed access and for sequential or direct processing, records may be retrieved in backward direction. A request may be for updating or not updating a record. A non-update direct request to retrieve a record can optionally cause positioning at the following record for subsequent sequential access.

For a keyed request, you specify either a generic key or a full key to which the key field of the record is to be matched. A generic key can match several records while a full key matches only one record. You can also specify that, if the key does not match the key of any record in the file, the record with the next greater key will be processed.

For retrieval, a request is either for a data record to be placed in a work area in the processing program (move mode) or for the address of the record within VSAM's I/O buffer to be passed to the processing program (locate mode).

Address of the Work Area for, or Pointer to, a Data Record (AREA): For retrieval, update, insertion, or addition of a record, you must provide a work area in which the record is to be processed (move mode). For retrieval, you can have VSAM give you the address of the record within VSAM's I/O buffer (locate mode) in this field.

Size of the Work Area for a Data Record (AREALEN): This parameter specifies either the length of the work area in which a record is placed (for move mode) or the four-byte address of the record in VSAM's I/O buffer (for locate mode). Having a work area that is too small is considered a logical error.

Length of the Data Record Being Processed (RECLEN): For storage, your processing program indicates the length to VSAM; for retrieval, VSAM indicates it to your program.

Length of the Key (KEYLEN): This parameter is required only for processing by generic key. For ordinary keyed access, the full key length is available from the catalog.

Address of the Area Containing the Search Argument (ARG): The search argument is either a key (including a relative-record number) or an RBA. If the OPTCD parameter indicates a generic key, the KEYLEN parameter tells how many high-order (leftmost) bytes of the search argument will be used.

Address of the Next RPL in a Chain (NXTRPL): You can process several records with a single GET, PUT, or ERASE by chaining RPLs together. For example, each RPL in a chain could contain a unique search argument and point to a unique work area. A single GET macro would retrieve a record for each RPL in the chain. A chain of RPLs is processed as a single request. (Chaining RPLs is not the same as issuing concurrent requests that require VSAM to keep track of multiple positions in a file.)

Transaction-ID (TRANSID): With this parameter you can create a logical relationship between I/O requests issued for different VSAM files.

GENCB: Generating Control Blocks and Lists

You can use the GENCB macro to generate an ACB, EXLST, or RPL during the execution of your processing program, rather than to assemble it with the corresponding macro. GENCB is coded in the same way as the other macros, but it generates one or more copies of a control block or list and allows you to code parameter values in more ways.

Connecting and Disconnecting a Processing Program and a File

OPEN connects a processing program to a file, so that VSAM can satisfy the program's request for data. CLOSE completes processing and frees resources that were obtained by the OPEN routine. TCLOSE causes buffers to be written out and the catalog to be updated.

OPEN: Connecting a Processing Program to a File

The OPEN macro calls the Open routine, which verifies that the processing program has authority to process the file, constructs VSAM control blocks and establishes linkages to VSAM routines. By examining the DLBL statement indicated by the DDNAME operand in the ACB macro and the volume information in the catalog, Open verifies that the necessary volumes have been mounted. When you are opening a key-sequenced file or an alternate index, VSAM issues an error code to warn you if the data has been updated separately from its index.

CLOSE: Disconnecting a Processing Program from a File

The Close routine completes any I/O operations that are outstanding when a processing program issues a CLOSE macro for a file. It writes any output buffers that have not been stored.

Close updates the catalog entries for any changes in the attributes of a file; it also updates the statistics on file processing (such as number of records inserted). The addition of records to a file may cause its end-of-file indicator to change, in which case Close updates the end-of-file indicator in the catalog. These end-of-file indicators help ensure that the entire file is accessible. If an error prevents VSAM from updating the indicators, the file is flagged as not properly closed. When a processing program subsequently issues an OPEN macro, it is given an error code indicating the failure.

Because it is essential for the integrity of a file that it is closed properly, DOS/VSE automatically attempts to close all open VSAM files within the partition at both normal and abnormal termination of a job step. If any control blocks for a file have been destroyed through an error in your pro-

gram, this file cannot be closed and a message is issued to the operator. EXLST routines are not entered during automatic CLOSE.

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

TCLOSE: Securing Records Added to a File

The TCLOSE macro performs the functions of CLOSE, except that it leaves the program and the file connected so that you can continue processing without reopening the file. You can use the TCLOSE macro to protect data while the file is being loaded or extended.

Manipulating the Information Relating the Program and the Data

The MODCB, SHOWCB, and TESTCB macros are used for modifying, displaying, and testing the contents of an ACB, EXLST, or RPL.

MODCB: Modifying the Contents of Control Blocks and Lists

The MODCB macro is used to specify new values for fields in an ACB, EXLST, or RPL. For example, to use a single RPL to retrieve directly the first record having a certain generic key and then to retrieve sequentially the rest of the records having that generic key, you would use MODCB to alter the RPL to change from direct to sequential access.

SHOWCB: Displaying Fields of Control Blocks and Lists

SHOWCB allows you to examing the contents of fields in an ACB, EXLST, or RPL. VSAM displays the requested fields in an area you provide. You can also display fields in addition to those defined in the macros. For example, when a file is open, you can display various counts, such as number of control interval splits, number of deleted records, and number of index levels. The RBA of the last record accessed and the error codes set in the ACB or RPL after macro execution can also be displayed.

TESTCB: Testing the Contents of Control Blocks and Lists

The TESTCB macro enables you to test the contents of a field or combination of fields in an ACB, EXLST, or RPL for a particular value and to alter the sequence of your processing steps as a result of the test. Thus, TESTCB is similar to a branch instruction. You can test the error codes set in the ACB or the RPL, for instance, or the attributes of a file, such as record length.

Requesting Access to a File

All of the preceding macros prepare to process a file. The request macros, GET, PUT, POINT, ERASE, and WRTBFR, initiate an access to data. Another request macro, ENDREQ, is provided to (1) terminate processing of a request when completion is not required, or (2) free VSAM from having to keep track of a position in the file. Each of these macros is associated with an RPL (or chain of RPLs) that fully defines the request. The only parameter that is needed with a request macro is the address of the RPL that defines the request.

Sharing Resources among Files and Displaying Catalog Information

Normally, buffers and control blocks are allocated statically to a file at the time the file is opened; they are freed when the file is closed. As long as the file is open, these buffers and control blocks cannot be used by any other file.

The Shared Resources facility, however, allows you to share buffers, I/O control blocks, and channel programs among several VSAM files within a

partition, and to manage I/O buffers. These buffers and control blocks are allocated out of a common resource pool at the time you issue an I/O request for a file. When the request is satisfied, the same buffers and control blocks can be assigned to another file (for direct requests).

Sharing these resources optimizes their use and also reduces the amount of virtual storage required (the working set) per partition. The facility is especially useful in an environment in which (a) many VSAM files are open and it is therefore difficult to predict the amount of activity that will occur at a given time, or (b) each transaction may refer to several files.

Managing I/O buffers includes:

- Deferring write operations for direct PUT requests, thus reducing the number of I/O operations.
- Correlating deferred requests by a transaction ID.
- Writing out buffers whose writing has been deferred.

Managing I/O buffers should enable you to speed up direct processing of VSAM files whose activity is unpredictable.

When you share resources for sequential access, you have to establish positioning before you can issue your initial retrieval request, because with shared resources VSAM does not automatically position itself at the beginning of a file opened for sequential access. Also note that you may not use shared resources to load records into an empty file.

The macros you use to share resources and write I/O buffers are:

- BLDVRP (build VSAM resource pool)
- DLVRP (delete VSAM resource pool)
- WRTBFR (write buffer)

In addition, the SHOWCAT macro is provided to display, for non-open files, the catalog information needed for the proper specification of some of the BLDVRP operands.

The ACB, RPL, SHOWCB, MODCB, and TESTCB macros have been extended to provide for sharing resources and managing I/O buffers.

Chapter 5: VSAM Macro Instruction Format

This chapter describes how to code the VSAM macros; it gives the format of each macro instruction and examples of its use.

There are macros for:

- Specifying information that relates the program and the data
- Connecting and disconnecting a processing program and a file
- Defining requests for access to data
- Manipulating the information relating the program and the data
- Requesting access to a file
- Sharing resources among files and displaying catalog information.

Notational Conventions

In the VSAM macros, you can code **address** as a symbolic name that generates a relocatable A-type address constant. Except for the ACB, EXLST, and RPL macros, you can also code an address as a register, using either ordinary register notation (with registers 2 through 12) or, if shown in the format description as a decimal number in parentheses, special register notation. For example:

```
{RPL=address|(1)}
```

means that you can specify either a symbolic address or any of the registers 2 to 12 or register 1. If you do use register notation, the use of register 1 is preferable.

You can code a **value** (number) as any absolute expression, except for a self-defining character term.

You can code a **name** according to the rules of the assembler. The control block manipulation macros (GENCB, SHOWCB, MODCB, and TESTCB) can be coded in even more ways as shown in Appendix F.

Some operands of the VSAM macros can have more than one parameter. These operands are shown with parentheses around the parameters (for example, the MACRF operand of the ACB macro). This means that you can code the operand, if it has only one parameter, with or without parentheses around the parameter:

```
MACRF=option
MACRF=(option)
```

However, if the operand is coded with two or more parameters, enclosing parentheses are required:

```
MACRF=(option,option,...)
```

Specifying Information that Relates the Program and the Data

Before you can open a file for processing, you must identify the file and provide some basic information about the ways you are going to process it. To specify this information, you use the ACB (access method control block) macro and the EXLST (exit list) macro. In the ACB macro, you specify the file to be processed and the types of access you want to use. In the EXLST macro, you specify the addresses of exit routines that you supply.

As an alternative to using the ACB and EXLST macros, you can use the GENCB (generate control block) macro to generate processing specifica-

tions during the execution of a processing program, rather than during assembly or compilation of the program.

ACB Macro (Specifying an Access Method Control Block)

Assembly of the ACB macro produces an access method control block (ACB). The ACB identifies the VSAM file that you want to process and indicates the types of requests that you want to make. The ACB is similar to a DTF (define the file) in that it identifies the file to be processed. However, you specify most information (such as key length or record format) about the file in the DEFINE command of Access Method Services. That information then resides in the VSAM catalog and is brought into virtual storage when the ACB is opened. Refer to "Buffer Specifications" in *VSE/VSAM Programmer's Reference* for examples of ACB buffer parameter use.

Code the values for the ACB-macro operands as absolute numeric expressions, character strings, codes, and expressions that generate valid relocatable A-type address constants. Ordinary register notation with registers 2 through 12 can also be used for address.

The format of the ACB macro is:

```
name   ACB   [AM=VSAM]
              [,BUFND=number]
              [,BUFNI=number]
              [,BUFSP=number]
              [,DDNAME=filename]
              [,EXLST=address]
              [,MACRF=([ADR][,CNV][,KEY]
                   [,DFR|NDF]
                   [,DIR][,SEQ][,SKP]
                   [,IN][,OUT]
                   [,NRM|AIX]
                   [,NRS|RST]
                   [,NSR|LSR]
                   [,NUB|UBF]))]
              [,MAREA=address]
              [,MLEN=number]
              [,PARMS=(CLOSDSP={KEEP|DELETE|DATE})]
              [,PASSWD=address]
              [,STRNO=number]
```

`name`
one through eight characters that provide a symbolic address for the ACB that is assembled, and also serve (if you omit the DDNAME operand) as the filename of the DLBL job control statement.

`AM=VSAM`
specifies that this is a VSAM control block. You may want to specify this operand for documentation purposes if your installation also uses VTAM.

`BUFND=number`
specifies the number of I/O buffers to be used to hold control intervals containing data records. Each buffer is the size of one data control interval. The allowable minimum specification (and also the default) is the number specified for STRNO, plus one. (The default for STRNO is one.) If you specify the BUFND operand, but your specification is less than the

minimum, VSAM overrides your specification and uses the minimum. However, VSAM issues no message to inform you of this.

VSAM will increase the number of data buffers you specify if the amount of virtual storage available for buffers differs from the storage requirements indicated by the BUFND and BUFNI operands. See the BUFSP operand for an explanation.

`BUFNI=number`

specifies the number of I/O buffers to be used to hold index control intervals (index records). Each buffer is the size of one index control interval. The minimum number you can specify is the number specified for the STRNO operand. (If you omit STRNO, BUFNI must be at least one, because the default for STRNO is one.) If BUFNI is omitted, the default is the number specified for STRNO, because the smallest number of index buffers allowed is one for each string. If you specify the BUFNI operand, but your specification is less than the minimum, VSAM overrides your specification and uses the minimum. However, VSAM issues no message to inform you of this.

VSAM will increase the number of index buffers you specify if the amount of virtual storage available for buffers differs from the storage requirements indicated by the BUFND and BUFNI operands. See the BUFSP operand for an explanation.

`BUFSP=number`

specifies the size, in bytes, of an area for data and index I/O buffers. VSAM issues a GETVIS macro to obtain the buffer area in your processing partition. It must be at least as large as the buffer space size recorded in the catalog entry for the file. If your specification is too small, VSAM overrides it and uses the value recorded in the catalog for buffer space size. However, VSAM issues no message to inform you of this.

If you do not specify the BUFSP operand, the buffer space size will be the larger of (1) the size recorded in the catalog or (2) the size determined from the values specified for BUFND and BUFNI. (The size recorded in the catalog was specified by the BUFFERSPACE parameter in the DEFINE command of Access Method Services. If that parameter was omitted when the file was defined, a default value was set in the catalog by Access Method Services. This default value, the minimum amount of buffer space allowed by VSAM, is enough space for two data control intervals and one index control interval.)

You can also specify buffer space by means of the BUFSP=number operand on the DLBL statement that identifies the file to be processed. This value overrides the BUFSP operand in the ACB macro. It also overrides the BUFFERSPACE parameter in the DEFINE command if it is greater than the BUFFERSPACE parameter value.

If the values you code for BUFND, BUFNI, and BUFSP are not consistent with each other, VSAM increases the number of buffers to conform to the size of the buffer area.

If BUFSP is greater than the minimum requirements and greater than the BUFND and BUFNI requirements, the extra space will be allocated between data and index buffers as follows:

- If the ACB MACRF operand indicates direct processing only: First, BUFND and BUFNI are allocated as specified. Then, all additional space is allocated to index buffers.

- If the ACB MACRF operand indicates sequential processing: First, BUFND and BUFNI are allocated as specified. Then, one additional buffer is allocated to the index and the remaining space is allocated to data buffers. Any space remaining, but insufficient for a single data buffer, is allocated to an index buffer.

If BUFSP is greater than the minimum requirements, but less than the BUFND and BUFNI requirements, the buffer space will be changed to conform to the BUFND and BUFNI requirements.

If you provide your own pool of I/O buffers for control interval (CNV) access (MACRF=UBF), the BUFND, BUFNI, and BUFSP operands have no effect. The AREA and AREALEN parameters in the RPL macro then define the area for user buffers.

`DDNAME=filename`

specifies a character string of up to seven bytes and is the same as the filename parameter specified in the DLBL statement that identifies the VSAM file or path to be processed.

If the 'file-ID' in the DLBL statement indicates a path, but you want to process only the alternate index of the path, you must specify MACRF=AIX (see the discussion of the MACRF operand). If the file-ID does not indicate a path, the AIX option is ignored.

If you omit the DDNAME operand, you must specify the DLBL filename as the name (label) of the ACB macro.

`EXLST=address`

specifies the address of a list of user exit-routine addresses. The list is generated by the EXLST macro (or the GENCB macro). If you use the EXLST macro, you can specify its name (label) here as the address of the exit list. If you use the GENCB macro, you can specify the address of the EXLST returned by GENCB in register 1. Omitting this operand indicates that you have no user exit routines.

```
MACRF= ([ADR][,CNV][,KEY]
       [,DFR|NDF]
       [,DIR][,SEQ][,SKP]
       [,IN][,OUT]
       [,NRM|AIX]
       [,NRS|RST]
       [,NSR|LSR]
       [,NUB|UBF])
```

specifies the kind(s) of processing you will do with the file. The options must be meaningful for the file. For example, if you specify keyed access for an entry-sequenced file, you will not be able to open the file. You must specify all of the types of access you are going to use, whether you use them concurrently or by switching from one to the other.

"File Disposition" in *VSE/VSAM Programmer's Reference* provides information about the interaction between the DLBL DISP parameter and the ACB MACRF specification when a file is opened.

Figure 5-1 gives the options; they are arranged in groups, and each group has a default value (indicated by underlining). You can specify options in any order. You may specify both DIR and SEQ; with keyed access, you may specify SKP as well. If you specify OUT and want simply to retrieve some records as well as update, delete, or insert others, you need not also specify IN. You may specify both ADR and KEY to process a key-sequenced file.

Beginning with VSE/VSAM Release 2:

- You are no longer allowed to specify both KEY and ADR (or both KEY and CNV) for a SHAREOPTIONS 4 key-sequenced output file. Attempts to do this result in an OPEN failure.
- If a SHAREOPTIONS 4 key-sequenced file is open under one ACB for keyed output access, an attempt to open it under another ACB with MACRF=(OUT,KEY) or (MACRF=(OUT,CNV) will fail.
- If a SHAREOPTIONS 4 key-sequenced file is open under one ACB for addressed or control output access, an attempt to open it under another ACB with MACRF=(OUT,KEY) will fail.

ADR Addressed access (for key-sequenced and entry-sequenced files).
CNV Control-interval access.
KEY Keyed access (for key-sequenced or relative-record files).

DFR Write operations are to be deferred when possible.*
NDF Write operations are not to be deferred

DIR Direct processing.
SEQ Sequential processing.
SKP Skip sequential processing (for key-sequenced or relative-record files).

IN Retrieve records only.
OUT Retrieve, insert, add-to-end, or update records (keyed access); retrieve, update, or add-to-end (addressed access).

NRM The file (or path) named in the DDNAME operand is to be processed.
AIX The alternate index of the path specified in the DDNAME operand is to be processed. If no path is specified there, this option is ignored. The AIX option causes the path restrictions (that is, the restrictions limiting the access through a path) to be ignored so that the alternate index can be processed like a key-sequenced file. The alternate index of the path can be opened for input (IN), output (OUT), or it can be reset (RST), provided it was defined with the REUSE attribute (in the DEFINE ALTERNATEINDEX command).

NRS The Open routine does not reset the file to 'empty'. Output operations will cause updating or extension of the existing record. DISP=OLD on the DLBL statement is equivalent to MACRF=NRS and will override MACRF=RST.
RST The Open routine resets the catalog information about the file (cluster or alternate index) to its original status, that is, to the status it had before it was opened for the first time. The file must have been defined with the REUSE attribute for RST to be effective. Although the file is not erased, you can handle it like a new file and use it as a workfile. After the Open routine has performed the reset operation, the RST option is equivalent to the OUT option. DISP=NEW on the DLBL statement is equivalent to MACRF=RST and will override MACRF=NRS.

NSR Non-shared resources (normal operation)
LSR Local shared resources*

NUB No user buffers; VSAM supplies buffers for I/O operations (KEY, ADR, and CNV access).
UBF User buffers (only CNV access can be specified). VSAM will read and write control intervals in a buffer you supply. It is pointed to by the AREA parameter of the RPL.

Mutually exclusive options are:

AIX and NRM	NDF and DFR
IN and RST	LSR and UBF
NRS and RST	LSR and RST
NUB and UBF	NSR and DFR
NSR and LSR	

*See "Sharing Resources among Files" for more information.

Figure 5-1. MACRF Options

MAREA=address

specifies the address of an optional OPEN/CLOSE/TCLOSE message area. (See "OPEN/CLOSE/TCLOSE Message Area" under "Connecting and Disconnecting a Processing Program and a File.")

MLEN=number

specifies the length of an optional OPEN/CLOSE/TCLOSE message area. The default is 0, the maximum value can be 32,768 bytes.

PARMS=(CLOSDSP={KEEP|DELETE|DATE})

specifies the CLOSE disposition for a reusable file. Options specified in the DLBL's DISP=(,option) job control statement, override the options specified in this parameter.

- KEEP indicates that the file and its contents are to be preserved. DISP=(,KEEP) on the DLBL statement is equivalent to PARM=(CLOSDSP=KEEP) and will override any CLOSDSP specified in the ACB.
- DELETE indicates that the file and its contents need not be preserved. VSAM is free to release resources associated with the file. DISP=(,DELETE) on the DLBL statement is equivalent to PARM=(CLOSDSP=DELETE) and will override any CLOSDSP specified in the ACB.
- DATE indicates that disposition depends on the current system date and the file's expiration date. If the expiration date is yet future relative to the system date, the file is treated as though KEEP were specified. Otherwise the file is treated as though DELETE were specified. DISP=(,DATE) on the DLBL statement is equivalent to PARM=(CLOSDSP=DATE) and will override any CLOSDSP specified in the ACB.

If disposition parameters indicate that file resources can be freed, VSAM releases as many resources as allowed by the sharing status and by the characteristics defined for the file. For example:

- If the file is open for another ACB, disposition processing must be deferred to avoid destroying an in-use file.
- If the file was defined with NOREUSE, no resources can be freed.
- If the file was defined with REUSE (including implicitly defined files), secondary extents are released and the catalog is updated to indicate that the file is empty.
- If the file was defined with NOALLOCATE, all extents for the file are released and the catalog is updated to indicate that the file is empty.

In each of these cases of freeing resources note that the catalog entry for the file is not deleted.

PASSWD=address

specifies the address of a field that contains the highest-level password required for the type(s) of access indicated by the MACRF operand. The first byte of the field pointed to contains the length (in binary) of the password (maximum of 8 bytes). A zero in the length byte indicates that no password is supplied. If the file is password-protected and you do not supply a required password in the ACB, VSAM gives the console operator the opportunity to supply it when opening the file.

`STRNO=number`

indicates how many concurrent active requests VSAM is to handle. The default is one.

During initial load of a file, VSAM ignores your specification and sets the value to one because a file can be loaded by one string only. After the file is loaded and successfully closed, it can be reopened and processed by as many strings as specified under STRNO.

Several requests, with the corresponding RPLs pointing to the same ACB, can be active at the same time. Thus, you can access simultaneously (a) different parts of a file, and (b) in different modes of operation (sequential and direct, or keyed and addressed, for example). You may, for example, process one part of a file sequentially (forward or backward) and intermix sequential processing with direct requests to any part of the file, without affecting the sequential positioning.

Each request is activated by its own RPL and action (GET, PUT, etc.) macro. Positioning information is maintained separately for each RPL, so that each request can be processed independently from all other requests.

A request is defined either by a single RPL or by a chain of RPLs (see "Defining Requests for Access to Data." Specify for STRNO the total number of RPLs or chains of RPLs that you will use to define requests. For a chain of RPLs VSAM needs to remember only one position. However, each position beyond the minimum number that VSAM needs to remember requires additional virtual-storage space for:

- A minimum of one data I/O buffer and, for keyed access, one index I/O buffer (the size of an I/O buffer is the control-interval size of a file).
- Internal control blocks and other areas.

VSAM remembers its position in the file for any sequential or update request. For example, sequential access depends on VSAM being able to determine the location of the next record from the location of the present record. Updating or deleting a record depends on VSAM remembering its location after you retrieve it. Also, processing a record in the I/O buffer requires VSAM to remember its location in the buffer.

Note: If the number of concurrent requests (RPLs or chain of RPLs) exceeds the number you have specified in the ACB STRNO operand, you must disconnect a request from its RPL by means of the ENDREQ macro before you can issue another concurrent request. The ENDREQ macro is discussed under "Requesting Access to a File."

EXLST Macro (Specifying an Exit List)

Assembly of the EXLST (exit list) macro produces an optional list of addresses of user exit routines. An exit routine is entered when VSAM detects the condition (such as an I/O error) that the routine is supposed to handle. The exit list is associated with an ACB by the EXLST operand of the ACB macro. Two or more ACBs can refer to the same exit list.

The number of exit addresses in a list is variable and depends on the number of operands you code. You cannot add addresses to the list after it is generated, but you can change an address or the indication of whether or not an exit is active (with the MODCB macro).

Values for EXLST macro operands can be specified as codes and expressions that generate valid relocatable A-type address constants.

The format of the EXLST macro is:

```
[name]  EXLST   [AM=VSAM]
                [,EODAD=(address[,{A|N}][,L])]
                [,EXCPAD=(address[,{A|N}][,L])]
                [,JRNAD=(address[,{A|N}][,L])]
                [,LERAD=(address[,{A|N}][,L])]
                [,SYNAD=(address[,{A|N}][,L])]
```

name
: one through eight characters that provide a symbolic address for the exit list that is established.

AM=VSAM
: specifies that this is a VSAM control block. You may want to specify this operand for documentation purposes if your installation also uses VTAM.

EODAD
: specifies that an exit is provided for special processing when the end of a file is reached by sequential or skip sequential access.

EXCPAD
: specifies that an exit is provided to receive control from VSAM when an I/O operation is started

JRNAD
: specifies that an exit is provided for journalizing as you process data records.

LERAD
: specifies that an exit is provided for analyzing logical errors.

SYNAD
: specifies that an exit is provided for analyzing physical errors.

address
: is the address of a user-supplied exit routine. The address must always be specified first.

A | N
: specifies that the exit routine is active (A) or not active (N). VSAM does not enter a routine whose exit is marked not active.

L
: specifies that the address is the address of an eight-byte field that contains the name of an exit module in the core image library that VSAM is to load for exit processing. If L is omitted, the address gives the entry point of the exit routine in virtual storage. L can precede or follow the A or N specification.

EODAD Exit Routine to Process End-of-File: An EODAD routine finishes the processing of a file when VSAM reaches the end of the file. VSAM exits to this routine when: (1) you attempt to sequentially retrieve or point to a record beyond the last record in the file, that is, the record with the highest key or the highest relative-record number (for keyed access), or with the highest RBA (for addressed access); (2) during sequential backward retrieval when the records in reverse sequence are exhausted or; (3) when you have

specified control-interval access and user buffers and there is no more data after a GET request or a PUT for update request.

If your program retrieves records sequentially with a request defined by a chain of RPLs, your EODAD routine must determine whether the end of the file was reached for the first RPL in the chain. If not, then one or more records have been retrieved but not yet processed by your program.

If you do not have this exit routine, VSAM exits to the routine for analyzing logical errors (see the LERAD operand). If you do not have the LERAD exit routine, VSAM returns to your processing program at the instruction following the last executed instruction. In that case, register 15 contains X'08', and register 1 contains the address of the RPL. Your program can examine the feedback field in the RPL with the SHOWCB or TESTCB macro to see whether VSAM has reached the end of the file.

When VSAM exits to the EODAD routine, the contents of the registers are:

Register	Contents
0	Unpredictable.
1	Address of the request parameter list that defines the request that occasioned VSAM's reaching the end of the file. The register must contain this address if you return to VSAM.
2-13	Same as when the request macro was issued. Register 13, by convention, contains the address of your processing program's 72-byte save area, which may not be used as a save area by the EODAD routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the EODAD routine.

If the EODAD exit routine returns to VSAM and you issue another GET macro, VSAM enters the EODAD exit routine again. This can cause your program to loop. If, however, you reach end-of-file during keyed access and then change to addressed access, additional records may be retrieved provided they are physically after the last record in key sequence (because of a control-interval or control-area split).

EXCPAD Exit Routine: An EXCPAD routine receives control from VSAM when an I/O operation is started. By supplying an EXCPAD exit routine, you can overlap VSAM I/O operations with the execution of your processing program.

The exit routine must return to VSAM, so that VSAM can return to your processing program at the instruction following the I/O request macro.

When VSAM exits to the EXCPAD routine, the contents of the registers are:

Register	Contents
0	Unpredictable.
1	Address of a parameter list with the following contents: **Offset:** X'00' Address of the RPL. X'04' Address of the IORB. X'08' EXCPAD lock word. X'0C' 116 bytes available to the user.
2-13	Same as when the request macro was issued. Register 13, by convention, contains the address of the user's 72-byte save area, which must not be used as a save area by the EXCPAD routine (because EXCPAD must return control to VSAM).
14	Return address to VSAM.
15	Address of EXCPAD routine.

If the exit routine uses register 1, it must restore that register with the parameter-list address before returning to VSAM. (The routine must return for completion of the request that caused VSAM to exit.)

The EXCPAD routine may test the traffic bit of the IORB to determine whether the VSAM I/O operation has been completed. However, the routine must not change the contents of the IORB because the IORB is being used by VSAM.

The EXCPAD lock word normally contains zero, in which case the routine may issue any other VSAM request, except a CLOSE. When a control-interval or control-area split occurs, the lock word contains the address of the RPL for the request that is causing the split. In that case, the EXCPAD routine must either complete the request that is causing the split, because a second (simultaneous) split in the same file results in a system deadlock, or issue a request against another file. However, if SHAREOPTIONS(4) was specified for the file that is causing the split, no further split can be handled for any other file with SHAREOPTIONS(4), because the corresponding system resource is locked.

The EXCPAD exit routine may be entered more than once for a VSAM request because a request may require more than one I/O operation.

The EXCPAD routine is not entered in the following cases:

- When the I/O operation completes before VSAM is ready to wait on it.
- During processing to complete pending I/O at CLOSE time.
- During Upgrade processing.

JRNAD Exit Routine to Journalize Transactions: A JRNAD routine records transactions made against a file and keeps track of RBA changes. VSAM exits to this routine each time the processing program issues a GET, PUT, or ERASE and each time data is shifted right or left in a control interval or is moved to another control interval (because one or more records have been inserted, deleted, shortened, or lengthened). VSAM takes the JRNAD exit before transmitting to direct-access storage the contents of a control interval in which there was an RBA change. (You need to know whether RBAs have changed during keyed processing if later on you want to process your key-sequenced file with addressed access.)

VSAM also takes the JRNAD exit whenever a segment of a spanned record is transmitted to or from direct-access storage. This allows you to keep track of the control intervals occupied by a spanned record.

The JRNAD exit must return to VSAM for completion of the request that caused VSAM to exit.

When VSAM exits to the JRNAD routine, the contents of the registers are:

Register	Contents		
0	Unpredictable.		
1	Address of a parameter list with the following format:		
	4 bytes	Address of the request parameter list that defines the request that caused VSAM to exit to the routine.	
	4 bytes	Address of a 5-byte field that identifies the file being processed. This field has the format:	
		4-bytes	Address of the ACB that is specified by the RPL that defines the request that occasioned the JRNAD exit's being taken.
		1 byte	Reserved.
	4 bytes	For RBA changes, the RBA of the first byte of data that is being shifted or moved. For a GET or PUT request against a spanned record seg-	

Register	Contents	
		ment, the RBA of the first byte of the segment.
	4 bytes	For RBA changes, the number of bytes of data that is being shifted or moved (this number does not include free space, if any, or control information — except for a control-area split, when the whole contents of a control interval are moved to a new control interval). For a GET or PUT request against a spanned record segment, the number of bytes in the segment.
	4 bytes	For RBA changes only, the RBA of the first byte to which data is being shifted or moved.
	1 byte	Indication of the reason VSAM exited to the JRNAD routine: X'00' GET request X'04' PUT request X'08' ERASE request X'0C' RBA request X'10' GET request against a spanned record segment X'14' PUT request against a spanned record segment X'18' Reserved X'1C' Reserved
	1 byte	Reserved
2-13	Unpredictable.	
14	Return address to VSAM.	
15	Entry address to the JRNAD routine.	

If, in your exit routine, you intend to issue the GENCB, MODCB, SHOWCB, or TESTCB macros, make sure that you save the contents of register 14 before you issue the macro and restore these contents in register 14 before your exit routine returns to VSAM. The same applies accordingly if, in your exit routine, you intend to use register 1. Your exit routine must return to VSAM for completion of the request that caused VSAM to exit.

For journalizing transactions (when VSAM exits because of a GET, PUT, or ERASE), you can use the SHOWCB macro, for example, to display information in the RPL about the record that was retrieved, stored, or deleted – by specifying:

```
FIELDS=(AREA,KEYLEN,RBA,RECLEN)
```

You can also use the TESTCB macro to determine whether a GET or a PUT was for update (OPTCD=UPD).

You cannot use the keywords RBA or RECLEN to display the RBA or length, respectively, of a spanned record segment retrieved or stored. Instead, this information is given in the parameter list at offsets 8 and 12, respectively.

For recording RBA changes, you must calculate how many records there are in the data being shifted or moved, so you can keep track of the new RBA for each one. With fixed-length records, you calculate the number by dividing the record length into the number of bytes of data being shifted. With variable-length records, you could calculate the number by using a table that not only identifies the records (by associating a record's key with its RBA), but also gives their lengths.

Some control-interval splits cause data to be moved to two new control intervals, and control-area splits normally cause the contents of many control intervals to be moved. In these cases, VSAM exits to the JRNAD routine for each separate movement of data to a new control interval.

If your JRNAD routine only journals transactions, it should ignore calls with the reason code X'0C' and return to VSAM; conversely, if it only records RBA changes, it should ignore all calls with reason codes other than X'0C'.

The only journalling you can do during processing of a path is to record transactions made against the base cluster; access to the alternate index during retrieval of a base record or during upgrading cannot be journalled. Journalling for path processing is triggered by the specification of the JRNAD exit in the EXLST of the ACB identifying the base cluster.

The JRNAD exit must be indicated as active before the file for which the exit is to be used is opened, and the exit must not be made inactive during processing. If you define more than one ACB for a file and if you want to have a JRNAD routine, the first ACB you open for the file must specify the exit list that identifies the routine.

LERAD Exit Routine to Analyze Logical Errors: A LERAD routine analyzes logical errors and all other error conditions except I/O errors encountered by VSAM during execution of a GET, PUT, POINT, ENDREQ or ERASE macro. The routine determines what error has occurred by issuing a SHOWCB or TESTCB macro to examine the feedback (FDBK) field in the RPL. The contents of FDBK will be 0000xx, where xx is the error code indicating the type of error.

If the routine cannot correct the error, it should either:

- Close the file, or
- Return to VSAM (which will return to your processing program at the instruction following the last executed instruction).

If you do not have the LERAD exit routine and VSAM encounters a logical error, VSAM returns to your processing program at the instruction following the last executed instruction. Register 15 then contains X'08', and register 1 contains the address of the RPL. Your program can examine the feedback field in the RPL with the SHOWCB or TESTCB macro to identify the logical error.

When VSAM exits to the LERAD routine, the contents of the registers are:

Register	Contents
0	Unpredictable.
1	Address of the RPL that contains the feedback field the routine should examine. The register must contain this address if you return to VSAM.
2-13	Same as when the request macro was issued. Register 13, by convention, contains the address of your processing program's 72-byte save area, which may not be used as a save area by the LERAD routine if the routine returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the LERAD routine. The register does not contain the logical-error indicator.

SYNAD Exit Routine to Analyze Physical Errors: A SYNAD routine can analyze physical I/O errors that were detected by VSAM during execution of a GET, ENDREQ, PUT, POINT, ERASE, or CLOSE macro and that the system error routine was unable to correct. The exit routine determines what error has occurred (reading or writing either the data or the index component) by issuing a SHOWCB or TESTCB macro to examine the feedback (FDBK) field in the RPL. The contents of FDBK will be 0000xx, where xx is the error code indicating the type of error.

If your exit routine cannot correct or bypass the error, it is recommended that the routine (1) issues the PDUMP macro to obtain a dump of the contents of all pertinent control blocks, including the IORB involved in the failing I/O operation; (2) closes the files used by your program; and (3) ends the job. If the error occurred while VSAM was closing the file or index, and if another

error occurs after the exit routine issues a CLOSE macro, VSAM does not exit to the routine a second time.

If the exit routine returns to VSAM, whether the error was corrected or the file closed, VSAM drops the request and returns to your processing program at the instruction following the last executed instruction.

If you do not have this exit routine and VSAM detects a physical error, VSAM returns to your processing program at the instruction following the last executed instruction. Register 15 then contains X'0C', and register 1 contains the address of the RPL. Your program can examine the feedback field in the RPL with the SHOWCB or TESTCB macro to identify the physical error.

An I/O error that occurs when processing a data control interval during the execution of a sequential GET request positions VSAM at the next control interval in key sequence for keyed access or in entry sequence for addressed access. The next GET after the error will return the first record from the control interval following the one with the error. When processing an index control interval, VSAM is not positioned at the next index control interval, and you should not attempt to process further.

Errors that occur while VSAM writes a control interval cause the loss of positioning. When VSAM exits to the SYNAD routine, the contents of the registers are:

Register	Contents
0	Unpredictable.
1	Address of the RPL that contains a feedback return code and the address of a message area, if any. If you issued a request macro, the RPL is the one pointed to by the request macro; if you issued a CLOSE macro, the RPL was built by VSAM to process the close request. Register 1 must contain this address if the SYNAD routine returns to VSAM
2-13	Same as when the request macro or CLOSE macro was issued. Register 13, by convention, contains the address of your processing program's 72-byte save area, which may not be used by the SYNAD routine if it returns control to VSAM.
14	Return address to VSAM.
15	Entry address to the SYNAD routine. The register does not contain the physical-error indicator.

Connecting and Disconnecting a Processing Program and a File

The information you have specified in the ACB and EXLST macros must be connected with the file to be processed so that you can gain access to the data. To this purpose, you must supply, in the job stream, job control statements defining the file and issue, in your program, an OPEN macro for the ACB you have set up for the file.

After your last request for access to the file, you will normally issue a CLOSE macro to complete processing of that file and disconnect your program from the file. If you have not issued a CLOSE macro before end-of-job or if your job terminates abnormally, your file might not be closed properly and subsequent processing of that file might cause errors.

Because it is essential for the integrity of a file that it is closed properly, VSE automatically attempts to close all open VSAM files in the partition at both normal and abnormal termination of a job step. If any control blocks for a file have been destroyed through an error in your program, this file cannot be closed and a message is given to the operator. EXLST routines are not entered during an automatic CLOSE.

The TCLOSE macro performs the functions of CLOSE, except that it leaves the program and the file connected so that you can continue processing without reopening the file.

OPEN Macro (Connect Program and Data)

The OPEN macro calls the open routine, which verifies that the processing program has authority to process the file. Open constructs VSAM control blocks and establishes linkages to those VSAM routines that are needed to process your file(s).

By examining the DLBL statement indicated by the DDNAME operand in the ACB macro and the volume information in the catalog, the open routine verifies that the necessary volumes have been mounted. If a key-sequenced file is being opened, VSAM issues an error code to warn you if the data has been updated separately from its index.

The format of the OPEN macro is:

```
[name]  OPEN   address[,address...]
```

`name`
one through eight characters that provide a symbolic address for the OPEN macro.

`address`
specifies the address of the ACB or DTF for the file(s) to be opened. You can specify the address in register notation (using a register from 2 through 12, in parentheses) or specify it with an expression that generates a valid relocatable A-type address constant.

You can specify up to 16 addresses of ACBs and DTFs that define the files to be opened.

A return code is set in register 15 to indicate whether the ACBs were opened successfully. ACBs should be coded together to ensure that the return code will apply to all of them. If, for example, you coded:

```
OPEN ACB1,DTF1,ACB2
```

the return code will apply to ACB2 only. If ACB2 opened successfully and ACB1 did not, the return code will still be X'00'. (The VSAM Open routine sets register 15 to zero when it receives control after a DTF has been opened.) To ensure that the return code is valid and applies to both ACBs, the macro should be coded in the following way:

```
OPEN DTF1,ACB1,ACB2
```

The VSAM OPEN sets one of the following return codes in register 15:

Return Code	Meaning
X'00'	All ACBs were opened successfully.
X'04'	All ACBs opened successfully, but one or more ACBs had a warning message.
X'08'	One or more ACBs were not opened successfully. The entries with errors are restored to their pre-open status.

If register 15 contains X'04', an error code is set in one or more ACBs to indicate a warning message. All ACBs are open and, unless you prevent it, processing will continue on the file that the message applies to. You can use the ERROR keyword of the SHOWCB or TESTCB macro to examine the code.

If register 15 contains X'08', an error code is set in one or more ACBs. Again, you can use the ERROR keyword of the SHOWCB or TESTCB macro to examine the code. Note that register 15 contains the maximum (worst) return code encountered while opening a list of ACBs. This means that some of the ACBs in the list may have been opened successfully, even though register 15 contains X'04' or X'08'.

For an explanation of the VSAM OPEN error codes, see *VSE/VSAM Messages and Codes.*

CLOSE Macro (Disconnect Program and Data)

The Close routine completes any I/O operations that are outstanding when a processing program issues a CLOSE macro for a file. It writes any output buffers that have not been stored.

Close updates the catalog entries of the file, including pointers to the end of the file and statistics on file processing (such as number of records inserted). If the file was being loaded and the SPEED option was specified (in the DEFINE command), the close routine formats the last control area in the file to ensure that the entire file is accessible.

Close restores the ACB to the status that it had before the file was opened and frees the virtual storage that the open routine used to construct VSAM control blocks.

You must specify a CLOSE macro to change from loading a file to retrieving records from that file in the same run.

Verifying End-of-File

If a job fails when processing a key-sequenced file, for example because of a power failure, and no CLOSE or automatic CLOSE has been executed for that file, you can issue the Access Method Services VERIFY command to have the catalog information for this file corrected. See "Verifying a File's Accessibility" for information on the VERIFY command.

The format of the CLOSE macro is:

```
[name]  CLOSE      address[,address...]
```

`name`
one through eight characters that provide a symbolic address for the CLOSE macro.

`address`
specifies up to 16 addresses of ACBs and DTFs that define the file(s) to be closed. You can specify address in register notation (using a register from 2 through 12 – in parentheses) or specify it with an expression that generates a valid relocatable A-type address constant.

A return code is set in register 15 to indicate whether the ACBs were closed successfully. ACBs should be coded together (following the DTFs) to ensure that the return code will apply to all of them. If, for example, you coded:

```
CLOSE ACB1,DTF1,ACB2
```

the return code will apply to ACB2 only. If ACB2 closed successfully and ACB1 did not, the return code will still be X'00'. (The VSAM Close routine sets register 15 to zero when it receives control after a DTF has been closed.) To ensure that the return code is valid and applies to both ACBs, write the macro in the following way:

```
CLOSE DTF1,ACB1,ACB2
```

The VSAM close routine sets one of the following return codes in register 15:

Return Code	Meaning
X'00'	All ACBs were closed successfully.
X'04'	One or more ACBs were not closed successfully.
X'08'	One or more Close routines could not be loaded because there was not enough virtual storage space, or the modules could not be found. Processing cannot continue.

If register 15 contains X'04', an error code is set in one or more ACBs. You can use the ERROR keyword of the SHOWCB or TESTCB macro to examine the error code. For an explanation of the VSAM CLOSE (and TCLOSE) error codes, see *VSE/VSAM Messages and Codes.*

TCLOSE Macro (Temporary Close)

A TCLOSE macro completes outstanding I/O operations and updates the catalog. Processing can continue without reopening the file. You use the TCLOSE macro to protect data while the file is being loaded or extended and the SPEED option was specified when the file was defined. When TCLOSE is issued, the close routine formats the last control area in the file to ensure that all of the data that has been loaded is accessible.

The TCLOSE macro cannot be used to change the processing mode for a file from sequential load to retrieve in the same run.

The format of the TCLOSE macro is:

```
[name] TCLOSE     address[,address...]
```

`name`
one through eight characters that provide a symbolic address for the TCLOSE macro.

`address`
specifies up to 16 addresses of ACBs. You can specify address in register notation (using a register from 2 through 12 – in parentheses) or specify it with an expression that generates a valid relocatable A-type address constant. You cannot specify the address of DTFs with TCLOSE.

The return codes and error codes are identical to those of the CLOSE macro.

The TCLOSE macro has no effect when the LSR (local shared resources) option is specified in the ACB macro.

OPEN/CLOSE/TCLOSE Message Area

After you have issued an OPEN, CLOSE, or TCLOSE macro, the ACB error code is either zero, indicating that the files were opened or closed successfully, or non-zero, indicating that a warning or error condition occurred. You can examine this code by specifying the ERROR keyword in the SHOWCB or TESTCB macro. However, during an OPEN, CLOSE, or TCLOSE more than one warning or error condition may be detected, in which case the error code which you get when you specify the ERROR keyword reflects only the warning or error condition which occurred last. The error code does not indicate any other (earlier) conditions which might have occurred during the OPEN, CLOSE, or TCLOSE.

In order to save such multiple warning or error conditions, you can provide a message area in which those conditions are to be stored, together with additional information. This message area is connected to the ACB when you specify the following parameters in the ACB macro:

```
MAREA=address
MLEN=length
```

If MAREA or MLEN are not specified or a length of zero has been specified for MLEN, no message area is provided and the ACB error code is then the only indication for errors or warnings which occurred during OPEN, CLOSE, or TCLOSE. If you have specified both MAREA and MLEN (with a non-zero value) and error or warning conditions are detected, appropriate messages are stored into the message area.

The OPEN/CLOSE/TCLOSE message area is also used by VSAM record management if resources such as buffers and control blocks are shared among files. If a GET request is issued for a file using the common resource pool, it can happen that (owing to deferred write operations for PUT requests) the resource pool is filled up with modified buffers forcing VSAM to write a buffer for another file before it can satisfy the GET request. If an error occurs in writing out such a buffer, this is indicated in the message area, together with the ACB name of the affected file.

The message information provided by VSAM consists of the message area header and the message list.

The message area header contains statistical, pointer, and general information. Its contents are unrelated to the individual messages. The format of the message area header is:

Byte	Meaning
0	Flag byte Bit 0=1: At least one warning or error condition has occurred and the complete header is stored. Bit 0=0: Either no warning or error condition has occurred or the message area is too short for the complete header. No further header information and no messages are stored. Bits 1-7 Reserved (set to binary zero)
1-2	Length of message area header
3	Request type code: X'01' OPEN X'02' CLOSE X'03' TCLOSE X'04' GET (for shared resources only)
4-11	Filename used for ACB
12-13	Total number of error or warning conditions issued by OPEN, CLOSE, TCLOSE, or (for shared resources only) by record management
14-15	Number of messages stored into message area
16-19	Address of message list, that is, of first message in the message area

Apart from the flag byte, message area header information is stored only when a warning or error condition was detected (the ACB or RPL error code is non-zero) and the length of the message area (MLEN) is large enough to accommodate the full message area header. Thus, before accessing bytes 1-19 of the message header information, you should test byte 0 to see whether further information was stored at all.

The message list contains the individual messages corresponding to the warning or error conditions detected. It is pointed to by bytes 16-19 of the message area header. Within the message list, the individual messages are stored continuously one after another in the form of variable-length records. The number of messages stored is contained in the message area header (bytes 14-15). Before investigating the message list, you should check whether the stored-message count is zero or greater than zero.

The format of a message is as follows:

Byte	Contents
0-1	Total length of message (including length bytes).
2	ACB error code corresponding to error or warning condition represented by this message, or (for shared resources) RPL error code indicating write error.
3	Function-type code (indicates the component which produced the error or warning condition and the state of the upgrade set): X'00' Condition occurred during accessing the base cluster. Upgrade set is correct. X'01' Condition occurred during accessing the base cluster. Upgrade set may be incorrect as a consequence of this request. X'02' Condition occurred during accessing the AIX over a base cluster. Upgrade set is correct. X'03' Condition occurred during accessing the AIX over a base cluster. Upgrade set may be incorrect as a consequence of this request. X'04' Condition occurred during upgrade processing. Upgrade set is correct. X'05' Condition occurred during upgrade processing. Upgrade set may be incorrect as a consequence of this request. X'06' Condition occurred during writing of a buffer which does not belong to the file for which the GET request was issued (for shared resources only).
4-nn	File-ID of the component indicated by the function-type code (up to 44 bytes), or (for shared resources) name of the ACB identifying the file for which the buffer write error occurred.

Exceptional Conditions

1. Message area (MLEN) is too small to contain the complete message area header:

   Byte 0, bit 0=0 (whole message area is overwritten with binary zeros)

2. Message area is too small to contain a complete message:

   Byte 0, bit 0=1 (header exists)
   Bytes 12-13≠0 (warning or error conditions have occurred)
   Bytes 14-15=0 (no message stored)

3. Message area is too small to contain all messages:

   Byte 0, bit 0=1 (header exists)
   Bytes 12-13≠0 (warning or error conditions have occurred)
   Bytes 14-15≠0, but less than bytes 12-13 (not all messages are stored)

Defining Requests for Access to Data

After you have connected your program to the file, you must **define** your requests for access to the file before you can actually issue a request.

RPL Macro (Creating a Request Parameter List)

You define a request with the RPL macro, which produces a 'request parameter list' (RPL). Each request macro (GET, PUT, POINT, ERASE, and ENDREQ) has one and only one operand, the address of the request parameter list that defines the request. Thus, the information a request macro needs to access a record in a file (such as the ACB of the file to which the request is directed, or the search argument for the record) is always in the RPL instead of in the request macro itself.

The RPL does not indicate a specific request, such as GET or PUT, for example; you can use a single RPL, without modification, for several requests. However, if you want to use the same RPL for different types of processing (for both direct and sequential processing, for example), you must modify the RPL (with the MODCB macro) each time you change from one type of processing to another.

As was pointed out in the discussion of the STRNO operand of the ACB macro, several requests, with the corresponding RPLs pointing to the same ACB, can be active at the same time. You may specify any number of RPLs for requests requiring concurrent positioning, provided you do not exceed the maximum number of concurrent active requests you have specified in the STRNO operand. The requests can be for sequential or direct retrieval or both, and they can be for records in the same part of the file or in different parts.

Values for RPL-macro operands can be specified as absolute numeric expressions, character strings, codes, and expressions that generate valid relocatable A-type address constants. Ordinary register notation with registers 2 through 12 can also be used for addresses.

The format of the RPL macro is:

```
name    RPL     [ACB=address]
                [,AM=VSAM]
                [,AREA=address]
                [,AREALEN=number]
                [,ARG=address]
                [,KEYLEN=number]
                [,NXTRPL=address]
                [,OPTCD=([ADR|CNV|KEY]
                        [,DIR|SEQ|SKP]
                        [ARD|LRD]
                        [,FWD|BWD]
                        [,NSP|NUP|UPD
                        [,KEQ|KGE]
                        [,FKS|GEN]
                        [,LOC|MVE])]
                [,RECLEN=number]
                [,TRANSID=number]
```

`name`

one through eight characters that provide a symbolic address for the request parameter list that is generated. You can use it in the request macros to give the address of the list. You can also use it in the NXTRPL

operand of the RPL macro, when you are chaining request parameter lists, to indicate the address of the next list.

`ACB=address`

specifies the address of the access method control block that identifies the file to which access will be requested. If you used the ACB macro to generate the control block, you can specify the label of that macro for the address. If you omit this operand, you must issue a MODCB macro to specify the address of the file's ACB before you can issue a request against the RPL.

`AM=VSAM`

specifies that this is a VSAM control block. You may want to specify this operand for documentation purposes if your installation also uses VTAM.

`AREA=address`

specifies the address of your I/O work area to and from which VSAM moves the record (OPTCD=MVE) for GET and PUT requests. You process the record in this work area. If you process the records in VSAM's I/O buffer (OPTCD=LOC), VSAM puts into this work area the address of the record in the I/O buffer (GET only).

If you omit this operand , you must issue a MODCB macro to specify the address of your work area before you can issue a request against the RPL.

When you specify user buffers (MACRF=UBF in the ACB) for control-interval (CNV) access, AREA specifies the address of a single I/O buffer. VSAM uses the buffer to read and write control intervals.

`AREALEN=number`

specifies the length, in bytes, of the work area. For OPTCD=MVE, the work area must be large enough to contain the largest record retrieved. For OPTCD=LOC, the work area must be at least 4 bytes long to contain the address of the record in the I/O buffer. For OPTCD=CNV, the work area must be at least the size of a control interval.

If you omit this operand, you must issue a MODCB macro to specify the length of your work area before you can issue a request against the RPL.

`ARG=address`

specifies the address of a field that contains the search argument for:

- Direct or skip sequential retrieval (GET)
- Sequential positioning (POINT)
- Direct or skip sequential storage (PUT) for a relative-record file

For keyed access (OPTCD=KEY), the search argument may be a

- Full key (OPTCD=FKS)
- Generic key (OPTCD=GEN) – in this case you must also indicate its size in the KEYLEN operand
- Relative-record number (which is treated as a key)

For addressed access (OPTCD=ADR), the search argument is always an RBA (relative byte address). To determine the RBA of a record to which you have gained access sequentially or directly by key, you can use the SHOWCB macro to display the RBA of the last record processed. (See "The SHOWCB Macro", later in this section.)

For control-interval access with user buffering and user-supplied RBA, the record is written only to this RBA if positioning is not established by a previous request.

When records are inserted (sequentially or directly) into a key-sequenced file, VSAM obtains the key from the record itself. When records are inserted sequentially into a relative-record file, VSAM returns the assigned relative-record number in the ARG field (as a four-byte binary number).

`KEYLEN=number`

When a generic key is used as a search argument (OPTCD=GEN), this operand specifies the length of the generic key in number of bytes. KEYLEN can be any value from 1 to 255.

If, for example, the full key is 50 bytes long and KEYLEN=10 is specified, VSAM uses the leftmost 10 bytes of the 50-byte key field for comparison with the search argument. The length of the full key is in the catalog. It can be obtained through the KEYLEN parameter of the SHOWCB macro. You place the key (full or generic) in a field pointed to by the ARG parameter.

`NXTRPL=address`

indicates the address of the next RPL in a chain of RPLs; it is required when you chain several RPLs together.

The standard request for access to a file retrieves, stores, or deletes a single record by means of one RPL specified in the request macro. If you want to retrieve or store more than one record with a single GET or PUT, you can chain several RPLs together so that each RPL indicates a different data record. For example, each RPL in the chain could contain a unique search argument and point to a unique work area. For a GET against such a chain of RPLs, VSAM retrieves a record for each RPL in the chain.

The positioning information, normally maintained for each RPL, is maintained only once for the total chain, so a chain of RPLs is processed as a single request. (Chaining RPLs is not the same as processing concurrent requests, where each request requires that VSAM keep track of a position in the file. See the discussion of the STRNO operand under"The ACB Macro."

Figure 5-2 shows how to build a chain of RPLs by specifying the NXTRPL operand. When you issue a request that is defined by a chain of RPLs, specify in the request macro the address of the first RPL in the chain. This request macro determines the request type for the whole chain, and the same major operation, GET for example, is performed for all RPLs in the chain. However, other options such as the request options, which you specify in the OPTCD operand of the RPL macro, may vary from one RPL to another. Thus, an RPL with the option SEQ may be followed by an RPL with the option DIR.

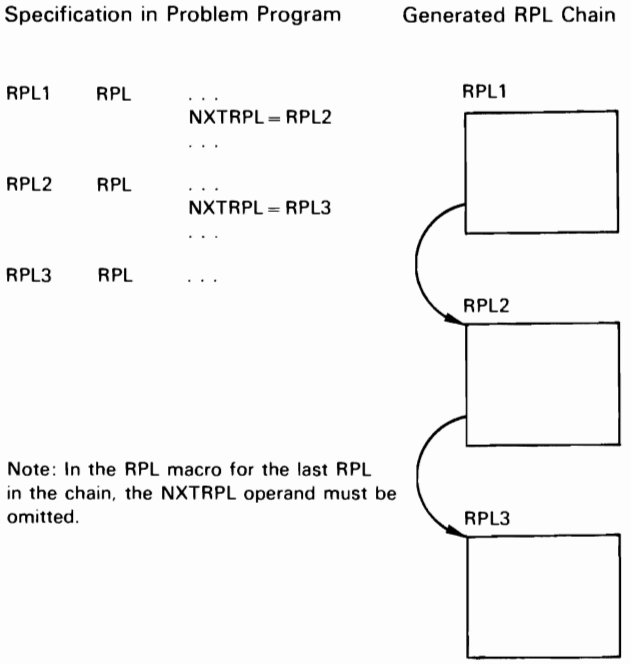


Figure 5-2. Example of an RPL Chain Built by Specifying the NXTRPL Operand

Except for the last RPL in a chain of RPLs, you cannot update or delete records, only retrieve records or store new records. Also you cannot process records in VSAM's I/O buffer with chained RPLs. So OPTCD=UPD and OPTCD=LOC are invalid for chained RPLs (except for the last RPL in a chain).

With chained RPLs, the following types of requests cause VSAM to position itself at the record following the one identified by the last RPL in the chain:

- POINT
- Sequential or skip sequential GET
- Direct GET with positioning requested (OPTCD=NSP)

```
OPTCD=([ADR|CNV|KEY]
       [,DIR|SEQ|SKP]
       [,ARD|LRD]
       [,FWD|BWD]
       [,NSP|NUP|UPD]
       [,KEQ|KGE]
       [,FKS|GEN]
       [,LOC|MVE])
```

specifies the type of access to be gained to the file through the requests defined by this RPL. Options are arranged in groups, and each group has a default value. You can specify only one option in each group; therefore, if your ACB indicates both sequential and direct processing for example, you must modify the RPL when you switch from one to the other. In other words you can use the same RPL for different types of request (GET, PUT, POINT, for example) by modifying the RPL. Because VSAM ignores inapplicable option groups, there is no need for you to zero out

options that are not required before you go from one type of request to another. For more about modifying an RPL, see "MODCB macro." The following table gives the options; they are arranged in groups, and each group has a default value (indicated by underlining):

KEY Keyed access (for key-sequenced and relative-record files). For a key-sequenced file, you can change from keyed to addressed access at any time without positioning. If you change from keyed to control-interval access, the results are unpredictable and no error code will be issued.

ADR Addressed access (for key-sequenced and entry-sequenced files, not for relative-record files). If you change from addressed to keyed access, you must reestablish positioning or the request will terminate with an error. If you change from addressed to control-interval access, the results are unpredictable and no error code will be issued.

CNV Control-interval access (provided for special applications such as utilities). If you change from control-interval to keyed access, you must reestablish positioning or the request will terminate with an error. If you change from control-interval to addressed access, the results are unpredictable and no error code will be issued.

SEQ Sequential processing.

DIR Direct processing.

SKP Skip sequential processing (for keyed access only).

FWD Forward processing of a file.

BWD Backward processing of a file (see "Specifying Processing Options for a Request," later in this section). Backward processing is only allowed for keyed (KEY) or addressed (ADR) access and for sequential (SEQ) or direct (DIR) processing.

ARD The search argument given in your argument (ARG) field determines the record to be located, retrieved, or stored.

LRD The last record of the file is to be located (POINT) or retrieved (GET direct). LRD can only be used in conjunction with OPTCD=BWD.

NUP Request is not for update (you will not update or delete a record you are retrieving; a record you are storing is new). For a direct request, positioning will be released.

NSP For direct processing only, request is not for update, and VSAM will be positioned at the next record for subsequent sequential processing.

UPD Request is for update; you must issue a GET for update before you can issue a PUT for update or an ERASE. However, if you supply your own buffers for control-interval access, you can issue a PUT for update without a preceding GET.

KEQ The search argument must equal the key of the data record (for keyed direct or skip sequential retrieval or keyed sequential pointing).

KGE If the search argument does not equal the key of a record the request applies to the record with the next greater key (for keyed direct or skip sequential retrieval or keyed sequential pointing). If the search argument is a relative-record number, KEQ and KGE apply to a POINT request only. KGE is ignored if BWD is specified.

FKS The entire key is to be used for a search argument (for keyed direct or skip sequential retrieval or keyed sequential pointing).

GEN A generic key is to be used for a search argument (for keyed direct or skip sequential retrieval or keyed sequential pointing). You must specify the length of the generic key in the KEYLEN operand. GEN is ignored for relative-record files and if BWD is specified.

MVE For retrieval and storage, VSAM moves a data record between the I/O buffer and your work area. MVE must also be specified when you supply your own buffers for control-interval access.

LOC For retrieval, you can process the record in VSAM's I/O buffer. VSAM will pass you a pointer to the record in the buffer. If you want to update the record, you will have to move it to your work area before issuing a PUT macro (OPTCD=MVE).

The chapter "Specifying Processing Options for a Request" describes in more detail the options you can specify in the OPTCD operand of the RPL macro.

`RECLEN=number`
specifies the length, in bytes, of a data record stored by a PUT request. For fixed length records, the length need only be set once. For GET requests, VSAM indicates the length of the record in this field. To process a file with records of different lengths you can examine the field with the SHOWCB or TESTCB macro and modify it with the MODCB macro.

`TRANSID=number`
specifies a number that relates modified buffers in a buffer pool for a subsequent write operation (with the WRTBFR macro). It is used in shared resource applications and is described under "Sharing Resources Among Files."

Specifying Processing Options for a Request

This section mainly deals with keyed and addressed access as applied to the different types of processing (sequential, skip sequential, direct) and types of files. Control-interval access and move/locate mode are described at the end of this section. See also "Examples of Using the Request Macros."

Keyed and Addressed Access

You can gain access to a record in a key-sequenced file by keyed or addressed access, in a relative-record file only by keyed access, and in an entry-sequenced file only by addressed access.

An alternate index or a path is treated like a key-sequenced file, except that addressed access is not allowed for an alternate-index path; all key references in the RPL are to the alternate key (instead of the base cluster's prime key).

You can process spanned records in a key-sequenced file by keyed (direct or sequential) access, and in an entry-sequenced file by addressed (direct or sequential) access. In either case, the entire record is returned. You cannot process spanned records in a key-sequenced file by addressed access, because the control intervals that contain the spanned record may not be physically contiguous.

You may process a file in backward direction by keyed or addressed access.

Figure 5-3 summarizes the use of keyed and addressed access to retrieve, add (insert), update, or erase records in key-sequenced, entry-sequenced, and relative-record files. Sequential BWD means that the previous, instead of the next record in sequence (FWD) is to be accessed (see the BWD option of the OPTCD operand). Direct backward (BWD) is mainly used to prepare for a following GET sequential backward.

Sequential and Direct Processing

VSAM allows both sequential and direct processing for each of its three types of files.

Sequential processing of a record depends on the position, with respect to the key, relative-record number, or address, of the previously processed record; direct processing does not. With sequential access, records retrieved by key are in key sequence, records retrieved by relative-record number are in numerical order, and records retrieved by address are in entry (RBA) sequence. To retrieve or store records sequentially after initial positioning, you

Type of File	Type of Access	Type of Processing	Retrieve Records	Add Records	Update Records	Delete Records
key sequenced	keyed	sequential FWD	yes	yes	yes	yes
		sequential BWD	yes	no	yes	yes
		skip sequential	yes	yes	yes	yes
		direct(FWD or BWD)	yes	yes[2]	yes	yes
	addr	sequential FWD	yes	no	yes[1]	yes
		sequential BWD	yes	no	yes[1]	yes
		direct(FWD or BWD)	yes	no	yes[1]	yes
entry sequenced	addr	sequential FWD	yes	to end	yes[1]	no
		sequential BWD	yes	no	yes[1]	no
		direct(FWD or BWD)	yes	to end[2]	yes[1]	no
relative record	keyed	sequential FWD	yes	yes	yes[1]	yes
		sequential BWD	yes	no	yes[1]	yes
		skip sequential	yes	yes	yes[1]	yes
		direct(FWD or BWD)	yes	yes[2]	yes[1]	yes

[1] The length of the records cannot be changed.
[2] 'no' for backward (BWD) processing.

Figure 5-3. Summary of Processing Options

do not need to specify a key, relative-record number, or RBA. VSAM automatically retrieves or stores the next record in order. Apart from OPEN's positioning to the first record of a file, initial positioning can be established by (1) pointing to the desired record, or (2) inserting a record into the file (keyed access with FWD only), or (3) using direct processing and (a) retrieving a record for update (UPD) or (b) specifying OPTCD=NSP.

A variation of normal sequential retrieval is sequential backward processing. Instead of retrieving the next record in relation to current positioning in the file, the previous record is retrieved. Sequential backward processing is available for keyed and addressed access.

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

Keyed Access

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

Keyed access provides for retrieval, update (including lengthening or shortening a record in a key-sequenced file, as well as altering its contents, except for the key), insertion, addition, and deletion. Each of these actions can be sequential, skip sequential, or direct:

With sequential processing, records are retrieved or stored in ascending key or relative-record sequence, starting from the beginning of the file or another position that you select. You do not have to supply a search argument for VSAM to process the records.

When you specify SEQ and BWD in the OPTCD operand of the RPL macro, VSAM returns the previous, instead of the next record in the file (in relation to current positioning). The previous record is the one which has the next

lower key (or relative-record number). With the SEQ and BWD options, you can retrieve, update, or erase records, but you cannot insert or add records.

With direct processing, records are retrieved by the search argument (key or relative-record number) you supply. Records can be processed in any order, without regard to the sequence of records processed before or after.

With skip sequential processing, records are retrieved by search argument, but in **ascending** key or relative-record sequence (no backward processing). Thus, skip sequential combines features of both sequential and direct processing.

The subject is discussed below in more detail for keyed retrieval, storage, and deletion.

Sequential (SEQ) Retrieval

If you specify KEY and SEQ for a **key-sequenced file**, the record to be retrieved depends on where VSAM is positioned in the file. When your program opens the file, VSAM is positioned at the first record in the file to begin sequential processing. However, if sequential processing is not to begin with the first record of the file, you can issue a POINT macro to position VSAM at the record whose key you specify. (If the specified key is generic, that is, a leading portion of the key field, then VSAM is positioned to the first of the records that have the same generic key.) A subsequent GET macro retrieves the record VSAM is positioned at and, at the same time, positions VSAM at the record with the next higher key. In the POINT macro you can also indicate the direction in which the file is to be processed subsequently, by specifying either FWD or BWD.

When you are accessing a base cluster through a **path,** records from the base cluster are returned according to ascending or, if you are retrieving the previous record, descending alternate key values. If several records contain the same (non-unique) alternate key, these records are retrieved in the order in which they were entered into the alternate index (even if BWD was specified). In addition, although register 15 contains X'00', a warning code (duplicate key) is set in the FDBK field of the RPL if there is at least one more data record with the same alternate key value. For example, if there are three data records with the alternate key "1234", the error code would be set during the retrieval of records one and two, and would be reset during retrieval of the third record.

Besides the error code, a function code is set in the RPL indicating whether the condition occurred during accessing the alternate index or the base cluster of a path or during upgrade processing (for a description of the function code, see "Return Codes for the Request Macros", later in this section).

If a base cluster is accessed in a partition, once using a path and once not using a path, a no record found or duplicate key error can occur. These errors can be avoided by using Local Shared Resources.

The example in Figure 5-4 illustrates backward sequential retrieval through a path with non-unique alternate keys.

Alternate index	Pointer	Record
Control Interval 1: Alternate Key 10	1 2 3	T U E
Control Interval 2: Alternate Key 20	1 2 3 4	S D A Y
Backward sequential retrieval results in the following sequence: S,D,A,Y,T,U,E		

Figure 5-4. Example of Backward Sequential Retrieval through a Path with Non-unique Alternate Keys

Keyed sequential retrieval for a **relative-record file** causes the records to be returned in ascending or, if you are retrieving the previous record, descending numerical order, based on the positioning for the file. Positioning is established in the same way as for a key-sequenced file, the relative-record number always being treated as a full 4-byte key. If one or more empty slots are encountered during sequential retrieval, they are skipped and the next (or previous) record is retrieved. The relative-record number of the retrieved record is returned in the ARG field of the RPL.

Sequential Backward (SEQ BWD) Retrieval

In order to process a file in backward direction or to switch from forward to backward processing or vice versa, you must position VSAM and, at the same time, indicate the direction of subsequent processing. Open always establishes forward processing direction so that a GET sequential backward immediately after Open results in a positioning error.

To position VSAM to the end of the file, issue a POINT macro with OPTCD=(BWD, LRD) specified in the RPL. A subsequent GET sequential backward retrieves the last record of the file. To locate and retrieve any other record in the file and establish backward processing direction at the same time, issue a POINT with OPTCD=(BWD,ARD) and a subsequent GET sequential backward (or a direct GET with OPTCD=(BWD,NSP)).

A read error during a GET with:

OPTCD=(SEQ,BWD)

does not cause the positioning to be lost. An immediately following GET with OPTCD=(SEQ,BWD) will cause VSAM to skip the next logical record in backward direction that can be retrieved without a read error.

Direct (DIR) Retrieval

Keyed direct retrieval for a **key-sequenced file** does not depend on previous positioning; VSAM searches the index from the highest level down to the sequence set to retrieve a record. You must specify the record to be retrieved by supplying, in the ARG field of the RPL, one of the following:

- the exact key of the record (OPTCD=KEQ)
- a key less than or equal to the key field of the record (OPTCD=KGE)
- a leading portion of the key, or generic key (OPTCD=GEN)

You can specify OPTCD=KGE when you do not know the exact key. If a record actually has the specified key, VSAM retrieves it; otherwise, it retrieves the record with the next higher key. Generic-key specification for

direct processing causes VSAM to retrieve the first record with a key whose leading portion is identical with the key in the ARG field. If you want to retrieve all the records with the generic key, specify NSP for your direct request, which causes VSAM to position itself at the next record in key sequence. You can then retrieve the remaining records with the same generic key sequentially.

To retrieve a record in the file and indicate backward processing direction for a subsequent GET sequential backward, issue a direct GET with OPTCD=(BWD,NSP,ARD), or LRD instead of ARD if you want to retrieve the last record in the file. The search argument must always be a full key (FKS) and must be the same as that of the data record (KEQ); KGE and GEN are ignored. A direct GET or a POINT with OPTCD=(BWD,LRD) against an empty file results in a no-record-found condition.

When you are accessing a base cluster through a **path** with direct access, a record from the base cluster is returned according to the alternate key value you have specified in the ARG field of the RPL macro. If the alternate key is not unique, the record which was first entered with that alternate key is returned and a warning code (duplicate key) is set in the FDBK field of the RPL. To retrieve the remaining records with the same alternate key, specify the NSP option when retrieving the first record and then change to sequential processing.

When you are processing a **relative-record file** with direct access, you must supply the 4-byte relative record number of the desired record in the ARG field of the RPL macro. If you request a deleted or non-existent record, the request will result in a no-record-found condition.

Skip Sequential (SKP) Retrieval

For skip sequential retrieval for a **key-sequenced file**, when you indicate the key of the next record to be retrieved, VSAM skips to its index entry by using horizontal pointers in the sequence set to get to the appropriate sequence-set index record to scan its entries. SKP is similar to direct processing, except that the key of the next record must always be higher in sequence than the key of the preceding record.

A **relative-record file** has no index. When you indicate the number of the next record to be retrieved, VSAM calculates the control interval containing the requested record and the position of the requested record within that control interval. As for a key-sequenced file, the relative-record numbers you specify must be ascending sequence for skip sequential retrieval.

For a **path**, skip sequential access is the same as direct access, except that the alternate key values have to be in ascending sequence.

Backward processing is not allowed for skip sequential retrieval.

Keyed Insertion

VSAM stores a record whenever you issue a PUT request against an RPL. A PUT request for update following a GET for update stores the record that the GET retrieved. To update a record, you must previously have retrieved it for update.

When you store records sequentially beyond the highest key in the file, VSAM automatically extends the file as though you were continuing to load records. VSAM does not use distributed free space for these records, but establishes new control areas at the end of the file. Free space is left in the new control areas and control intervals according to the file's FREESPACE specification in the catalog.

To store records in key (or relative-record) sequence throughout the file, you can use sequential, skip sequential, or direct access.

When you insert records into a key-sequenced file, you never have to specify a search argument; VSAM always obtains the key from the record itself. With sequential insertion or skip sequential insertion of consecutive records, VSAM creates new control intervals and control areas and free space is left in them according to the file's FREESPACE specification in the catalog. With direct insertion or skip sequential insertion of non-consecutive records, VSAM uses the free space.

For a **relative-record file**, sequential insertion causes a record to be inserted into the next slot (provided it is empty). The slot number is returned in the ARG field of the RPL. If the slot is not empty, a duplicate-record error condition will occur.

Direct or skip sequential insertion of a record into a relative-record file causes the record to be placed as specified by the relative-record number in the ARG field. You must insert the record into a slot which does not contain a record; otherwise, a duplicate-record error condition will occur.

If you insert a record after the current end-of-file of a relative-record file, the file is preformatted from the current end-of-file up to and including the control area that is to contain the inserted record. Preformatting mainly consists of inserting control information in the control areas and indicating that the slots are empty.

You can update and insert base data records via a **path**, provided the PUT request does not result in non-unique alternate-key values in an alternate index (in the upgrade set) which you have defined with the UNIQUEKEY parameter. The alternate indexes in the upgrade set are modified automatically when you insert or update a data record in the base cluster. When you update a previously retrieved base record via a path, you must not change the alternate key by which that record was retrieved or its prime key. If the updating of the alternate index results in an alternate index record with no pointers to the base cluster, that alternate index record is erased.

PUT insert requests with OPTCD=NUP or NSP are not allowed in backward direction.

Keyed Deletion

An ERASE macro instruction following a GET for update deletes the record that the GET retrieved. A record is physically erased in the file when you delete it. The space the record occupied is then available as free space.

You can erase a record from the base cluster of a **path** only if the base cluster is a key-sequenced file. The alternate indexes of the upgrade set are modified automatically when you erase a record. If the alternate key value of the erased record is unique, the alternate index data record with that alternate key is also deleted.

You can erase a record from a **relative-record file** after you have retrieved it for update. The record will be set to binary zeros and the control information for the slot will be updated to indicate an empty slot. You can reuse the vacated space by inserting another record of the same length in that location.

Addressed Access

Addressed access is the only form of access for an entry-sequenced file, using the RBA determined for a record when it was stored in the file. This form of access is also allowed for a key-sequenced file, but not for a path or for a relative-record file. For both key-sequenced and entry-sequenced files,

addressed access allows processing in backward direction (by specifying OPTCD=BWD in the RPL macro). Positioning is established as for keyed retrieval. You cannot add or insert records in backward direction.

Addressed access can be either sequential or direct for both key-sequenced and entry-sequenced files, but the processing allowed for a key-sequenced file is different from that allowed for an entry-sequenced file.

With a key-sequenced file, addressed access can be used to retrieve records, update their contents, and delete records, but the length of a record and the contents of its key field cannot be changed. Records cannot be added because VSAM does not allow changes to the file which could cause the index to change.

With an entry-sequenced file, addressed access can be used to retrieve records and to update their contents, but not to change their lengths. New records can be added to the end of the file. Records cannot be physically deleted because that would change the entry sequence of the records in the file (the RBAs of the records).

Keyed insertion, deletion, or update (length changing) of records can change the RBAs of these records. Therefore, to use addressed access to process a key-sequenced file, you may have to keep track of RBA changes. For this purpose VSAM passes back the RBA of each record retrieved, added, updated, or deleted. (See also the JRNAD exit discussed earlier in this section.)

Addressed Retrieval

Positioning for addressed sequential retrieval is done by RBA rather than by key. When a processing program opens a file for addressed access, VSAM is positioned at the first record in the file in entry sequence to begin addressed sequential processing. A POINT positions VSAM for sequential access beginning at the record whose RBA you have indicated. A sequential GET causes VSAM to retrieve the data record at which it is positioned and positions VSAM at the next or previous record in entry sequence depending on whether you have specified forward (FWD) or backward (BWD) processing in the RPL. If you use addressed sequential retrieval for a key-sequenced file, records will not be in their key sequence if there have been control-interval or control-area splits.

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

With direct processing, you may optionally specify that GET position VSAM at the next record in forward (FWD,NSP) or backward (BWD,NSP) sequence. Your program can then process the following or preceding records sequentially.

Addressed Deletion

You can use the ERASE macro with a key-sequenced file to delete a record that you have previously retrieved for update.

With an entry-sequenced file, you are responsible for marking a record you want to delete. In other words, as far as VSAM is concerned, the record is not deleted. You can reuse the space occupied by a record marked for deletion by retrieving the record for update and storing in its place a new record of the same length.

Addressed Insertion

VSAM does not insert new records into the middle of an entry-sequenced file, but adds them at the end. With addressed access of a key-sequenced file, VSAM does not insert or add new records. You cannot add or insert new records in backward direction.

When you store records sequentially beyond the highest key in the file, VSAM automatically extends the file as though you were continuing to load records.

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

To change the length of a record in an entry-sequenced file, you must store it either at the end of the file (as a new record) or in the place of a deleted record of the same length (as an update). You are responsible for marking the old version of the record as deleted.

Control-Interval Access

VSAM provides programmers of utilities and systems with control interval access. They retrieve and store the contents of a control interval, rather than a single record, by specifying control interval access in the macros and (for direct processing) giving the RBA of the control interval. They are responsible for maintaining the control information at the end of the control interval. The format of this information may change in future releases of VSAM.

Control-interval access is allowed for relative-record files, provided the size of the file is not changed by insertions or additions.

Control-interval access is not allowed when you process an alternate-index path or access records in backward direction (with the BWD option).

Processing a Record in a Work Area or in a Buffer

When your processing program retrieves a record, VSAM reads into virtual storage the contents of the entire control interval in which the record is stored. VSAM deblocks the records and either places the requested record in your program's work area (OPTCD=MVE) or leaves the record in VSAM's I/O buffer and gives you, in the AREA field, the address of the record in the buffer (OPTCD=LOC). VSAM indicates the length of the record to your program (in the RECLEN field) in both move mode and locate mode. You need not concern yourself with any physical attributes of stored records. Spanned records cannot be accessed in locate mode.

Examples of ACB, EXLST, and RPL Macros

Figure 5-5 shows an example of how you can specify VSAM control blocks by using the ACB, EXLST, and RPL macros. These control blocks are generated during assembly of your program. Default values will be provided for those parameters that are omitted.

Figure 5-6 shows the job control statements needed to open and process a file identified in an ACB macro (file ACBADR in the example).

Note: Continuation characters required in column 72 are not shown in the example.

```
ACBADR      ACB       EXLST=EXITS,
                      PASSWD=PASS,
                      BUFND=4,BUFNI=3,
                      BUFSP=11264,
                      MACRF=(KEY,SEQ,
                      DIR,OUT),
                      STRNO=2
EXITS       EXLST     EODAD=(ENDUP,N),
                      LERAD=LOGERR,
                      SYNAD=(IOERR,L),
                      EXCPAD=(OVERLP,A)
RETRVE      RPL       ACB=ACBADR,
                      AREA=WORK,
                      ARG=SEARCH,
                      AREALEN=125,
                      OPTCD=(DIR,NSP)
              .
              .
              .
PASS        DC        FL1'6',C'CHANGE'
WORK        DS        CL125
SEARCH      DS        CL4
IOERR       DC        C'PHYSICAL'
ENDUP                 End-of-file routine
              .
              .
              .
LOGERR                Logical-error routine
              .
              .
              .
OVERLP                I/O-overlap routine
              .
              .
```

ACB Macro:

Because the DDNAME operand is not specified, VSAM uses the name, ACBADR, of the ACB as the name (filename) of the associated file.

BUFND:	Four I/O buffers for data control intervals.
BUFNI:	Three I/O buffers for index control intervals.
BUFSP:	The size of the buffer space is sufficient to accommodate four data control intervals of 2048 bytes each and three index control intervals of 1024 bytes each.
EXLST:	Specifies that the label of the exit list associated with this ACB is named EXITS.
PASSWD:	Specifies the location of the password. The DC at PASS gives the password's length in the first byte and the password itself in the subsequent six bytes.
MACRF:	Specifies keyed-sequential and keyed-direct processing for both insertion and update.
STRNO:	Specifies that two requests will require concurrent positioning.

EXLST Macro:

EODAD:	The end-of-file routine is located at ENDUP and is not active.
LERAD:	The logical error routine is located at LOGERR and is active.
SYNAD:	The physical I/O error routine's name is located at IOERR.
EXCPAD:	The I/O-overlap routine is located at OVERLP and is active.

RPL Macro:

ACB:	Associates the RPL with the ACB named ACBADR.
AREA:	Address of work area is WORK.
AREALEN:	Length of work area is 125 bytes.
ARG:	The search argument is defined at SEARCH. Because the KEYLEN operand is omitted, VSAM uses the full key as search argument.
OPTCD:	Specifies direct processing with positioning at the next record for subsequent sequential processing.

Figure 5-5. Example of VSAM Macros Used to Specify VSAM Control Blocks for a File

```
// JOB
// DLBL IJSYSCT,'AMASTCAT',,VSAM
// DLBL ACBADR,'FILE1',,VSAM
// EXEC progname,SIZE=AUTO
    .
    .
    .
    OPEN ACBADR
    .
    .
    .
    GET RPL=RETRVE
    .
    .
    .
    CLOSE ACBADR
    .
    .
    .
/*
/&
```

FILE1 is the name of the file under which it is entered in the VSAM master catalog.

Figure 5-6. Example of Job Control Statements Needed to Open and Process a VSAM File

Manipulating the Information Relating the Program and the Data

An Access Method Control Block (ACB), Exit List (EXLST), and a Request Parameter List (RPL) can be dynamically generated, modified, displayed, or tested when your program is executed. You can use the GENCB macro to create a control block or list. You can use the MODCB macro to modify control blocks or lists dynamically, the SHOWCB macro to display selected fields, and the TESTCB macro to test the value of selected fields. The advantage of these macros, called the control block manipulation macros, is that you need not be concerned with changes in the formats or lengths of the ACB, RPL, and EXLST or with the displacements of their fields.

With GENCB and SHOWCB, you can specify an area for VSAM to build the block or list or you can let VSAM supply an area. If you want your program to be reenterable, you should either let VSAM supply the area or you should use a GETVIS macro to supply the area when your program is executed.

Each control block manipulation macro has a parameter list (not the same as an RPL) which is built when the macro is assembled. This parameter list is used by VSAM routines during program execution to generate, modify, display, or test the ACB, EXLST, or RPL. Parameter lists can be built without using the control block manipulation macros, as described in Appendix G. To allow you to make your program reenterable or to share macro parameter lists, the GENCB, MODCB, SHOWCB, and TESTCB macros have four forms:

- The *standard* form builds a parameter list inline along with the instructions produced by the macro. The standard form does not allow reenterable code or shared parameter lists.
- The *list* form builds a parameter list alone; it does not include the instructions produced by the macro. The list form is used together with the execute form to allow you to share parameter lists between macros and, if the list is built outside your code, to make your program reenterable. If the list is built outside your code, you must obtain the area for it during program execution by a GETVIS macro.
- The *execute* form contains the instructions produced by the macro to take action on a parameter list; you can also use it to change the parameter list before VSAM acts on it. The execute form is used together with the list form to allow you to share parameter lists between macros.
- The *generate* form produces both a parameter list and executable instructions. It builds the parameter list in an area you specify. You must obtain the area during program execution by a GETVIS macro.

The following descriptions of each macro show the standard form. The list, execute, and generate forms of the macros are nearly identical to the standard forms. They are discussed together at the end of this section.

GENCB Macro

The GENCB macro generates an ACB, an EXLST, or an RPL when it is executed. You can use it in place of the ACB, EXLST, and RPL macros to avoid (1) reassembling your programs should the format or length of the control block or list change, and (2) generating more than one copy of a control block or list.

GENCB generates the control block(s) or list(s) either in an area you specify or, if you do not specify an area, in an area obtained by VSAM in your partition.

When you issue a GENCB macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue a GENCB macro from within one of your exit routines such as LERAD or SYNAD, your program must provide a second 72-byte save area for use by VSAM, because the original save area is still in use by the external VSAM routine.

The operands of the GENCB macro are specified as absolute numeric expressions, as character strings, as codes, as expressions that generate valid relocatable A-type address constants, in register notation, as S-type address constants, and as indirect S-type address constants. "Appendix F: Operand Notation for VSAM Macros" gives all the ways of coding each operand for the macros that work at execution.

If you use register notation to specify specific addresses in your GENCB macro, be sure that these registers contain the correct addresses before you issue the GENCB macro. This is necessary because the assembler-generated instructions for this macro store the addresses contained in the specified registers in the appropriate control fields.

The standard form of the GENCB macro is:

```
[name]  GENCB   BLK={ACB|EXLST|RPL}
                [,AM=VSAM]
                [,COPIES=number]
                [,keyword=value]
                [,LENGTH=number]
                [,WAREA=address]
```

`name`
one through eight characters that provide a symbolic name for the GENCB macro.

`AM=VSAM`
specifies that this is a VSAM control block. You may want to specify this operand for documentation purposes if your installation also uses VTAM.

`BLK={ACB|EXLST|RPL}`
specifies whether you want to generate an ACB, an EXLST, or an RPL.

`COPIES=number`
specifies the number of control blocks or lists you want VSAM to generate. The default is 1. If you generate two or more, they are generated next to each other. They are identical, so you must use MODCB to tailor them for a particular file or request.

VSAM returns, in register 1, the address of the first (or only) control block and, in register 0, the total length of the control block(s) built. You can find out the length of each control block by dividing the length of the area by the number of copies. The address of each control block can then be calculated by this offset from the address in register 1.

`keyword=value`
The operands you code are identical to those of the ACB, EXLST, and RPL macros, except that you can code them in more ways, as described in Appendix F. If you do not code any operands, VSAM builds:

- For BLK=ACB, an ACB with default values provided by VSAM when you open the file. You must supply the DDNAME=filename operand before the file is opened.

- For BLK=EXLST, a complete EXLST with zeros for addresses and all entries flagged inactive.
- For BLK=RPL, an RPL with default values.

`LENGTH=number`

specifies the length of the area, if any, you provided by the WAREA operand. You can determine the length required for a control block or list by using the SHOWCB macro.

`WAREA=address`

specifies the address of an area in which you want VSAM to generate the control block(s) or list(s). The area must begin on a fullword boundary. If WAREA is specified, the LENGTH operand must also be specified. If you do not specify WAREA, VSAM obtains an area in your processing partition in which to generate the control block(s) or list(s). When control is returned to you, register 1 contains the address of the control block or list and register 0 contains the total length of the control block(s) or list(s).

Examples of the GENCB Macro: Figure 5-7 shows examples of how to specify VSAM control blocks by using the GENCB macro. With GENCB, the control blocks are created dynamically during execution of the program. The same parameters are specified in this example as are specified in the previous example of ACB, EXLST, and RPL macros (see Figure 5-5). VSAM obtains space for each control block in your partition. The address of each control block is set in register 1 after the GENCB is executed.

```
*       GENERATE VSAM CONTROL BLOCKS
        GENCB   BLK=EXLST,EODAD=(ENDUP,N),
                LERAD=LOGERR,
                SYNAD=(IOERR,L)
        LTR     15,15                           GENCB successful?
        BNZ     GENERR                          No, go to error routine
        LR      3,1                             Yes, save EXLST address
        GENCB   BLK=ACB,EXLST=(3),PASSWD=PASS,
                BUFND=4,BUFNI=3,BUFSP=11064,
                MACRF=(KEY,SEQ,DIR,OUT),
                DDNAME=VFILENM
        LTR     15,15                           GENCB successful?
        BNZ     GENERR                          No, go to error routine
        LR      2,1                             Yes, save ACB address
*
*
        GENCB   BLK=RPL,AREA=WORK,
                AREALEN=125,OPTCD=(DIR,NSP),
                ARG=SEARCH,ACB=(2)
        LTR     15,15                           GENCB successful?
        BNZ     GENERR                          No, go to error routine
        LR      4,1                             Yes, save RPL address
*
*       PROCESSING ROUTINES
*
        GET     RPL=(4)
*
*       CONSTANTS AND WORK AREAS
*
PASS    DC      FL1'6',C'CHANGE'
WORK    DS      CL125
SEARCH  DS      CL4
```

Figure 5-7. Examples of Specifying VSAM Control Blocks by Using GENCB Macro. (The continuation characters required in column 72 are not shown.)

MODCB Macro

The MODCB macro modifies the addresses, values, options, and names that you can establish with the ACB, EXLST, RPL, and GENCB macros in an ACB, EXLST, or RPL.

If you want to modify only the length of a data record (the value of the RECLEN field of the corresponding RPL), you can do so without any call to a VSAM routine by issuing the MODCB macro in the following short form:

MODCB RPL=(1),RECLEN=(0)

The address of the RPL must be contained in register 1. The record length, stored in register 0, will be placed into the RPL. No parameter list is created.

The operands of the MODCB macro are specified as absolute numeric expressions, as character strings, as codes, as expressions that generate relocatable A-type address constants, in register notation, as S-type address constants, and as indirect S-type address constants. "Appendix F: Operand Notation for VSAM Macros" gives all the ways of coding each operand for the macros that work at execution.

When you issue a MODCB macro (not the short form described above), register 13 must contain the address of a 72-byte save area, that you are providing. When you issue a MODCB macro from within one of your exit routines such as LERAD or SYNAD, your program must provide a second 72-byte save area for use by VSAM because the original save area is still in use by the external VSAM routine.

The standard form of the MODCB macro is:

```
[name]  MODCB   {ACB|EXLST|RPL}=address
                ,keyword=value
                [,AM=VSAM]
```

`name`
one through eight characters that provide a symbolic name for the MODCB macro.

`AM=VSAM`
specifies that this is a VSAM control block. You may want to specify this operand for documentation purposes if your installation also uses VTAM.

`{ACB|EXLST|RPL}=address`
specifies whether you want to modify an ACB, an EXLST, or an RPL and specifies its address. You cannot modify an open ACB. Neither can you use MODCB to activate or deactivate a JRNAD exit if the ACB to which the EXLST is connected is already open (see the discussion of JRNAD in the EXLST macro). You can modify a field in an EXLST at any time, but you cannot add entries to or delete entries from it. You cannot modify an active RPL: that is, one that defines a request that has been issued but not completed.

With the execute form of MODCB, you can change the address of the block or list to be modified, but not the type.

`keyword=value`
The operands you code are identical to those for the ACB, EXLST, and RPL macros, except that:

- You can code them in more ways, as shown in Appendix F.

- There are no defaults for the options of the ACB MACRF operand or the RPL OPTCD operand. With OPTCD, when you set on a new option with the MODCB macro, the old option is automatically turned off, because you can specify only one option in each of its groups (see "The RPL Macro."
- You can make an address in an EXLST active or not active without specifying the address by coding: keyword=(, {A|N}).
- When you specify an address for an entry in an EXLST that previously contained zeros (possible if you generated a default list with the GENCB macro), you must code keyword=(addr,A) to make the address active, because A is not a default for the MODCB macro.

The MODCB macro cannot be used to reset a MACRF option which was set in an ACB unless this option is mutually exclusive with the new intended option. For example, if the options

KEY,SEQ,OUT

were set and you wish to have the options

ADR,SEQ,OUT

instead, then specifying MACRF=ADR in a MODCB macro results in options

KEY,ADR,SEQ,OUT

being set in the pertinent ACB.

Examples of the MODCB Macro: Figure 5-8 shows two examples of modifying VSAM control blocks by using the MODCB macro.

The MODCB (short form) is used to place the length of a record in the RPL when variable-length records are being added to a file:

```
MODCB   RPL=(1),RECLEN=(0)        Current length in reg. 0
LTR     15,15                     MODCB successful?
BNZ     MODERR                    No, go to error routine
PUT     RPL=(1)                   Yes, write record
```

The MODCB is used to activate the EODAD exit specified in the GENCB example of Figure 5-7:

```
MODCB   EXLST=(3),EODAD=(,A)
LTR     15,15                     MODCB successful?
BNZ     MODERR                    No, go to error routine
```

Figure 5-8. Examples of Modifying VSAM Control Blocks

SHOWCB Macro

The SHOWCB macro displays fields in an ACB, EXLST or RPL. VSAM places these fields in an area that you provide. They are independent of the format of the control block or list you are displaying: the fields are displayed in the order that you specify the keywords for them.

The operands of the SHOWCB macro are specified as absolute numeric expressions, as character strings, as codes, as expressions that generate valid relocatable A-type address constants, in register notation, as S-type address constants, and as indirect S-type address constants. "Appendix F: Operand Notation for VSAM Macros" gives all the ways of coding each operand for the macros that work at execution.

The standard form of the SHOWCB macro is:

```
[name] SHOWCB  [{ACB|EXLST|RPL}=address,]
               [,AM=VSAM]
               AREA=address,
               FIELDS=(keyword[,keyword...]),
               LENGTH=number
               [,OBJECT={DATA,INDEX}]
```

If you want to display only the length of a data record (the RECLEN field of the corresponding RPL), you can do so without any call to a VSAM routine by issuing the SHOWCB macro in the following short form:

SHOWCB RPL=(1),RECLEN=(0)

The address of the RPL must be contained in register 1. The record length will be put into register 0. No parameter list is created.

When you issue a SHOWCB macro (not the short form described above), register 13 must contain the address of a 72-byte save area that you are providing. When you issue a SHOWCB macro from within one of your exit routines such as LERAD or SYNAD, your program must provide a second 72-byte save area for use by VSAM because the original save area is still in use by the external VSAM routine.

`{ACB|EXLST|RPL}=address`

This operand specifies whether you want to display an ACB, an EXLST, or an RPL and specifies its address.

In the standard and list forms of SHOWCB, you can omit this operand if you are displaying only the standard length of a control block or list (see "Length of a Control Block or List" following the discussion of the FIELDS=operand below). With the execute form of SHOWCB, you can change the address of the block or list to be displayed, but not the type.

`AM=VSAM`

specifies that this is a VSAM control block. You may want to specify this operand for documentation purposes if your installation also uses VTAM.

`AREA=address`

specifies the address of the area in virtual storage that you are providing for VSAM to display the items you specify in the FIELDS operand. The items are in the area in the order you specify the keywords. The area must begin on a fullword boundary.

`FIELDS=(keyword[,keyword...])`

There are three groups of keywords you can code for the FIELDS operand of the SHOWCB macro:

- The keywords that you can code with the ACB, EXLST, RPL, and GENCB macros
- The length of an ACB, RPL, or EXLST
- The attributes of an open file or index indicated by the ACB.

Keywords of the ACB, EXLST, and RPL Macros: The keywords in this group require one fullword each for display, except DDNAME which requires two fullwords. The keywords are identical to those of the ACB, EXLST, and RPL macros, except that:

- You can code the operands in more ways, as shown in Appendix F.
- You do not code the address, value, option, or name to which the keyword is equal.
- In relation to the ACB macro, you cannot display the MACRF options, but you can display, with the keyword ERROR, the error code (in the rightmost byte of the display word) from the Open or Close routine (see the OPEN and CLOSE macros); you can test the MACRF options with the TESTCB macro.
- In relation to the EXLST macro, you cannot display the codes that indicate whether an exit address is active or not active or is the address of the name of a routine to be loaded; you can test them with the TESTCB macro.
- In relation to the RPL macro, you cannot display the OPTCD options, but you can code the keyword FDBK to display error codes (in the rightmost byte of the display word) from the request macros and the keyword RBA to display the relative byte address of the last record processed; you can test the OPTCD options with the TESTCB macro.

  You can code the keyword AIXPC to display the number of key or RBA pointers in the most recently processed alternate index record.

  You can code the keyword FTNCD to display, after a logical or physical error, the *function code* which indicates whether the respective condition occurred during processing of the base cluster or the alternate index of a path or during upgrade processing. (For details, see "Return Codes for the Request Macros" under "Requesting Access to a File.")

Length of a Control Block or List: You can code the keyword ACBLEN, EXLLEN, or RPLLEN to display either the standard length of an ACB, EXLST, or RPL, or the actual length of a particular block or list. You display a standard length by omitting the {ACB|EXLST|RPL} operand and coding only one (or more) of these length keywords and no other keywords. You display the actual length of a block or list by specifying the {ACB|EXLST|RPL} operand and the corresponding length keyword.

Attributes of an Open File: After a file is opened, the ACB contains information that it does not contain before it is opened or after it is closed. Whether you are displaying the attributes of the data or the index of a key-sequenced file is determined by the OBJECT operand. Each item displayed requires one fullword in your work area, except STMST which requires two fullwords. You can display the following items:

Operand	Meaning
AVSPAC	Number of bytes of available space in the data or index component.
BUFNO	Number of buffers being used for the data or index component.
CINV	Size of a control interval in the data or index component.
FS	Percent of free control intervals in each data control area of a key-sequenced file.
KEYLEN	Full length of the prime key or alternate key field in each logical record (depending on whether or not you access the base cluster via a path).
LRECL	Maximum length of a logical record, or for an index, the index control interval size minus seven bytes.
NCIS	Number of control-interval splits in the file.
NDELR	Number of data records deleted from the file.
NEXCP	Number of times EXCP was issued by VSAM I/O routines.

Operand	Meaning
NEXT	Number of logical extents, data spaces, or portions of data spaces, now allocated to the data or index component.
NINSR	Number of data records inserted into the file. For a relative-record file, number of valid records (non-empty slots in the file). For a key-sequenced file, number of records inserted between the records, not records initially loaded or added to the end of the file.
NIXL	Number of levels in the index of a key-sequenced file.
NLOGR	Number of data records in the file. For a relative-record file, total number of slots (empty or non-empty) in the used control intervals.
NRETR	Number of data records retrieved from the file.
NSSS	Number of data control-area splits in a key-sequenced file.
NUPDR	Number of data records updated in the file.
RKP	Displacement of the prime key or alternate key field from the beginning of a data record (depending on whether or not you access the base cluster via a path); the same value is displayed whether the object is index or data.
STMST	System timestamp; the time and day (in microseconds) when the data or index component was last closed. Bits 52 through 63 of the field are unused.
STRMAX	Maximum number of requests which were concurrently active since the resource pool was built. Used in shared resource applications (see the BLDVRP macro under "Sharing Resources Among Files").

`LENGTH=number`

specifies the length of the display area you are providing (by way of the AREA operand). Each field in the ACB and RPL takes a fullword, except for DDNAME and STMST in the ACB, which take two fullwords. Each EXLST operand takes only one fullword, because you cannot display the codes A, N, and L.

`OBJECT={DATA | INDEX}`

specifies, for the open ACB of a key-sequenced file, whether the fields being displayed are for the data or the index. VSAM will display the same values for KEYLEN regardless of your specification in the OBJECT operand. The same is true for field RKP.

If you specify INDEX, VSAM's display is all zeros for the following fields:

FS	NINSR	NUPDR
NCIS	NRETR	
NDELR	NSSS	

Examples of the SHOWCB Macro: The following examples show how to display information from VSAM control blocks using the SHOWCB macro. Continuation characters required in column 72 are not shown in the example.

The SHOWCB macro is used to display statistics about an open file:

```
          SHOWCB ACB=(2),AREA=DISPLAY,LENGTH=12,
                 FIELDS=(KEYLEN,LRECL,RKP)
          LTR    15,15                          SHOWCB successful?
          BNZ    SHOWERR                        No, go to error routine
          .
          .
DISPLAY   DS     0F                             Align on fullword boundary
KEYLEN    DS     F
LRECL     DS     F
RKP       DS     F
```

The SHOWCB macro is used to display the length and RBA of a record that has been retrieved:

```
          GET    RPL=(4)
          LTR    15,15
          GNZ    GETRR
          SHOWCB RPL=(4),AREA=DISPLAY,LENGTH=8,
                 FIELDS=(RECLEN,RBA)
          LTR    15,15                          SHOWCB successful?
          BNZ    SHOWERR                        No, go to error routine
          .
          .
DISPLAY   DS     0F                             Align on fullword boundary
RECLEN    DS     F
RBA       DS     F
```

TESTCB Macro

The TESTCB macro tests values in an ACB, EXLST, or RPL against values that you specify in the macro.

You can examine the condition code after issuing a TESTCB macro and examining the return code in register 15. For keywords specified as an option (such as A for an operand of the EXLST macro), a test is for an equal or unequal comparison; for keywords specified as an address or value, a test is for an equal, unequal, high, low, not-high, or not-low comparison. In the comparison, A to B, B is the address, value, or option that you specify in the TESTCB macro. For example, if you test for a value in an ACB, a high comparison means the value in the block is higher than the value you specified in the TESTCB macro.

The standard form of the TESTCB macro is:

```
[name]  TESTCB  [{ACB|EXLST|RPL}=address,]
                [,AM=VSAM]
                [ERET=address,]
                keyword=value
                [OBJECT={DATA|INDEX}]
```

When you issue a TESTCB macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue a TESTCB macro from within one of your exit routines such as LERAD or SYNAD, your program must provide a second 72-byte save area for use by VSAM because the original save area is still in use by the external VSAM routine.

`{ACB|EXLST|RPL}=address`

This operand specifies whether you want to test an ACB, an EXLST, or an RPL and specifies its address.

In the standard and list forms of TESTCB, you can omit this operand if you are testing only the standard length of a control block or list (see "Length of a Control Block or List") following the discussion of the

keyword=value below). With the execute form of TESTCB, you can change the address of the block or list to be tested, but not the type.

`ERET=address`

specifies the address of a user-written routine that VSAM gives control if, because of an error, it is unable to test for the condition you specified (return code in register 15 is not X'00'). When the ERET routine receives control, it should inspect the return code. If the return code is X'04', an error code will be tested in register 0. See "Return Codes from the Control-Block Manipulation Macros" for the error codes that can be tested by TESTCB.

After completing its processing, the ERET routine can terminate the job or pass control to a point in the processing program that it determines. It cannot return to VSAM.

`keyword=value`

specifies a field and a value. The contents of the field are compared with the value and the condition code is set. You can specify only one keyword at a time. There are three groups of operands that you can code with the TESTCB macro:

- The addresses, values, options, and names that you can code with the ACB, EXLST, RPL, and GENCB macros
- The length of a control block or list
- The attributes of an open file or index indicated by the ACB.

If you code more than one operand, each of them must compare equal to the corresponding value in the block or list for you to get an equal condition.

Operands of the ACB, EXLST, and RPL Macros: The operands in this group are identical to those of the ACB, EXLST, and RPL macros, except that:

- You can code the operands in more ways, as shown in Appendix F.
- In relation to the ACB macro, you can test for error codes from the Open and Close routines by coding ERROR=*code* (as any absolute expression, except for a self-defining character term). When an ACB is opened for a path, the base cluster ACB is tested. However, you can test the alternate index ACB by specifying MACRF=AIX in the ACB macro.
- In relation to the EXLST macro, you can test whether an EXLST has an exit of a certain type by coding keyword=0.
- In relation to the EXLST macro, you can test whether an address in an EXLST is active or not active or is the address of the name of a routine to be loaded, without specifying the address by coding: keyword=[,{A|N}][,L].
- In relation to the RPL macro, you can code the operand FDBK=*code* (as any absolute expression, except for a self-defining character term) to test for error codes from the request macros (see "Return Codes for the Request Macros"). You can code the operand RBA=*number* to test the relative byte address of the last record processed.
- In relation to the RPL macro, you can code the operand AIXPC=*number* to find out the number of key or RBA pointers in the most recently processed alternate index record.

You can code the operand AIXFLAG=AIXPKP to test whether the alternate index record just processed contains prime key pointers (or, if not, RBA pointers).

You can code the operand FTNCD=*number* to test, after a logical or physical error, the *function code* which indicates whether the respective condition occurred during processing of the base cluster or the alternate index of a path or during upgrade processing. (For details, see "Return Codes for the Request Macros" under "Requesting Access to a File").

Length of a Control Block or List: You can code the operand EXLLEN=*length,* ACBLEN=*length,* or RPLLEN=*length* to test either the standard length of an EXLST, ACB, or RPL; or the actual length of a particular ACB, RPL, or EXLST. You test for a standard length by omitting the {ACB|EXLST|RPL} operand and coding only one (or more) of these length operands and no other operands. You can test the actual length of a control block or list by specifying the {ACB|EXLST|RPL} operand and the corresponding length operand.

Attributes of an Open File or Index: After a file is opened, the ACB contains information that it does not contain before it is opened or after it is closed. Whether you are testing for the attributes of the data or the index of a key-sequenced file is determined by the OBJECT operand. By coding OFLAGS=OPEN, you can test whether the file is open. You can test the following fields:

Operand	Meaning
AVSPAC	Number of bytes of available space in the data or index component.
BUFNO	Number of buffers being used for the data or index component.
CINV	Size of a control interval in the data or index component.
FS	Percent of free control intervals in each data control area of a key-sequenced file.
KEYLEN	Full length of the prime key or alternate key field in each logical record (depending on whether or not you access the base cluster via a path).
LRECL	Maximum length of a logical record or, for an index, the index control interval size minus seven bytes.
NCIS	Number of control-interval splits in the file.
NDELR	Number of data records deleted from the file.
NEXCP	Number of EXCP commands issued since the data or the index was opened.
NEXT	Number of logical extents, data spaces or portions of data spaces, now allocated to the data or index component.
NINSR	Number of records inserted into the file. For a relative-record file, number of valid records, that is, non-empty slots in the file.
NIXL	Number of levels in the index of a key-sequenced file.
NLOGR	Number of data records in the file. For a relative-record file, total number of slots (empty or non-empty) in the used control intervals.
NRETR	Number of data records retrieved from the file.
NSSS	Number of control-area splits in a key-sequenced file.
NUPDR	Number of data records updated in the file.
RKP	Displacement of the prime key or alternate key field from the beginning of a data record (depending on whether or not you access the base cluster via a path); the same value is displayed whether the object is index or data.
STMST	System timestamp; the time and day (in microseconds) when the data or index component was last closed. Bits 52 through 63 of the fields are unused.

You can also test for these attributes:

Operand	Meaning
ATRB=ESDS	Entry-sequenced file
ATRB=KSDS	Key-sequenced file
ATRB=RRDS	Relative-record file
ATRB=WCK	VSAM is verifying write operations
ATRB=SSWD	Sequence set of the index is adjacent to the file
ATRB=REPL	Index records are replicated
ATRB=SPAN	File contains spanned records
ATRB=UNQ	Unique alternate keys in alternate index

Furthermore, you can determine whether the opened object is a path, a base cluster, or an alternate index by coding:

OPENOBJ=PATH	Alternate index/base cluster pair (path)
OPENOBJ=BASE	Base cluster
OPENOBJ=AIX	Alternate index

OBJECT={DATA | INDEX}
specifies, for the open ACB of a key-sequenced file, whether the field being tested is for the data or the index. KEYLEN and RKP will contain the same value, no matter whether the data or the index is being tested. FS, NCIS, NDELR, NINSR, NIXL, NLOGR, NRETR, NSSS, and NUPDR will contain zeros if the index is being tested.

Examples of the TESTCB Macro: The following example shows how the TESTCB macro can be used to test values in a VSAM control block.

Continuation characters required in column 72 are not shown in the example.

The TESTCB is used to determine whether or not a file is open:

```
          TESTCB ACB=(2),OFLAGS=OPEN,
                 ERET=TESTERR
          BE     OPEN
          B      UNOPEN
          .
          .
TESTERR   .....
```

The TESTCB is used to determine whether the LERAD exit routine was entered because of an end-of-file condition or a processing error (the example assumes that no EODAD exit routine was provided).

```
LOGERR    TESTCB RPL=(4),FDBK=4,
                 ERET=TESTERR
          BE     EODATA
          B      ERROR
          .
          .
TESTERR   ......
```

Return Codes from the Control-Block Manipulation Macros: When VSAM returns to your processing program after a GENCB, MODCB, SHOWCB, or TESTCB request, register 15 contains one of the following return codes:

Return Code	Meaning
X'00'	Operation successfully completed.
X'04'	An error occurred; the error code described in *VSE/VSAM Messages and Codes* identifies the specific error.
X'0C'	Request was not executed because an error was encountered while VSAM routines were being loaded.

If register 15 contains X'04', an error code is set in register 0, which indicates the type of error. Make sure that, before issuing the macro, you save the contents of register 0 if you want to use its contents later on. For an explanation of the error codes, see *VSE/VSAM Messages and Codes.*

List, Execute, and Generate Forms of the Control-Block Manipulation Macros

The list and execute forms of the control block manipulation macros (GENCB, MODCB, SHOWCB, and TESTCB) allow you to save virtual storage by using one parameter list for two or more macros. You can also make your program reenterable, that is, executable by more than one task at a time. While the generate form of the macros enables you to make programs reenterable it does not allow shared parameter lists.

List and Execute Forms

The list form of GENCB, MODCB, SHOWCB, and TESTCB has the same parameters as the standard form, except that it includes the parameter MF={L|(L,address[,label])}.

The parameter list of the macro is created inline when MF=L is coded. This version is not reenterable and register notation cannot be used for macro parameter addresses.

When MF=(L,address[,label]) is coded, the parameter list of the macro is created in the area specified by *address.* This form is reenterable. You must supply the area by a GETVIS macro when your program is executed. You can determine the size of the parameter list by coding the third operand *label.* VSAM equates *label* to the length of the list.

The execute form produces the executable code of the macros. The execute form is also identical to the standard form, except that it includes the operand MF=(E,address), where *address* points to the parameter list created by the list form of the macro. All of the other operands of the macro are optional and are coded only to change entries in the parameter list before the list is used. However, you cannot use the execute form to add or delete entries from the parameter list or to change the type of list.

Generate Form

The generate form of the macros allows you to make your program reenterable, but it does not create shared parameter lists. The generate form is the same as the standard form, except that you code MF=(G,address[,label]). The parameter list is created in an area pointed to by *address.* To ensure that the parameter list is reenterable, *address* should be coded in register notation. You must obtain this area by a GETVIS macro when the program is executed. You can determine the size of the parameter list by coding the third operand *label.* VSAM equates *label* to the length of the list.

Examples of the List, Execute, and Generate Forms: Figure 5-9 and 5-10 show the use of the list, execute, and generate forms of the control block manipulation macros.

In Figure 5-9, MODCB is used to place the length of a record in the RPL before the record is written. The list and execute forms are used so that only one parameter list is created even though the macro is issued several times. This list form is not reenterable.

In Figure 5-10, the generate form is used to create an ACB. It is reenterable because both the ACB itself and the parameter list of the GENCB macro are created in areas obtained through a GETVIS macro.

Continuation characters required in column 72 are not shown in the examples.

```
        MODCB  MF=(E,LENMOD),RECLEN=(7)      Current length in reg. 7
        LTR    15,15                        MODCB successful?
        BNZ    MODERR                       No, go to error routine
        PUT    RPL=LIST                     Yes, write record
        .
        .
        .
        MODCB  MF=(E,LENMOD)                Length is 100 bytes
        LTR    15,15                        MODCB successful?
        BNZ    MODERR                       No, go to error routine
        PUT    RPL=LIST                     Yes, write record
        .
        .
        .
LENMOD  MODCB  RPL=LIST,RECLEN=100,MF=L     List form has default
                                            length - 100 bytes
```

Figure 5-9. Examples of the List and Execute Form

```
         LA     10,PARMLEN                  Load length for GETVIS
         GETVIS ADDRESS=(8),LENGTH=(10)     Get area for parm, list
         LTR    15,15                       GETVIS successful?
         BNZ    VISERR                      No, go to error routine
         GENCB  BLK=ACB,MF=(G,(8),PARMLEN),
                 EXLST=(3),BUFND=4,BUFNI=3,
                 BUFSP=11-64,DDNAME=VFILENM,
                 MACRF=(KEY,SEQ,DIR,OUT),
                 PASSWD=PASS
         LTR    15,15                       GENCB successful?
         BNZ    GENERR                      No, go to error routine
         LR     2,1                         Yes, save ACB address
         .
         .
         .
PASS     DC     FL1'6',C'CHANGE'
```

Figure 5-10. Example of the Generate Form

Requesting Access to a File

After you have set up your control blocks and opened your file, you can actually gain access to the file by issuing a request macro. Because you must have fully defined your request in an RPL, a request macro has only one operand — the address of the RPL (or the first RPL in a chain of RPLs) that defines the request.

The VSAM request macros are: GET, PUT, POINT, ERASE, and ENDREQ. With these macros you can initiate all requests for access to data. You can retrieve, store (insert or add), update, and delete records, and you can cancel a request in order to issue a request defined by another RPL.

VSAM utilizes the RPS (rotational position sensing) feature if your VSE system includes support for RPS and if the direct access storage device has the RPS feature. The use of the RPS feature is independent of any specification in your program.

If a base cluster is accessed in a partition, once using a path and once not using a path, a no record found or duplicate key error can occur. Errors of this kind can be avoided by using Local Shared Resources.

The use of the request macros and the request options is illustrated in examples which follow the discussion of the macros.

GET Macro (Retrieve a Record)

When your program issues a request macro, its processing does not continue until VSAM completes the request. At that time, VSAM sets a return code in register 15. If end-of-file is reached or an error or other special condition occurs during the request, VSAM sets a code containing additional information in the feedback (FDBK) field of the RPL and takes any required exit. The return codes and codes set in the feedback field of the RPL are described later in this section. The format of the macro is:

```
[name] GET     RPL={address|(1)}
```

`name`
one through eight characters that provide a symbolic address for the GET macro.

`RPL=address|(1)`
specifies the address of the RPL (or the first RPL in a chain of RPLs) that defines the GET request.

You may specify the address in register notation (using a register from 2 through 12, enclosed in parentheses) or specify it with an expression that generates a valid relocatable A-type address constant.

When you issue a GET macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue the macro from within one of your exit routines such as LERAD or SYNAD, you must provide a second 72-byte save area for use by VSAM.

This macro retrieves the next record in key sequence or the record with the next higher relative-record number with RPL operand OPTCD=(KEY,SEQ), and the next record in entry sequence with OPTCD=(ADR,SEQ). It retrieves the record specified by the key or relative record number in the search-argument field with OPTCD=(KEY,SKP) or OPTCD=(KEY,DIR), and by the RBA in the search-argument field with OPTCD=(ADR,DIR). With skip sequential retrieval, each key or relative-record number that you specify must be

greater in collating sequence than the key or relative-record number of the previous record retrieved.

GET retrieves the next control interval with OPTCD=(CNV,SEQ) and the control interval specified by the RBA in the search-argument field with OPTCD=(CNV,DIR).

You must issue a GET with OPTCD=UPD to update (PUT with OPTCD=UPD) or to delete (ERASE) a record. You can have the record moved to your work area (OPTCD=MVE) or you can have VSAM leave the record in its I/O buffer and pass you the address of the record (OPTCD=LOC). The AREA operand of the RPL macro points to your work area or to a field in which VSAM will place a record address.

You can also keep VSAM positioned for subsequent sequential or skip sequential procesing when you issue a direct GET request with OPTCD=(DIR,NSP) or OPTCD=(DIR,UPD). With OPTCD=(DIR,UPD) however, positioning is canceled when you issue a PUT for update or an ERASE following the GET for update.

If VSAM does not already have positioning for the RPL (or chain of RPLs) for which the GET request is to be issued, then you may have to issue an ENDREQ macro for a different RPL. An ENDREQ must be issued to free a position if the number of positions that VSAM must remember is already the same as the value specified in the STRNO=number operand of the pertinent ACB macro. At any particular time, VSAM will be remembering positions for any request macro in process by VSAM, and for a succeeding request for any RPL (or chain of RPLs) for which the preceding request was one of the following:

Macro	OPTCD=value(s) in RPL
GET	DIR,LOC DIR,MVE,NSP DIR,MVE,UPD
POINT	any
PUT	DIR,NSP SEQ SKP
ERASE	SEQ SKP

PUT Macro (Store a Record)

```
[name] PUT     RPL={address|(1)}
```

`name`
one through eight characters that provide a symbolic address for the PUT macro.

`RPL=address|(1)`
specifies the address of the RPL (or the first RPL in a chain of RPLs) that defines the PUT request. You can specify address in register notation (using a register from 2 through 12, in parentheses) or specify it with an expression that generates a valid relocatable A-type address constant.

When you issue a PUT macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue the macro from within one of your exit routines such as LERAD or SYNAD, you must provide a second 72-byte save area for use by VSAM.

This macro stores a new record in key sequence or relative-record sequence if one of the following combinations of options is set in the RPL:

OPTCD=(KEY,DIR,NSP)
OPTCD=(KEY,SKP,NUP)
OPTCD=(KEY,DIR,NUP)
OPTCD=(KEY,SEQ,NUP)
OPTCD=(KEY,SEQ,NSP)

If you specify OPTCD=(KEY,DIR,NSP), VSAM is kept positioned at the next record in key sequence or relative-record sequence for subsequent sequential processing.

PUT stores a new record at the end of an entry-sequenced file with OTPCD= ADR (you cannot store a new record in a key-sequenced file with addressed access).

With skip sequential storage, OPTCD=(KEY,SKP), the key or relative-record number of each record that you store must be greater in collating sequence than the key or relative-record number of the previous record stored.

With control-interval access, OPTCD=(CNV,NUP), PUT stores a new control interval at the end of an entry sequenced file.

When loading or extending a file with the PUT macro, you must specify sequential or skip sequential processing(OPTCD=SEQ or OPTCD=SKP).

To store a changed record or control interval, you must have previously retrieved it with option OPTCD=UPD set in the RPL for both the GET and the PUT. You cannot change the length of a record in either a relative record file or an entry-sequenced file.

The record to be added or updated with a PUT macro must be in your work area(OPTCD=MVE); you cannot use OPTCD=LOC with the PUT macro. The AREA operand of the RPL macro points to your work area.

You may have to write an ENDREQ macro before you can issue a PUT NUP or PUT NSP request in your program. Information about this possible requirement is given at the end of the discussion of the GET macro.

POINT Macro (Position for Access)

```
[name] POINT  RPL={address|(1)}
```

`name`

one through eight characters that provide a symbolic address for the POINT macro.

`RPL=address|(1)`

specifies the address of the RPL (or the first RPL in a chain of RPLs) that defines the POINT request. You can specify address in register notation (using a register from 2 through 12, in parentheses) or specify it with an expression that generates a valid relocatable A-type address constant.

When you issue a POINT macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue the macro from within one of your exit routines such as LERAD or SYNAD, you must provide a second 72-byte save area for use by VSAM.

When OPTCD=KEY was specified in the pertinent RPL, this macro positions VSAM at the record whose key or relative-record number you specify in the search argument field. You can use the macro to position VSAM for subsequent sequential or skip sequential processing, either forward or backward in the file.

When OPTCD=ADR or OPTCD=CNV was specified in the pertinent RPL, the POINT macro positions VSAM at the record or control interval whose RBA you specify in the search argument field. You can cause the macro to position VSAM for subsequent sequential processing, either forward or backward in the file.

Note: You cannot issue the POINT macro in one mode of access-OPTCD=ADR, for example, change to another mode such as OPTCD=KEY, and then request VSAM to continue processing the file sequentially. This will result in termination of the request with an error.

VSAM can also be positioned for sequential processing by either a direct GET or a direct PUT as described in the preceeding sections on the GET and PUT macros.

You may have to issue an ENDREQ macro before you can issue a POINT request in your program. Information about this possible requirement is given at the end of the discussion of the GET macro.

ERASE Macro (Delete a Record)

```
[name] ERASE   RPL={address|(1)}
```

`name`
one through eight characters that provide a symbolic address for the ERASE macro.

`RPL=address|(1)`
specifies the address of the RPL (or the first RPL in a chain of RPLs) that defines the ERASE request. You can specify address in register notation (using a register from 2 through 12, in parentheses) or specify it with an expression that generates a valid relocatable A-type address constant.

When you issue an ERASE macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue the macro from within one of your exit routines such as LERAD or SYNAD, you must provide a second 72-byte save area for use by VSAM.

This macro deletes the record previously retrieved for update(with the GET macro,OPTCD=UPD). You can delete records in a key-sequenced file by keyed or addressed access, but you cannot delete records in an entry-sequenced file. You can delete records in a relative-record file by keyed access. You cannot delete control intervals (OPTCD=CNV).

ENDREQ Macro (Terminate a Request)

```
[name] ENDREQ RPL={address|(1)}
```

`name`
one through eight characters that provide a symbolic address for the ENDREQ macro.

`RPL=address|(1)`
specifies the address of the RPL (or first RPL in a chain of RPLs) that defines the request to be terminated. You can specify address in register notation (using a register from 2 through 12, in parentheses) or specify it with an expression that generates a valid relocatable A-type address constant.

When you issue an ENDREQ macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue the macro

from within one of your exit routines such as LERAD or SYNAD, you must provide a second 72-byte save area for use by VSAM.

This macro causes VSAM to end a request, that is, to forget its position for the specified RPL and to release its associated buffers for use by another RPL. Before you can issue a request specifying an RPL for which an ENDREQ macro was executed, you have to reposition VSAM.

An ENDREQ macro is required in your program whenever you have already issued as many concurrent active requests as you have specifed for STRNO operand of the ACB and you want to issue yet another request. (See the discussion at the end of the GET macro section.)

If an I/O operation was started, it will be allowed to complete. Also, I/O operations required to maintain the integrity of the file will be performed.

If the request involves a chain of RPLs, all records specified by the request may not be processed. For example, two RPLs are chained in a PUT request to add two new records to the file and an ENDREQ is issued after VSAM started the I/O operation to add the first record. That I/O operation will be completed and, if it causes a control-interval split, subsequent I/O operations will be performed to complete the split and update the index. However, VSAM will then return control to the processing program without adding the second record.

The ENDREQ macro causes VSAM to cancel the position in the file established for that request and also invalidates data and index buffers to force refreshing of all requests subsequent to the end request.

Return Codes for the Request Macros

When VSAM returns to your processing program, a return code in register 15 indicates what happened. If an error occured, additional information about the request will be in the RPL. Your processing program can examine the feedback field of the RPL with the SHOWCB or TESTCB macro. Register 1 contains the address of the RPL that defines the request that caused the error.

Control is returned to the instruction following your action macro when (1) the request is completed, (2) the request was not accepted because another request was using the RPL, or (3) an error occured and you did not have an active exit routine. If you have an active exit routine and it returns control to VSAM after processing the error, VSAM then returns control to the instruction following the action macro.

When you gain control after a request, register 15 will contain one of the following return codes:

Return Code	Meaning
X'00'	Request completed successfully; the RPL might contain additional (non-error) information about the request. (See "Error Codes from Request Macros" in *VSE/VSAM Messages and Codes.*)
X'04'	This request was not accepted because end-of-file was encountered or a request from another task is active on the same RPL; no additional information in the RPL.
X'08	Logical error; the error code in the RPL identifies the specific error. (See "Error Codes from Request Macros" in *VSE/VSAM Messages and Codes*).
X'0C'	Uncorrectable I/O error; the error code in the RPL identifies the specific error. (See "Error Codes from Request Macros" in *VSE/VSAM Messages and Codes*).

If the request is completed with a logical error (X'08), your LERAD exit routine is entered if you specified the LERAD exit in the EXLST and it is active. When you reach end-of-file, error code X'04 is set and your EODAD

exit routine will be entered. If you have no EODAD exit routine or it is inactive, your LERAD exit routine is entered.

If the request completed with an I/O (physical) error (X'0C'), your SYNAD exit routine is entered if you specified the SYNAD exit in the EXLST and it is active. The return code is not set in register 15 when the exit routine is entered. VSAM sets the return code in register 15 and returns control to the instruction following the action macro after you complete the exit routine.

The feedback field in the RPL (FDBK operand in SHOWCB and TESTCB) is a three-byte field with the following format:

0000xx

where:

xx = an error code which describes the error or, if the return code is zero, additional information about the request. The error codes are listed in "Error Codes from Request Macros" in *VSE/VSAM Messages and Codes*).

Besides the return code (set in register 15) and the error code (which you may obtain by specifying FDBK in the SHOWCB macro) a function code is provided for alternate-index processing. This function code is set on logical or physical errors detected by VSAM and indicates whether the respective error condition occured during accessing the base cluster or the alternate index. In addition, the function code indicates whether or not the upgrade set is still correct after the request that failed. The function codes and their meanings are:

Function Code	Meaning
X'00'	Condition occurred during accessing the base cluster. Upgrade set is correct.
X'01'	Condition occured during accessing the base cluster. Upgrade set may be incorrect as a consequence of this request.
X'02'	Condition occured during accessing the AIX over a base cluster. Upgrade set is correct.
X'03'	Condition occured during accessing the AIX over a base cluster. Upgrade set may be incorrect as a consequence of this request.
X'04'	Condition occurred during upgrade processing. Upgrade set is correct.
X'05'	Condition occured during upgrade processing. Upgrade set may be incorrect as a result as a consequence of this request.

You can display or test the function code by specifying the keyword FTNCD in the SHOWCB or TESTCB macro, respectively.

Examples of Using the Request Macros

The following examples show the essential macros and operands required to do the indicated operations. They illustrate the relationship between the ACB MACRF operand, the RPL OPTCD operand, and the request macros themselves. They show how to use the other operands as required by the assumptions for each example.

For your convenience in reading them, the examples show macros that generate control blocks at assembly (ACB, EXLST, and RPL) at the beginning of the example, rather than at the end where they would normally be placed with program constants. Each example assumes that the file has been opened and that it will be closed. Only nominal checks for errors are shown. Exit routines to analyze errors are not indicated.

Note: The continuation characters required in column 72 are not shown in the examples, nor are the asterisks required in column 1 of the comment cards shown.

How to Retrieve a Record: The GET Macro

Examples 1, 2, 3, 4, and 5 illustrate keyed and addressed, direct sequential, and skip sequential retrieval.

Example 1: Keyed Sequential Retrieval

Assumptions. Records moved to a work area. Fixed-length records, 100 bytes. Control blocks generated at assembly.

```
INPUT    ACB     MACRF=(KEY,SEQ,IN)     All MACRF and OPTCD options
                                         specified are defaults and
                                         could have been omitted.
RETRVE   RPL     ACB=INPUT
                 AREA=IN,AREALEN=100,
                 OPTCD=(KEY,SEQ,NUP,MVE)
         .
         .
         .
LOOP     GET     RPL=RETRVE             This GET or identical GETs
                                         can be issued, with no change
                                         in the request parameter list,
                                         to retrieve subsequent records
                                         in key sequence.
         LTR     15,15
         BNZ     ERROR
         .
         .
         .
         B       LOOP
ERROR    ...                            Request was not accepted or
         .                              failed.
         .
         .
IN       DS      CL100                  IN contains a data record
                                         after GET is completed.
```

Discussion. The records are retrieved in key sequence. No search argument has to be specified; VSAM is positioned at the first record in key sequence when the file is opened, and the next record is retrieved automatically as each GET is issued. The branch to ERROR will also be taken if the end of the file is reached.

Example 2: Skip Sequential Retrieval

Assumptions. Variable-length records: they are processed in the I/O buffer. The search argument is a full key, compared greater than or equal. Control blocks are generated at the time of execution. Key length is eight bytes.

```
         GENCB         BLK=ACB,                  VSAM gets an area in virtual
                       DDNAME=INPUT,             storage to generate the
                       MACRF=(KEY,SKP,IN)        access method control block
                                                 and returns the address in
                                                 register 1.
         LTR           15,15
         BNZ           CHECKO
         LR            2,1                       Address of ACB
         GENCB         BLK=RPL,ACB=(2),
                       AREA=RCDADDR,AREALEN=4,
                       ARG=SCRHKEY,
                       OPTCD=(KEY,SKP,NUP,KGE,FKS,LOC)
         LTR           15,15
         BNZ           CHECK0
         LR            3,1                       Address of the request
         .                                       parameter list.
         .
         .
LOOP     MVC           SRCHKEY,table             Search argument for
                                                 retrieval, moved in from
                                                 a table or a
                                                 transaction record.
         GET           RPL=(3)
         LTR           15,15
         BNZ           ERROR
         LR            1,3                        Put RPL address in register 1.
         SHOWCB        RPL=(1),                   Display the length of the
                       RECLEN=(0)                 record.
         LTR           15,15
         BNZ           CHECK0
         ST            0,RCDLEN                   Save the record length.
         .
         .
         .
         B             LOOP
ERROR    ...                                      Request was not accepted
                                                  or failed.
CHECKO   ...                                      Generation or display failed
         .
         .
         .
RCDADDR  DS            F                          Work area into which VSAM
                                                  puts the address of a data
                                                  record within the I/O buffer
                                                  (OPTCD=LOC).
SRCHKEY  DS            CL8                        Search argument for
                                                  retrieval.
RCDLEN   DS            F                          For retrieving variable
                                                  record lengths.
```

Discussion. The records are retrieved in key sequence, but some records are skipped. Skip sequential retrieval is very similar to keyed direct retrieval (see example 4), except that you must retrieve records in ascending sequence (with skips) rather than in a random sequence.

Internally, with skip sequential retrieval, VSAM uses only the sequence set of the index to skip ahead; with direct retrieval it searches the index from top to bottom to locate a record.

Example 3: Addressed Sequential Retrieval

Assumptions. Many records are retrieved with one GET request. Records are moved to work areas (only option); they are of fixed length, 20 bytes long. Chain of RPLs is generated during execution.

```
BLOCK     ACB       DDNAME=INPUT,
                    MACRF=(ADR,SEQ,IN)
          .
          .
          .
          GENCB     BLK=RPL,
                    COPIES=10,
                    ACB=BLOCK,
                    OPTCD=(ADR,SEQ,NUP,MVE)
          LTR       15,15
          BNZ       CHECKO
          LA        5,WORKAREA              Address of the first
                                            work area.
          LA        3,10                    Number of lists(10).
          LR        2,1                     Address of the first list.
          LR        1,0                     Length of all of the lists.
                                            Registers 0 and 1 contain
                                            length and address of the
                                            generated control blocks when
                                            VSAM returns control after
                                            GENCB.
          SR        0,0                     Prepare for following
                                            division.
          DR        0,3                     Divide number of lists into
                                            length of all the lists.
          LR        3,1                     Save the resulting length of a
                                            single list for an offset.
          LR        4,2                     Save address of the first
                                            list.
```

Do the following 6 instructions 10 times to set up all of the request parameter lists. The tenth time, register 4 must be set to 0 to indicate the last request parameter list in the chain.

```
          AR        4,3                     Address of the next list.
          MODCB     RPL=(2),                In each request parameter
                    NXTRPL=(4),             list, indicate the address
                    AREA=(5),AREALEN=20     of the next list and the
                                            address and length of the
                                            work area.
          LTR       15,15
          BNZ       CHECKO
          AR        2,3                     Address the next line.
          LA        5,20(5)                 Address the next work area.
          .                                 Restore register 2 to address
          .                                 the first list before
          .                                 continuing to process.
LOOP      GET       RPL=(2)
          LTR       15,15
          BNZ       ERROR
          .                                 Process the ten records that
          .                                 have been retrieved by the
          .                                 GET.
          B         LOOP
CHECKO    ...
ERROR     ...                               Display the feedback field
                                            (FIELDS=FDBK) of each
                                            request parameter list to
                                            find out which one had an
                                            error.
WORKAREA  DS        CL200                   Space for a work area for
                                            each of the 10 request
                                            parameter lists.
```

Discussion. The records are retrieved in entry sequence. In a key-sequenced file that has had control-interval or control-area splits, it is likely that the entry sequence of the records is no longer the same as their key sequence. Each of the ten RPLs in the chain identifies a record to be retrieved by the GET. VSAM moves each record into the work area provided for the request parameter list that identifies the record.

If an error occurred for one of the RPLs in the chain and you have supplied error analysis routines, VSAM takes a LERAD or SYNAD exit before returning to your program. Register 15 indicates the status of the request: a code of 0 indicates that no error was associated with any of the RPLs; any other code indicates that an error occured for one of the RPLs. Issue a SHOWCB for each RPL in the chain to find out which one had an error. VSAM does not process any of the RPLs beyond the one with an error.

Example 4: Keyed Direct Retrieval

Assumptions. Fixed-length records are processed in the I/O buffer. Key length is 15 bytes. The search argument is a 5-byte generic key, compared equal. Control blocks are generated during assembly.

```
INPUT    ACB     MACRF=(KEY,DIR,IN)
RETRVE   RPL     ACB=INPUT,              You specify all parameters
                 AREA=IN,AREA=4          for the request in the RPL
                 OPTCD=(KEY,DIR,         macro.
                 NUP,KEQ,GEN,LOC),
                 ARG=KEYAREA,KEYLEN=5
         .
         .
         .
LOOP     MVC     KEYAREA,table           Search argument for retrieval,
                                         moved in from a table or a
                                         transaction record.
         GET     RPL=RETRVE              This GET or identical GETs can
                                         be issued with no change in the
                                         RPL: just specify each new
                                         search argument in the field
                                         KEYAREA.
         LTR     15,15
         BNZ     ERROR
         .
         .                               Process the record.
         .
         B       LOOP
ERROR    ...                             Request was not accepted
         .                               or failed.
         .
         .
IN       DS      CL4                     VSAM stores the record here.
         .
         .
KEYAREA  DS      CL5                     You specify the search
                                         argument here.
```

Discussion. The generic key specifies a class of records. For example, if you search on the first third of employee number, you get the first of presumably several records that start with the specified characters. To retrieve all of the records in that class, either switch to sequential access(see example 7) or to a full-key search with greater-than-or-equal comparison (example 2), increasing the key of each record you retrieve to the next possible key value.

Example 5: Addressed Direct Retrieval

Assumptions. Fixed-length records,20 bytes long, are moved to a work area.

```
BLOCK    ACB      DDNAME=INPUT,            Access-method control block
                  MACRF=(ADR,DIR,IN)       generated at assembly.
         GENCB    BLK=RPL,COPIES=1,        Request parameter list
                  ACB=BLOCK,               generated at execution.
                  OPTCD=(ADR,DIR,
                  NUP,MVE),
                  ARG=SRCHADR,
                  AREA=IN,AREALEN=20
         LTR      15,15
         BNZ      CHECKO
         LR       2,1                      Address of the list.
         .
         .
         .
LOOP     MVC      SRCHADR,table            Search argument for retrieval,
                                           calculated or moved in from a
                                           table or a transaction record.
         GET      RPL=(2)
         LTR      15,15
         BNZ      ERROR
         .
         .                                 Process the record.
         .
         B        LOOP
CHECKO   ...                               Generation failed.
ERROR    ...                               Request was not accepted
         .                                 or failed.
         .
         .
IN       DS       CL20                     VSAM puts a record here
                                           for each GET request.
SRCHADR  DS       CL4                      You specify the RBA search
                                           argument here for each request.
```

Discussion. The RBA provided for a search argument must match the RBA of a record. Keyed insertion and deletion of records in a key-sequenced file will probably cause the RBAs of some records to change. Therefore, if you process a key-sequenced file by addressed direct access(or by addressed sequential access using POINT), you need to keep track of changes. You can use the JRNAD exit for this purpose.

How to Position for Subsequent Sequential Access: The GET and POINT Macros

Examples 6 and 7 illustrate positioning both with the POINT macro and with direct access followed by sequential access.

Example 6: Keyed Positioning with POINT

Assumptions. Sequential access. The search argument(for positioning) is a full key of 5 bytes, compared equal. Records are 50 bytes long. Control blocks are generated during assembly.

```
BLOCK    ACB      DDNAME=10                Default MACRF options
                                           sufficient.
POSITION RPL      ACB=BLOCK,               ARG operand and KEQ and
                  AREA=WORK,AREALEN=50,    FKS OPTCD options define
                  ARG=SRCHKEY,             the POINT request.
                  OPTCD=(KEY,SEQ,
                  KEQ,FKS)
         .
         .
         .
LOOP     MVC      SRCHKEY,table            Search argument for
                                           positioning, moved in from a
                                           table or transaction record.
         POINT    RPL=POSITION
         LTR      15,15
         BNZ      ERROR
LOOP1    GET      RPL=POSITION
         LTR      15,15
         BNZ      ERROR
         .                                 Process the record.  Decide
         .                                 whether to skip another
         .                                 position (forward or
                                           backward).
         BE       LOOP                     Yes, skip.
         B        LOOP1                    No, continue in consecutive
                                           sequence.
ERROR    ...                               Request was not accepted or
         .                                 failed.
         .
         .
SRCHKEY  DS       CL5                      Search-argument field for
                                           POINT request.
WORK     DS       CL50                     VSAM puts a record here for
                                           each GET request.
```

Discussion. No access is gained to a record with POINT. POINT causes VSAM to be positioned ahead or back to the specified record for a subsequent sequential GET request, which retrieves the record. If, after positioning, you issue a direct request by way of the same RPL, VSAM does not remember the position established by the POINT. VSAM would then either be positioned somewhere else or not positioned at all, depending on whether OPTCD=NSP or UPD was specified or OPTCD=NUP (see example 7).

Positioning by address is identical to positioning by key, except that the search argument is an RBA, which must match with the RBA of a record in the file.

Example 7: Switching from Direct to Keyed Sequential Retrieval

Assumptions. Records are moved to a work area. The search argument (for the direct request preceeding sequential requests) is a generic key, 8 bytes long, compared equal. Records are of fixed-length, 100 bytes long. Positioning is requested for direct requests. Control blocks are generated during assembly.

```
INPUT    ACB     MACRF=(KEY,DIR,          Both direct and sequential
                 SEQ,IN)                  access specified.
RETRVE   RPL     ACB=INPUT,
                 AREA=IN,AREALEN=100,
                 OPTCD=(KEY,DIR,
                 NSP,KEQ,GEN,MVE),        NSP specifies that VSAM is
                 ARG=KEYAREA,KEYLEN=8     to remember its position.
         .
         .
         .
LOOP     MVC     KEYAREA,table            Search argument for direct
                                          retrieval, moved in from a
                                          table or transaction record.
LOOP1    GET     RPL=RETRVE
         LTR     15,15
         BNZ     ERROR
         .                                Decide whether to switch from
         .                                one type of access to the
         .                                other.  If now sequential:
                                          To remain sequential, branch
                                          to LOOP1
                                          To switch to direct, branch
                                          to DIR
                                          If now direct:
                                          To remain direct, branch
                                          to LOOP
                                          To switch to sequential,
                                          branch to SEQ
SEQ      MODCB   RPL=RETRVE,              Alter request parameter list
                 OPTCD=SEQ                for sequential access.
         LTR     15,15
         BNZ     CHECKO
         B       LOOP1                    No search argument required.
DIR      MODCB   RPL=RETRVE,              Alter request parameter list
                 OPTCD=DIR                for direct access.
         LTR     15,15
         BNZ     CHECK0
         B       LOOP                     Prepare new search argument.
ERROR    ...                              Request was not accepted or
                                          failed.
CHECKO   ...                              Modification failed.
         .
         .
         .
IN       DS      CL100                    VSAM puts retrieved records
                                          here.
KEYAREA  DS      CL8                      Specify the generic key for
                                          a direct request here.
```

Discussion. Positioning is associated with an RPL; thus to switch from direct to sequential access without independently establishing positioning for the sequential access, modify a single RPL that alternately defines requests for both types of access rather than use a different RPL for each type.

With direct retrieval, VSAM does not remember its position for subsequent retrieval unless you explicitly requested this(OPTCD=NSP). After a direct GET for update (OPTCD=UPD), VSAM is positioned for a subsequent PUT,ERASE, or sequential GET (if you modify OPTCD(DIR,UPD) to OPTCD=(SEQ,UPD)). If you modify OPTCD=(DIR,NUP) to

OPTCD=SEQ, you must issue a POINT to get VSAM positioned for sequential retrieval, as NUP indicates that no positioning is desired with a direct GET.

If you have chained many RPLs together, one position is remembered for the whole chain. For example, if you issue a GET that gives the address of the first RPL in the chain, the position of VSAM when the GET request is complete is at the record following the one defined by the last RPL in the chain. Therefore, modifying OPTCD=(DIR,NSP) in each RPL in a chain to OPTCD=SEQ implies continuing with sequential access relative to the last of the direct RPLs.

How to Chain Request Parameter Lists and Terminate a Request: The ENDREQ Macro

Example 8 illustrates how to chain RPLs. Example 9 illustrates the use of ENDREQ to cause VSAM to give up its position for a request to be able to remember its position for another request.

Example 8: Chaining Request Parameter Lists

Assumption. Records are 50 bytes long. Retrieved records are moved to a work area. Three RPLs are chained.

```
FIRST    RPL     ACB=BLOCK,
                 AREA=AREA1,AREALEN=50,
                 NXTRPL=SECOND
SECOND   RPL     ACB=BLOCK,
                 AREA=AREA2,AREALEN=50,
                 NXTRPL=THIRD
THIRD    RPL     ACB=BLOCK              Last list does not indicate
                 AREALEN=50             a next list.
                 AREALEN=50
         .
         .
         .
LOOP     GET     RPL=FIRST              Request gives the address of
                                        the first request parameter
                                        list.
         LTR     15,15
         BNZ     ERROR
         .                              Process the three records
         .                              retrieved by the GET.
         .
         B       LOOP
ERROR    ...                            Display the feedback field
                                        (FIELD=FDBK) of each request
                                        parameter list to find out
                                        which one had an error.
AREA1    DS      CL50                   A single GET request causes
                                        VSAM to put a record in each
                                        of AREA1,AREA2, and AREA3.
AREA2    DS      CL50
AREA3    DS      CL50
```

Discussion. If an error ocurred for one of the RPLs in the chain and you have supplied error-analysis routines, VSAM takes a LERAD or SYNAD exit before it returns control to your program. Register 15 is set to indicate the status of the request. A code of 0 indicates that no error was associated with any of the RPLs. Any other code indicates that an error occurred for one of the RPLs. You should issue a SHOWCB macro for each RPL in the chain to find out which one had an error. VSAM does not process any of the RPLs beyond the one with an error.

Example 9: Giving up Positioning for another Request

Asumptions. There are three RPLs, all of which require VSAM to remember its position, one only temporarily and two until VSAM is explicitly requested to forget its position. VSAM can remember only two positions concurrently (STRNO=2)

```
BLOCK    ACB      MACRF=(SEQ,DIR),
                  STRNO=2
SEQ      RPL      ACB=BLOCK,              VSAM must remember its
                  OPTCD=SEQ               position.
DIRUPD   RPL      ACB=BLOCK,              VSAM must remember its
                  OPTCD=(DIR,UPD)         position until explicitly
                                          requested to forget it by
                                          PUT or ENDREQ.
DIRNUP   RPL      ACB=BLOCK,              VSAM must be able to
                  OPTCD=(DIR,NUP)         temporarily remember its
         .                                position.
         .
         .
LOOP     GET      RPL=SEQ                 VSAM now remembers its
                                          position for this request.
         LTR      15,15
         BNZ      ERROR
         GET      RPL=DIRNUP              VSAM remembers its position
                                          for this request only while
                                          it is processing the request.
         LTR      15,15
         BNZ      ERROR
         GET      RPL=DIRUPD              VSAM can therefore remember
                                          its position for this request,
                                          even though STRNO=2.
         LTR      15,15
         BNZ      ERROR
         .
         .                                Decide whether to update the
         .                                record.
         BE       UPDATE
         B        FORGET                  No.
UPDATE   PUT      RPL=DIRUPD              Yes, update the record,
                                          causing VSAM to forget its
                                          position for DIRUPD.
         LTR      15,15
         BNZ      ERROR
         B        LOOP
FORGET   ENDREQ   RPL=DIRUPD              Cause VSAM to forget its
                                          position for DIRUPD.
         LTR      15,15
         BNZ      ERROR
         B        LOOP
ERROR    ...                              Request was not
                                          accepted or failed.
```

Discussion. The use of ENDREQ illustrated here is to cause VSAM to forget its position for one RPL so a request defined by another RPL can be issued.When PUT is issued after a GET RPL=DIRUPD request, ENDREQ need not be issued, because PUT causes VSAM to forget its position (the next GET with RPL=DIRUPD does not depend on VSAM's remembering its position). You need to cause VSAM to forget its position when you have issued requests for as many RPLs requiring concurrent positioning as the number you specified for STRNO (in the ACB macro) and you want to issue a request for yet another RPL. In the example, a GET with RPL=DIRNUP cannot be reissued unless VSAM is freed from remembering its position for either SEQ or DIRNUP. VSAM must be allowed to remember its position for SEQ because requests against this RPL are sequential and depend on VSAM's remembering its position.

To cause VSAM to give up its position associated with a chain of RPLs, specify the first RPL in the chain in your ENDREQ macro.

Because VSAM remembers its position after a direct GET with OPTCD=UPD, if no PUT or ENDREQ follows, you can switch to sequential access (OPTCD=(SEQ,UPD) or OPTCD=SEQ) and use the positioning for a GET.

How to Store a Record: The PUT Macro

Examples 10,11,12,and 13 illustrate the storage of records: keyed and addressed, sequential, skip sequential, and direct.

Example 10: Keyed Sequential Insertion

Assumptions. Records of variable length are moved from a work area (only option). These records are up to 250 bytes long. Key length is 15 bytes. Some records are inserted between existing records, others are added at the end of the file.

```
BLOCK     ACB      DDNAME=OUTPUT,
                   MACRF=(KEY,SEQ,OUT)
LIST      RPL      ACB=BLOCK,
                   AREA=BUILDRCD,AREALEN=250,
                   OPTCD=(KEY,SEQ,NUP,MVE)
          .
          .
LOOP      L        0,length              Put length of record to be
                                           inserted into register 0.
          LA       1,LIST                Put RPL address into register 1.
          MODCB    RPL=(1),              Modify record length in
                   RECLEN=(0)            request parameter list.
          LTR      15,15
          BNZ      CHECKO
          PUT      RPL=LIST
          LTR      15,15
          BNZ      ERROR
          B        LOOP
CHECKO    ...                            Modification failed.
ERROR     ...                            Request was not accepted
          .                              or failed.
          .
          .
BUILDRCD  DS      CL250                  Work area for building record.
```

Discussion. Sequential insertion does not require VSAM to be positioned at the point of insertion. VSAM automatically skips ahead (never back) to that point, as though you were using skip sequential insertion (see example 11). The difference between sequential and skip sequential insertion is that sequential insertion leaves free space in control intervals and control areas according to the file's FREESPACE specification in the catalog(which is entered by the Access Method Services DEFINE command). Skip sequential insertion (and direct insertion) uses the free space.

You must use sequential storage (as opposed to skip sequential or direct storage) when you load records into a file for the first time. Thereafter, you may use skip sequential and direct storage, but you should use sequential storage when you are inserting large numbers of records between two existing records or at the end of the file.

When you store records sequentially beyond the highest key in the file,VSAM automatically extends the file as though you were continuing to

load records. VSAM does not use distributed free space for these records, but establishes new control areas at the end of the file.

Example 11: Skip Sequential Insertion

Assumptions. Several records are inserted with one PUT request. The records are moved from a work area (only option). They are fixed-length, 100 bytes long.

```
OUTPUT   ACB      MACRF=(KEY,SKP,OUT)
         .
         .
         .
         GENCB    BLK=RPL,COPIES=5,        Generate 5 request parameter
                  ACB=OUTPUT,              lists at execution.
                  AREALEN=100,
                  OPTCD=(KEY,SKP,NUP,MVE)
         LTR      15,15
         BNZ      CHECKO
```

Calculate the length of each list and use register notation with the MODCB macro to complete each list. See example 3.

```
         MODCB    RPL=(2),
                  AREA=(3),
                  NXTRPL=(4)
         LTR      15,15
         BNZ      CHECKO
```

Increase the value in each register and repeat the MODCB until all five request parameter lists have been completed. The last time, register 4 must be set to 0.

```
         .
         .
         .
LOOP     ...                               Restore address of first list
                                           register 2.  Build 5 records
                                           in WORK.
         PUT      RPL=(2)                  Register 2 points to the first
                                           request parameter list in the
                                           chain.  The five records in
                                           WORK are stored with this one
                                           PUT request.
         LTR      15,15
         BNZ      ERROR
         .
         .
         .
         B        LOOP
CHECKO   ...                               Generation or modification
                                           failed.
ERROR    ...                               Display the feedback field
                                           in each request parameter
                                           list to find out if it had
                                           an error (see discussion
                                           for example 8).
WORK     DS       CL500                    Contains 5 100-byte work
                                           areas.
```

Discussion. You give no search argument for storage: VSAM knows the position of the key field in each record and extracts the key from it. Skip sequential insertion differs from keyed direct insertion in the sequence in which records may be inserted(ascending non-consecutive sequence versus random sequence) and in performance. With skip sequential insertion, VSAM uses only the sequence set of the index to find the point of insertion;

with keyed direct insertion, VSAM searches from the top level of the index down to the sequence set.

With skip sequential insertion, if you insert two or more records into a control interval, VSAM does not write the contents of the buffer to direct-access storage until you have inserted all of the records. With direct insertion, VSAM writes the contents of the buffer after you have inserted each record.

Example 12: Keyed Direct Insertion

Assumptions. Records are moved from a work area (only option.) They are of fixed length and 100 bytes long.

```
OUTPUT   ACB      MACRF=(KEY,DIR,OUT)
DIRECT   RPL      ACB=OUTPUT,
                  AREA=WORK,AREALEN=100,
                  OPTCD=(KEY,DIR,NUP,MVE),
                  RECLEN=100
         .
         .
         .
LOOP     PUT      RPL=DIRECT
         LTR      15,15
         BNZ      ERROR
         .
         .
         .
         B        LOOP
ERROR    ...                              Request failed.
         .
         .
         .
WORK     DS       CL100                   Work area.
```

Discussion. VSAM extracts the key from each record's key field: you give no search argument. Using keyed direct access is very similar to using skip sequential access. About the only differences are specifying DIR instead of SKP in the MACRF and OPTCD operands and being able to process records randomly instead of in ascending key sequence (with skips).

Example 13: Addressed Sequential Addition

Assumptions. Records are moved from work area (only option). They are of variable-length, up to 100 bytes long.

```
BLOCK    ACB     MACRF=(ADR,SEQ,OUT)
LIST     RPL     ACB=BLOCK,
                 AREA=NEWRCD,AREALEN=100,
                 OPTCD=(ADR,SEQ,MVE)
         .
         .
         .
LOOP     ...                          Build the record.
         L       0,length             Put the length of the record
                                      into register 0.
         LA      1,LIST               Put RPL address into register 1.
         MODCB   RPL=(1),             Indicate length of new record.
                 RECLEN=(0)
         LTR     15,15
         BNZ     CHECKO
         PUT     RPL=LIST
         LTR     15,15
         BNZ     ERROR
         B       LOOP
CHECKO   ...                          Modification failed.
ERROR    ...                          Request was not accepted
         .                            or failed.
         .
         .
NEWCRD   DS      CL100                Build record in this work
                                      area.
```

Discussion. With addressed access, you cannot insert records into or add records to a key-sequenced file, because the index is not used and VSAM cannot locate the control interval into which to insert the record. You can add records to, but not insert records into, an entry sequenced-file. Each record is stored in the next position after the last record in the file. You do not have to specify an RBA or do any explicit positioning (with the POINT macro). Addressed addition of records is always identical to loading a file: when the last control area is filled up, VSAM extends the file and establishes new control areas.

Actually, there is no difference between addressed sequential and addressed direct addition; each method stores a record in the next position after the last record in the file. However, you cannot use direct processing to load records into a file for the first time; you must use sequential processing.

How to Update a Record: The GET and PUT Macros

Examples 14,15,and 16 illustrate updating a record by first retrieving it and then storing it back with changes. (You cannot update a record without first retrieving it for update.)

Example 14: Keyed Sequential Update

Assumptions. Records are updated in a work area (only option). They are fixed-length, 50 bytes long. Not every record retrieved is also updated.

```
UPDATA   ACB     MACRF=(KEY,SEQ,OUT)
LIST     RPL     ACB=UPDATA,              UPD indicates the record
                 AREA=WORK,AREALEN=50,    may be stored back(or
                 OPTCD=(KEY,SEQ,          deleted).
                 UPD,MVE)
         .
         .
         .
LOOP     GET     RPL=LIST
         LTR     15,15
         BNZ     ERROR
         .                                Decide whether to update
         .                                the record.
         .
         BE      UPDATE
         B       LOOP                     Do not update it; retrieve
                                          another.
UPDATE   .                                Update the record and store
         .                                it back.
         .
         PUT     RPL=LIST
         LTR     15,15
         BNZ     ERROR
         B       LOOP
ERROR    ...                              Request was not accepted
         .                                or failed.
         .
         .
         .
WORK     DS      CL50                     VSAM places the retrieved
                                          record here.
```

Discussion. A GET update (OPTCD=UPD) must precede a PUT for update. Besides retrieving the record to be updated, GET positions VSAM at the record retrieved, in anticipation of the suceeding update (or deletion). It is not necessary for you to store back (or delete) the record that you retrieved for update. VSAM's position at the record previously retrieved allows you to issue another GET to retrieve the following record (OPTCD=(SEQ,UPD) or OPTCD=SEQ). Then, however, you cannot store back the previous record; the position for update was not maintained because of the following GET.

This example requires the use of a work area because you cannot update a record in the I/O buffer. Skip sequential retrieval (with OPTCD=UPD) can be used to update. Compare this example with example 2.

Example 15: Keyed Direct Update

Assumptions. Records are moved to and from a work area(only option). They are of variable-length, up to 120 bytes(with some lengths changed by update). The search argument is a full key of five bytes, compared equal.

```
INPUT    ACB     MACRF=(KEY,DIR,OUT)
UPDATE   RPL     ACB=INPUT,              UPD indicates the record
                 AREA=IN,AREALEN=120,    may be stored back (or
                 OPTCD=(KEY,DIR,         deleted).
                 UPD,KEQ,FKS,MVE),
                 ARG=KEYAREA,KEYLEN=5
         .
         .
         .
LOOP     GET     RPL=UPDATE
         LTR     15,15
         BNZ     ERROR
         LA      1,UPDATE                Put RPL address in register 1.
         SHOWCB  RPL=(1),                Display the length of the
                 RECLEN=(0)              record.
         LTR     15,15
         BNZ     CHECKO
         ST      0,RLNGTH                Save the record length.
         .                               Update the record.  Does the
         .                               update change the record's
         .                               length?
         B       STORE                   No, length not changed.
         L       0,length                Yes, load new length into
                                         register 0.
         LA      1,UPDATE                Put RPL address in register 1.
         MODCB   RPL=(1),                Modify length indication in
                 RECLEN=(0)              the request parameter list.
         LTR     15,15
         BNZ     CHECKO
STORE    PUT     RPL=UPDATE
         LTR     15,15
         BNZ     ERROR
         B       LOOP
ERROR    ...                             Request was not accepted
                                         or failed.
CHECKO   ...                             Display or modification
         .                               failed.
         .
         .
IN       DS      CL120                   Work area for retrieving,
                                         updating, and storing a
                                         record
KEYAREA  DS      CL5                     Search argument for
                                         retrieving a record.
RLNGTH   DS      F                       Area for displaying the
                                         length of a retrieved record.
```

Discussion. You cannot update records in the I/O buffer. A direct GET for update positions VSAM at the record retrieved, in anticipation of storing back (or deleting) the record. This positioning also allows you to switch to sequential access to retrieve the next record.

You do not have to store back a record that you retrieve for update, but if you wish to store it back you must do so before another retrieval, (using the same RPL). Or else, use two RPLs with STRNO=2, one RPL is used solely for GET DIR with UPD.

Example 16: Addressed Sequential Update

Assumptions. Entry-sequenced file. Records are processed in a work area. They are of variable-length, up to 200 bytes long (lengths are not changed by updates; the length of a record can never be changed if addressed access is used).

```
ENTRY    ACB     MACRF=(ADR,SEQ,OUT)
ADRUPD   RPL     ACB=ENTRY,                UPD indicates update
                 AREA=WORK,                (or deletion).
                 AREALEN=200,
                 OPTCD=(ADR,SEQ,UPD,MVE)
         .
         .
         .
LOOP     GET     RPL=ADRUPD
         LTR     15,15
         BNZ     ERROR
         LA      1,ADRUPD                  Put RPL address in register 1.
         SHOWCB  RPL=(1),                  Find out how long the record is.
                 RECLEN=(0)
         LTR     15,15
         BNZ     CHECKO
         ST      0,RLNGTH                  Save the record length.
         PUT     RPL=ADRUPD
         LTR     15,15
         BNZ     ERROR
         B       LOOP
ERROR    ...                               Request was not accepted
                                           or failed.
CHECKO   ...                               Display failed.
         .
         .
         .
WORK     DS      CL200                     Record-processing work area.
RLNGTH   DS      F                         Display area for length of
                                           records.
```

Discussion. The RBAs of records in an entry-sequenced file are fixed, and free space is not distributed. Therefore, it is not possible to change the length of records in an entry-sequenced file.

If you have inactive records in your entry-sequenced file, you may reuse the space they occupy by retrieving the records for update and restoring a new record in their place.

With a key-sequenced file, it is also impossible to change the length of records by addressed update because the index is not used and VSAM could not split a control interval if required because of changing record length.

Addressed direct update differs from sequential update in the specification of an RBA for a search argument.

How to Delete a Record: The GET and ERASE Macros

Examples 17 and 18 illustrate deleting a record from a key-sequenced file.

Example 17: Keyed Direct Deletion

Assumptions. Records are processed in a work area(only option). They are fixed-length, 50 bytes long. Not every record retrieved for deletion is deleted. The search argument is a full key, 5 bytes long, compared equal.

```
DELETE   ACB     MACRF=(KEY,DIR,OUT)
LIST     RPL     ACB=DELETE,              UPD indicates deletion.
                 AREA=WORK,AREALEN=50,
                 ARG=KEYFIELD,
                 OPTCD=(KEY,DIR,UPD,MVE,FKS,KEQ)
         .
         .
         .
LOOP     MVC     KEYFIELD,table           Search argument for retrieval,
                                          from a table or transaction
                                          record.
         GET     RPL=LIST
         LTR     15,15
         BNZ     ERROR
         .
         .
         .                                Decide whether to delete
                                          the record.
         BE      DELET
         B       LOOP                     No, retrieve the next record.
DELET    ERASE   RPL=LIST                 Yes, delete the record.
         LTR     15,15
         BNZ     ERROR
         B       LOOP
ERROR    ...                              Request was not accepted
                                          or failed.
WORK     DS      CL50                     Examine the data record here.
KEYFIELD DS      CL5                      Search argument.
```

Discussion. When you retrieve a record for deletion (OPTCD=UPD, same as retrieval for update), VSAM is positioned at the record retrieved, in anticipation of a suceeding ERASE (or PUT) request for that record. You are not required to issue such a request, however. Another GET request nullifies any previous positioning for deletion or update.

Keyed sequential retrieval for deletion varies from direct in not using a search argument(except for possible use of the POINT macro). Skip sequential retrieval for deletion (OPTCD=(SKP,UPD) has the same effect as direct, but it is faster or slower depending on the number of control intervals seperating the records being retrieved.

Example 18: Addressed Sequential Deletion

Assumptions. Records are processed in a work area. They are fixed-length, 100 bytes long. A key-sequenced file (you cannot delete records from an entry-sequenced file.) Not every record retrieved for deletion is deleted. Skipping is effected by issuing the POINT macro.

```
DELETE   ACB     MACRF=(ADR,SEQ,OUT)
REQUEST  RPL     ACB=DELETE,              UPD indicates deletion.
                 AREA=WORK,
                 AREALEN=100,
                 ARG=ADDR,
                 OPTCD=(ADR,SEQ,UPD,MVE)
         .
         .
         .
LOOP     ...                              Decide whether you need to
                                          skip to another position
                                          (forward or backward).
         B       RETRIEVE                 No, bypass the POINT.
         MVC     ADDR,RBA value           Yes, move search argument for
                                          POINT into search-argument
                                          field.
         POINT   RPL=REQUEST              Position VSAM to the record
                                          to be retrieved next.
         LTR     15,15
         BNZ     ERROR
RETRIEVE GET     RPL=REQUEST
         LTR     15,15
         BNZ     ERROR
         .
         .
         .                                Decide whether to delete
                                          the record.
         BE      DELET
         B       LOOP                     No, skip ERASE.
DELET    ERASE   RPL=REQUEST              Yes, delete the record.
         LTR     15,15
         BNZ     ERROR
         B       LOOP
ERROR    ...                              Request was not accepted
         .                                or failed.
         .
         .
ADDR     DS      F                        RBA search argument for
                                          POINT.
WORK     DS      CL100                    Work area.
```

Discussion. Addressed deletion is allowed only for a key-sequenced file. The records of an entry-sequenced file are fixed, both in their existence and in their location.

Sharing Resources among Files

Normally, buffers and control blocks are allocated statically to a file at the time the file is opened; they are freed when the file is closed. As long as the file is open, these buffers and control blocks cannot be used by another file.

The Shared Resource facility, however allows you to share buffers, I/O control blocks, and channel programs among several VSAM files within a partition and manage I/O buffers. These buffers and control blocks are allocated out of a common *resource pool* at the time you issue an I/O request for a file. When the request is satisfied, the same buffers and control blocks can be assigned to another file (for direct requests).

Sharing these resources optimizes their use and also reduces the amount of virtual storage required (the working set) per partition. The facility is especially useful in an environment in which (a) many VSAM files are open and it is therefore difficult to predict the amount of activity that will occur at a given time, or(b) each transaction may refer to several files.

Managing I/O buffers includes:

- Deferring write operations for direct PUT requests, thus reducing the number of I/O operations
- Correlating deferred requests by a transaction ID
- Writing out buffers whose writing has been deferred

Managing I/O buffers should enable you to speed up direct processing of VSAM files whose activity is unpredictable.

When you share resources for sequential access, you have to establish positioning before you can issue your initial retrieval request, because with shared resources VSAM does not automatically position itself at the beginning of a file opened for sequential access. You establish positioning by issuing a POINT macro or a direct GET with the RPL operand OPTCD=NSP. Also note that you may not use shared resources to load records into an empty file.

The macros you use to share resources and write I/O buffers are:

- BLDVRP (build VSAM resource pool)
- DLVRP (delete VSAM resource pool)
- WRTBFR (write buffer)

In addition the SHOWCAT macro is provided to display, for non-open files, the catalog information needed for the proper specification of some of the BLDRVP operands.

The ACB, RPL, SHOWCB, MODCB, and TESTCB macros have been extended to provide for sharing resources and managing I/O buffers.

The use of these macros is described in this chapter. The operand notations and the parameter lists for these macros are described in Appendixes F and G, respectively.

Providing a Resource Pool

To share resources, you must:

- Issue the BLDVRP macro to build a resource pool
- Code the LSR option in the MACRF operand in the ACBs of your files, and
- Issue OPEN to connect these files to the resource pool.

Deciding How Big a Pool to Provide

You have to provide a resource pool before any clusters or alternate indexes are opened to use it. Specifying the BUFFERS, KEYLEN, and STRNO operands of the BLDVRP macro requires knowledge of the size of the control intervals, data records (if spanned), and key fields in the components that will use the resource pool and knowledge about the way the components are processed. (See "Format of the BLDVRP Macro" below.)

Displaying Information about an Unopened File

The SHOWCAT macro enables you to get information about a component before its cluster or alternate index is opened. The program that is to issue BLDVRP can issue SHOWCAT on all of the components to find out the sizes of control intervals, records, and keys. This information enables the program to calculate values for the BUFFERS and KEYLEN operands of BLDVRP.

Displaying Statistics about a Buffer Pool.

You can get statistics about the usage of buffer pools to determine how you could improve a previous definition of a resource pool and the mix of files that use it. The SHOWCB macro enables you to get statistics about a buffer pool. The statistics are available from an ACB that describes an open file that is using the buffer pool. They reflect the usage of the buffer pool from the time it was built to the time SHOWCB is issued. All of the statistics except one are for a single buffer pool. To get statistics for the whole resource pool, issue SHOWCB for each buffer pool in it.

The statistics cannot be used to redefine the resource pool while it is in use. You have to make adjustments the next time you build it.

BLDVRP: Building a Resource Pool

To share resources, you must provide a resource pool which you build by issuing the BLDVRP (build the VSAM resource pool) macro. Issuing BLDVRP causes VSAM to share the I/O buffers, I/O control blocks, and channel programs of those files whose ACBs indicate the local shared resource (LSR) option. (See "OPEN: Connecting a File to a Resource Pool," below.) Control blocks and channel programs are shared automatically; you can control the sharing of buffers.

When issuing BLDVRP, you must specify one or more *buffer pools* within the resource pool, and also the size and number of buffers in each buffer pool. A file uses the buffer pool whose buffer size exactly matches the file's control interval size or, if this size is not available, the buffer pool with the next-larger buffer size. The file uses only the one buffer pool.

Format of the BLDVRP Macro

When you issue a BLDVRP macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue a BLDVRP macro from within one of your exit routines such as LERAD or SYNAD, your program must provide a second 72-byte save area for use by VSAM, because the original save area is still in use by the external VSAM routine. The BLDVRP macro has the following format:

```
[name]  BLDVRP  BUFFERS=(size(number)
                ,size(number),...),
                STRNO=number
                [,KEYLEN=length]
                [,MF={L|(e,{address|(1)})}]
```

BUFFERS=(size(number),size(number),...)

specifies the size and the number of buffers in each buffer in the resource pool. The number of buffer pools in the resource pool is implied by the number of **size(number)** pairs you specify. The buffer sizes should normally be set equal to the control interval sizes of the files to be processed. (You can find out the control interval size of a file by issuing the SHOWCAT macro for that file.) If you do not specify the exact buffer(=control interval)size for a file, VSAM will use buffers from the buffer pool with the next larger buffer size.

When you process a key-sequenced file, the index component, as well as the data component, shares the buffers of a buffer pool. When you use an alternate index to process a base cluster, the components of the alternate index and the base cluster share buffers. The components of alternate indexes in an upgrade set share buffers. Buffers of the appropriate size and number must be provided for all of these components, each of which uses the buffer pool whose buffers are exactly the right size or the next-larger size.

size is 512, 1024, 2048, 4096, or so on in increments of 4096 to a maximum of 32K bytes.

number is at least 3.

KEYLEN=length

specifies the maximum key length of the files that are to share the resource pool. The default is 255.

The keys whose lengths must be provided for are the prime key of each key-sequenced file and the alternate key of each alternate index that is used for processing or is upgraded. The key length (relative record number) of a relative record file is 4. If the buffer pool is to contain for entry-sequenced files only, specify KEYLEN=0. (You can find out the key length of a file by issuing the SHOWCAT macro for that file.)

MF=L

indicates that this is the list form of BLDVRP. The list form builds a parameter list when the macro is assembled-it is not executable. If you do not specify STRNO in the list form of BLDVRP, you must specify it in the execute form.

The list form of the BLDVRP macro is especially useful when the buffer sizes of the VSAM files are not known. In that case you can first retrieve from the VSAM catalog the control interval sizes of the files to be processed via the SHOWCAT macro and then enter these values in the BSDVRP parameter list.

The format of the BLDVRP parameter list is described in Appendix G.

MF=(E,{address|(1)})

indicates that this is the execute form of BLDVRP. **address** is the address of the parameter list built by the list form of BLDVRP.

If you use register notation, you may use register 1, as well as any register from 2 through 12. The execute form puts the address of the parameter list in register 1 and passes control to VSAM to process the list. You may, however, first change the values for STRNO and/or KEYLEN (which are both optional in the execute form of BLDVRP). BUFFERS may not be specified in the execute form of BLDVRP, because this operand affects the length of the parameter list.

If the MF operand is omitted, the standard form of the BLDVRP macro is assumed, which builds a parameter list, puts its address in register 1, and passes control to VSAM to processs the list.

`STRNO=number`

specifies the maximum number of requests that may be issued concurrently for all of the files that are to share the resource pool. The number must be at least one and no more than 255.

If you want to find out how effectively your resource pool is utilized during execution, you can display the maximum number of requests which were concurrently active since the resource pool was built by issuing a SHOWCB ACB=...,FIELDS=(STRMAX) in your processing program. Depending on the result, you may want to redefine STRNO=*number* the next time you build your resource pool. (You cannot redefine the pool while it is in use.)

The ACB specified in the SHOWCB macro can be any ACB that describes an open file that is using the resource pool.

Return Codes from BLDVRP

When VSAM returns to your processing program after a BLDVRP request, register 15 contains one of the following return codes:

Return Code Dec	Hex	Meaning
0	X'00'	VSAM completed the request.
4	X'04'	A resource pool already exists in the partition.
12	X'0C'	The request was not executed bycause an error was encountered while VSAM routines were being loaded.
20	X'14'	STRNO is less than one or greater than 255.
24	X'18'	BUFFERS is specified incorrectly: size or number is invalid.

OPEN: Connecting a File to a Resource Pool

After having built a resouce pool you cause a file to use that pool by specifying LSR in the MACRF operand of the file's ACB before you open the file:

```
ACB   MACRF=(...[,LSR|,NSR]...)
```

NSR, the default, indicates that the file will not use shared resources.

When you have specified LSR in the ACB, VSAM ignores the BUFND, BUFNI, BUFSP, and STRNO operands, because it uses the BUFFERS and STRNO values that you have specified in the BLDVRP macro.

Restrictions: UBF (user buffering) may not be specified together with LSR. LSR may not be specified in the ACB of an empty file (which implies that the file is to be loaded).

Apart from the standard error codes from Open you may get additional error codes in the ACB ERROR field when you try to open a file with the LSR option. These error codes are listed in *VSE/VSAM Messages and Codes.*

DLVRP: Deleting a Resource Pool

After all the files using the resource pool have been closed, you must delete the resource pool by issuing the DLVRP (delete VSAM resource pool) macro.

If you do not delete the resource pool with the DLVRP macro, it will automatically be deleted at the end of the job step, because it resides in virtual storage, which is invalidated at the end of a job step.

When you issue a DLVRP macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue a DLVRP macro from within one of your exit routines such as LERAD or SYNAD, your program must rpovide a second 72-byte save area for use by VSAM, because the original save area is still in use by the external VSAM routine.

The DLVRP macro has the following format:

```
[name]  DLVRP
```

No operand is required with the DLVRP macro.

When VSAM returns to your processing program after a DLVRP request, register 15 contains one of the following return codes:

Return Code Dec	Hex	Meaning
0	X'00	VSAM completed the request.
4	X'04'	There is no resource pool to be deleted.
8	X'08'	There is at least one other open file using the resource pool.
12	X'0C'	The request was not executed because an error was encountered while VSAM routines were being loaded.

Managing I/O Buffers

Managing I/O buffers includes:

- Deferring writes for direct PUT requests, which reduces the number of I/O operations
- Writing buffers that have been modified by related requests.
- Writing out buffers whose writing has been deferred.

Deferring Write Requests

VSAM normally writes out the contents of a buffer immediately for direct PUT requests. With shared resources, however, you can cause write operations for direct PUT requests to be deferred. Buffers are finally written out:

- When you issue the WRTBFR macro.
- When VSAM needs a buffer to satisfy a GET request.
- When a file using a buffer pool is closed. (Temporary CLOSE is ineffective – a no-op – against a file that is sharing buffers; nor does ENDREQ cause buffers in a resource pool to be written.)

Deferring writes saves I/O operations in cases where subsequent requests can be satisfied by the data in the buffer pool. Data processing performance with VSAM will be improved if control intervals are updated more than once.

You indicate that writes are to be deferred by coding the MACRF DFR option in the ACB, along with MACRF=LSR:

```
ACB    MACRF=(LSR[,DFR|NDF]...)
```

NDF, the default, indicates that writes are not to be deferred for direct PUTs.

The DFR option is incompatible with SHAREOPTIONS(4). (SHAREOPTIONS is a parameter of the DEFINE command of Access Method Services.) A request to open a file with SHAREOPTIONS(4) for deferred writes is rejected.

Relating Deferred Requests by Transaction-ID

You can relate action requests (GET, PUT, etc.) according to transaction by specifying the same ID in the RPLs that define the requests.

The purpose of relating the requests that belong to a transaction is to enable WRTBFR to cause all of the modified buffers used for this transaction to be written out together. When the WRTBFR request is complete, the transaction is physically complete. To relate requests, specify:

```
RPL TRANSID=number,...
```

`TRANSID=number`

specifies a number from 0 to 31. A number from 1 to 31 relates the request(s) defined by this RPL to the requests defined by other RPLs with the same transaction ID. The number 0, which is the default, indicates that the request defined by this RPL is not associated with other requests.

You can find out what transaction ID an RPL has by issuing SHOWCB or TESTCB:

```
SHOWCB FIELDS=([TRANSID],...)
```

TRANSID requires one fullword in the display work area.

```
TESTCB TRANSID=number,...
```

You can also change the transaction ID of an RPL by issuing the MODCB macro:

```
MODCB TRANSID=number
```

WRTBFR: Writing Buffers Whose Writing Has Been Deferred

If DFR is specified in the ACB of any file that is using a resource pool, you can use the WRTBFR (write buffer) macro to write:

- All modified buffers for a given file
- All modified buffers in the resource pool
- The least recently used modified buffers in each buffer pool in the resource pool
- All buffers that have been modified by requests with the same transaction ID (see "Relating Deferred Requests by Transaction ID" above).

You can specify the DFR option in an ACB without using the WRTBFR to write buffers – a buffer will be written when VSAM needs one to satisfy a GET request, or all modified buffers will be written when the last of the files that uses them is closed.

Using WRTBFR can improve performance, if you schedule WRTBFR to overlap other processing.

VSAM notifies the processing program when there are no more unmodified buffers into which to read the contents of a control interval. (VSAM would be forced to write buffers when another GET request required an I/O operation.) VSAM sets register 15 to 0 and puts 12 (X'0C') in the feedback (FDBK) field of the RPL that defines the PUT request that detects the condition.

VSAM also notifies the processing program when there are no buffers available to process your request. This is a logical error (register 15 contains 8 unless an exit is taken to a LERAD routine); the feedback (FDBK) field in the RPL contains 152 (X'98'). You may retry the request; it will get a buffer if one has been freed.

In addition, VSAM will notify the processing program when the number of active requests exceeds the STRNO value specified in the BLDVRP macro (reg. 15=X'08', RPL FDBK=X'40').

Format of the WRTBFR Macro

When you issue a WRTBFR Macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue a WRTBFR Macro from within one of your exit routines such as LERAD or SYNAD, your program must provide a second 72-byte save area for use by VSAM, because the original save area is still in use by the external VSAM routine.

The format of the WRTBFR Macro is:

```
[name] WRTBFR     RPL=address,
                  TYPE={ALL|DS|LRU(percent)|TRN}
```

`RPL=address`

specifies the address of the request parameter list that defines the WRTBFR request. An RPL need not be built especially for the WRTBFR; WRTBFR may use an inactive RPL that defines other request(s) (GET, PUT, etc.) for a file that is using the resource pool.

Only the ACB and the TRANSID operands of the RPL are meaningful for WRTBFR; all other RPL operands are ignored. Unlike the other action macros (GET, PUT, etc.), WRTBFR assumes that RPLs are not chained.

`TYPE={ALL|DS|LRU(percent)|TRN}`

specifies what buffers are to be written.

ALL specifies that all modified buffers in each buffer pool in the resource pool are to be written. (Closing all of the files that use a resource pool has the same effect.)

DS specifies that, for the file defined by the ACB to which the WRTBFR's RPL is related all modified buffers are to be written.

LRU(percent) specifies the percentage of the total number of buffers in each buffer pool in the resource pool that are to be examined for possible writing. The least recently used buffers are examined. (If *percent* is coded in register notation, only registers 1 and 13 may not be used.)

When using the DFR option it is possible for the buffer pool to become saturated with modified buffers. VSAM would then be forced to write out a buffer before satisfying any other GET request. To ensure that buffers are always available for GET requests (without having to wait for buffers to be written) you can periodically force out the least recently used part of each buffer pool through the LRU option. To help determine when to do this, VSAM sets a non-error return code of 12 (X'0C') in the FDBK field of the RPL whenever it determines that the next GET request will have to wait for a PUT before it can be satisfied.

TRN specifies that all buffers in a buffer pool are to be written that have been modified by requests with the same transaction ID as the one specified in the WRTBFR's RPL. Transaction IDs are no longer associated with these buffers.

Preventing Deadlock in Exclusive Control

Contention for VSAM data (the contents of a control interval) can lead to deadlocks, in which a processing program is prevented from continuing because its request for data cannot be satisfied.

Your processing program gets exclusive control of a buffer (control interval) whenever you issue a GET for update (RPL option OPTCD=UPD) to retrieve a record from that buffer. You are responsible for preventing a deadlock by releasing as soon as possible the buffer for which another request may be waiting. Two requests, for example, A and B, may engage in four different contests:

1. A wants exclusive control, but B has exclusive control (OPTCD=UPD). VSAM refuses A's request – A must either do without the data or retry its request.
2. A wants exclusive control, but B has read-only access to the data (OPTCD=NUP). VSAM gives A a separate copy of the data.
3. A wants read-only access to the data (NUP), but B has exclusive control. VSAM refuses A's request – A must either do without the data or retry its request.
4. A has read-only access to the data, and B has read-only access. VSAM gives A a separate copy of the data.

VSAM's action in a contest for data rests on the assumptions that, if a processing program has exclusive control of the data (OPTCD=UPD), it will (or at least might) update or delete it and that, if a processing program is updating or deleting the data, it has exclusive control of it.

In contests 1 and 3, B is responsible for giving up exclusive control of a control interval by way of an ENDREQ or a request for access to a different control interval. (The RPL that defines the ENDREQ or request is the one that was used to acquire exclusive control in the first place.)

SHOWCAT Macro

The SHOWCAT (show, or display, the catalog) macro enables you to retrieve information from a catalog about any non-open file defined in the catalog.

The entries in a catalog are interrelated.

More than one entry is required to describe an object and its associated objects (for example, a cluster and its data and index components); one entry points to one or more other entries, which point to yet others. Figure 5-11 shows the interrelationship among entries that describe the following types of objects:

- Alternate index (G)
- Cluster (C)
- Data component (D)
- Index component (I)
- Path (R)
- Upgrade set (Y)

For example, an alternate-index entry points to the entries of its data and index components, its base cluster, and its path. SHOWCAT enables you to follow the arrows pictured in Figure 5-11. You first issue SHOWCAT by specifying the 44-byte name (file-ID) of filename of the object you want to display. The information VSAM returns to you (only if EXTOPT is not specified) includes the control-interval numbers of the catalog entries that describe any associated objects. You then issue subsequent SHOWCATs to

retrieve information from these associated entries by specifying the control-interval numbers that VSAM has returned to you. The first time you issue SHOWCAT, VSAM searches a specific catalog (according to the following order) to locate the entry that describes the object to be displayed:

1. The job catalog (IJSYSUC)
2. The master catalog (IJSYSCT), or you can direct VSAM to search a catalog other than the job or master catalog by specifying the CATDSN and optionally the CATFIL parameters of the SHOWCAT macro.

You must provide DLBL (and EXTENT) cards for:

- The master catalog – if the entry is in the master catalog
- The master and the job catalogs – if the entry is in the job catalog
- The master catalog – if the entry is in a user catalog.

VSAM returns to you the address of the ACB that defines the catalog containing the entry to be displayed. The subsequent times you issue SHOWCAT, you can specify that address, which causes VSAM to search only the corresponding catalog.

Format of the SHOWCAT Macro

When you issue a SHOWCAT macro, register 13 must contain the address of a 72-byte save area that you are providing. When you issue a SHOWCAT macro from within one of your exit routines, such as LERAD or SYNAD, your program must provide a second 72-byte save area for use by VSAM, because the original save area is still in use by the external VSAM routine.

The SHOWCAT macro has the following format:

```
[name]  SHOWCAT    {DDNAME|NAME|CI=address}
                   ,AREA=address
                   [,ACB=address]
                   [{,CATDSN=address
                   [,CATFIL=address]}]
                   [,EXTOPT={VOLSER|HARBADS}]
                   [,MF=L|,MF=({E|B}
                   ,{address|(1)})]
```

`{DDNAME|NAME|CI=address}`
specifies the address of an area that identifies the catalog entry that contains the desired information.

`DDNAME=address`
Specifies the address of a seven-byte area containing the *filename* of the object to be displayed. The object can be a cluster (C), an alternate index (G), or a path (R). Using the indicated filename, SHOWCAT first retrieves the corresponding name *(file-ID)* of the object from the label cylinder and then the desired information from the catalog.

Either this parameter or the NAME or CI parameter must be provided. However, when issuing the first SHOWCAT for an object, specify DDNAME or NAME. VSAM then supplies the control interval numbers of any associated objects for subsequent SHOWCATs (in the work area supplied through the AREA operand – see Figure 5-12)

`NAME=address`
specifies the address of a 44-byte area containing the name (file-ID) of the object to be displayed. The name is left-justified and padded with blanks

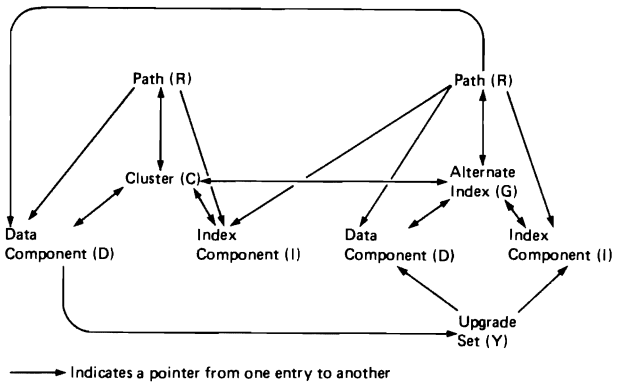


Figure 5-11. Interrelationship among Catalog Entries

on the right. The type of object named must be C, G, R, D, or I.

Either this parameter or the DDNAME or CI parameter must be specified. However, when issuing the first SHOWCAT for an object, specify DDNAME or NAME. VSAM then supplies the control interval numbers of any associated objects for subsequent SHOWCATs (in the work area supplied through the AREA operand – see Figure 5-12).

`CI=address`

specifies the address of a three-byte area that contains the control interval number of the catalog entry for the object to be displayed. The entry type of the object must be C, G, R, D, I, or Y. (Y can only be retrieved via CI).

Either this parameter or the DDNAME or NAME parameter must be specified. However, when you have already issued a SHOWCAT request for an object (with the DDNAME or NAME parameter), you then issue any subsequent SHOWCATs for its associated objects by specifying their control interval numbers (as returned to you via the previous SHOWCAT DDNAME or NAME request).

The three-byte area must be separate from the work area specified by the AREA operand, even though VSAM returns a control interval number in the work area.

`AREA=address`

specifies the address of a work area in which the catalog information is to be displayed. The first two bytes of this area must contain the length of the area, including these two length bytes.

The minimum size of the area is 64 bytes; (unless EXTOPT is specified (with EXTOPT the minimum size is 28 bytes); if it is smaller than the minimum size, you get a return code of 4 in register 15 and you can re-issue the SHOWCAT macro with a larger size. The format of the work area is described in Figure 5-12.

`ACB=address`

specifies the address of the ACB that defines the catalog containing the entry to be displayed. You issue the first SHOWCAT without ACB specified; VSAM searches the master catalog and, if provided, the job catalog or user catalog for the specified object and returns to you the address of the ACB that defines the containing catalog (in the work area supplied through the AREA operand – see Figure 5-12). When you subsequently issue SHOWCAT, you can specify that ACB address, which causes VSAM to go directly to the correct catalog without searching other

catalogs first. You should always include the ACB parameter when you specify CI instead of NAME.

`CATDSN=address[,CATFIL=address]`

CATDSN specifies the address of a 44-byte area containing the name (file-ID) of the catalog to be searched.

CATFIL specifies the address of an 8-byte area containing the filename of the catalog to be searched. File-ID and filename must be the same as those specified in the DLBL statement (if one is provided) for the catalog. CATDSN must always be specified if CATFIL is specified. CATFIL is always optional. You use these parameters to override the established order in which catalogs are searched. (VSAM always searches only one catalog for a specific entry.) That is, you must specify CATDSN if the object to be displayed is (1) not in the job or master catalog, or (2) in the master catalog and not the job catalog (in the case where filenames IJSYSUC and IJSYSCT are both specified).

`EXTOPT={VOLSER|HARBADS}`

indicates that either the volume serial number of the file's primary allocation volume (VOLSER) or the high allocated RBA for the file (HARBADS) is to be returned to you. This operand can only be issued for a D or I type catalog record.

The data returned for the EXTOPT operand replaces the associated object information in the user return area. If you need the associated object information as well as the EXTOPT data, you must issue separate SHOWCAT macros.

`MF=L`

specifies that the list form of the SHOWCAT macro is required. The list form builds a parameter list when the macro is assembled; it is not executable. AREA and {DDNAME|CI|NAME} are optional in the list form; if you do not specify them in the list form they must be specified in the execute form. In the list form, the operand addresses cannot be expressed in register notation. The format of the SHOWCAT parameter list is described in Appendix G.

`MF=({E|B},{address|(1)})`

specifies that the execute form of the SHOWCAT macro is required.

E indicates that the parameter list, whose address is given in **address** or in a register, is to be passed to VSAM for processing.

B indicates that the parameter list is to be built or modified, but is not to be passed to VSAM. This form of the macro is similar to the list form, except that it works at execution time and can modify a parameter list, as well as build it.

To build a parameter list, first issue SHOWCAT with only MF=(B,...) specified, to zero out the area in which it will be built.

address gives the address of the parameter list. If you use register notation, you may use register 1, as well as a register from 2 through 12. Register 1 is used to pass the parameter list to VSAM (if MF=E).

If the MF operand is omitted, the standard form of the SHOWCAT macro is assumed, which builds the parameter list, puts its address in register 1, and passes control to VSAM to process the list.

Return Codes from SHOWCAT

When VSAM returns to your processing program after a SHOWCAT request, register 15 contains one of the following return codes:

Return Code	Meaning
0 (0)	VSAM completed the request.
4 (4)	The area specified in the AREA operand is less than the minimum required (64 bytes) or is too small to display all associated objects (as many objects as possible are displayed).
8 (8)	Either the ACB address is invalid or the VSAM master catalog does not exist or could not be opened.
12 (0C)	The request was not executed because an error was encountered while VSAM routines were being loaded (see Note).
20 (14)	The named object or control interval does not exist (see Note).
24 (18)	An I/O error occurred in gaining access to the catalog (see Note).
28 (1C)	The specified CI number is invalid.
32 (20)	The specified object is not a C, D, G, I, R, or Y type of object (see Note).
36 (24)	The information in the catalog is at a different level than that in the CRA.
40 (28)	An unexpected error code was returned from catalog management to the SHOWCAT processor (see Note).
44 (2C)	An error occurred in searching the label area for the file-ID corresponding to the specified filename (see Note).
48 (3)	EXTOPT field name is not valid for SHOWCAT.
52 (34)	EXTOPT specified, but record type not D or I.

If a return code of 0 was passed in register 15, the requested catalog information is returned in the work area which you have supplied through the AREA operand and whose format is shown in Figure 5-12.

Note: In case the SHOWCAT return code in register 15 is 12, 20, 24, 28, 36, 40, 44, or 52, the work area contains the return code and reason code issued by VSAM catalog management as well as the module ID of the catalog management module in which the error was detected. The format of the work area is then as follows:

Offset	Length	Description
0	2	Length of work area
2	2	VSAM catalog return code* or (for return code 44) VSAM error code
4	2	VSAM catalog reason code*
6	2	VSAM catalog management module ID

* For an explanation of the codes, see "Appendix 2: Access Method Services and Return Codes" in *VSE/VSAM Messages and Codes.*

Offset	Length	Description
0(0)	2	Length of the work area, including the length of this field (provided by you).
2(2)	2	Length of the work area actually used by VSAM, including the length of this field and the preceeding field.
4(4)	4	The address of the ACB that defined the catalog that contains the entry which is to be displayed.
8(8)	1	Type of object about which information is returned: C Cluster D Data Component G Alternate index I Index R Path Y Upgrade set The following fields contains one set of information for C, G, R, and Y types and another set for D and I types:
For C, G, R, and Y types:		
9(9)	1	For C and Y types: reserved For G type: x... .... The alternate index might (1) or might not (0) be a member of an upgrade set. The way to find out for sure is to display information for the upgrade set of the base cluster and check whether it contains control interval numbers of entries that describe the components of the alternate index. Figure 5-11 shows you how to get from the alternate index's catalog entry to the entries that describe its components (G to C to D to Y to D and I). .xxx xxxx Reserved. For R type: x... .... The path is (1) or is not (0) defined with the UPDATE attribute (for upgrading alternate indexes). .xxx xxxx Reserved.
10(A)	2	The number of pairs of fields that follow. Each pair of fields identifies another catalog entry that describes an object associated with this C, G, R, or Y object. The possible types of associated objects are: With C: D, G, I, R. With G: C, D, I, R. With R: C, D, G, I. With Y: D, I. Figure 5-11 shows how the catalog entries for all these objects are interrelated.
12(C)	1	Type of associtated object the entry describes.
13(D)	3	The control-interval number of its first record.
16(10)		Next pair of fields, and so on. If the area is too small to display a pair of fields for each associated object, VSAM displays as many pairs as possible and returns a code of 4 in register 15. Each pair of fields occupies 4 bytes, except Y-type entries which require 8 bytes (4 for the data component and 4 for the index component of the alternate index in the upgrade set).
For D and I types:		
9(9)	1	Reserved.
10(A)	2	Relative position of the prime key in records in the data component. For the data component of an entry-sequenced or a relative record file there is no prime key, and this field is 0.
12(C)	2	Length of the prime key.
14(E)	4	Control-interval size of the data or index component.
18(12)	4	Maximum record size of the data or index component.
22(16)	2	The number of pairs for fields that follow. Each pair of fields identifies another catalog entry that describes an object associated with this D or I object. The possible types of associated objects are: With D: C, G, Y. With I: C, G. Figure 5-11 shows how the catalog entries for all these objects are interrelated.
24(18)	1	Type of associated object the entry describes.
25(19)	3	The control-interval number of its first record.
28(1C)		Next pair of fields, and so on. Fields for all associated objects can always be displayed (with the minimum AREA size specified).

Figure 5-12. Format of SHOWCAT Work Area

Appendix A: Sample Job Streams

The following sample job streams demonstrate many of the functions provided by Access Method Services. They are intended to be used only as a guide for the coding of your own job streams. They are not intended to be coded exactly as shown and used by an installation because the user data that is necessary to run the jobs is not provided herein.

Note: No DLBL statement for master catalog AMASTCAT appears in the following examples because it is assumed that the following required DLBL statement (that is `// DLBL IJSYSCT,'AMASTCAT',,VSAM`) has been placed in the permanent label area.

Example 1: Define a System's Catalogs

Before running jobs or, in the following cases, examples, the master catalog must be defined. In this example, the master catalog and one user catalog are defined. Both catalogs are password protected. The master catalog is defined with the RECOVERABLE attribute; the user catalog is not. The data space for each catalog is defined to be 15 cylinders. The amount of space explicitly specified for the data and index components of each catalog is taken from its respective catalog data space. Since the data space for both the data and index components is less than the total catalog data space, the remaining catalog data space is available for suballocation. The master catalog resides on volume VSER01 and the user catalog on volume VSER02. These volumes cannot be referenced by other VSAM catalogs, since they now belong to the catalog which they contain.

In the DEFINE command for D27UCAT1, a data space of 15 cylinders is defined and assigned to class 7 data space. The user catalog is immediately suballocated 4 of the cylinders in this class 7 space; the remaining 3 cylinders of class 7 space is available for later suballocation.

```
// DLBL     IJSYSCT,'AMASTCAT',,VSAM
// JOB      EXAMPL01
// ASSGN    SYS005,231
// DLBL     VOL4,'D27UCAT1',,VSAM
// ASSGN    SYS005,VSER02,,,20,300
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   MCAT -
            ( NAME(AMASTCAT) -
              MRPW(MCATMRPW) -
              UPDPW(MCATUPPW) -
              RDPW(MCATRDPW) -
              TO(99999) -
              ATT(3) -
              CODE(BEQUIET) -
              RECOVERABLE -
              FILE(IJSYSCT) -
              TRK(300) -
              ORIGIN(20) -
              VOL(VSER01) -
            ) -
            DATA -
            ( CYL(5,1)) -
            INDEX -
            ( CYL(3))
```

```
IF LASTCC = 0 -
    THEN -
DEFINE  UCAT -
        ( NAME(D27UCAT1) -
          MRPW(UCATMRPW) -
          UPDPW(UCATUPPW) -
          RDPW(UCATRDPW) -
          FOR(365) -
          (FILE(VOL2) -
          CYL(15) -
          VOL(VSER02) -
          CLASS(7) -
        ) -
        DATA -
        ( CYL(3,1)) -
        INDEX -
        ( CYL(1)) -
        CATALOG(AMASTCAT/MCATUPPW)
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The VOL2 DLBL statement describes the volume on which the user catalog is to be defined.

The DEFINE MCAT FILE parameter is required and names the *filename* of the DLBL statement describing the master catalog. The *filename* of the DLBL statement for the master catalog must be the same as this name.

The DEFINE UCAT FILE parameter is required and names the *filename* of the DLBL statement describing the user catalog.

Explanation of Commands

The first DEFINE command defines the master catalog for the system.

1. The MCAT parameter is required and NAME specifies the master catalog being defined.
2. The MRPW parameter specifies the master password of this catalog.
3. The UPDPW parameter specifies the update password of this catalog.
4. The RDPW parameter specifies the read password of this catalog.
5. The TO parameter specifies the maximum retention time.
6. The ATT parameter specifies the number of allowable attempts for catalog password prompting.
7. The CODE parameter specifies the code for password prompting.
8. The RECOVERABLE parameter specifies that all catalog entries are to be duplicated in a catalog recovery area. Note that one cylinder must be available in the catalog's data space for the allocation of the CRA.
9. The TRK parameter specifies the amount of space to be allocated to the catalog's data space. A space parameter is required.
10. The ORIGIN parameter specifies the beginning track number.

11. The VOL parameter is required and specifies the volume containing this catalog.
12. The DATA parameter specifies the amount of space to be allocated to the catalog's data component.
13. The INDEX parameter specifies the amount of space to be allocated to the catalog's index component. VSAM adds together the amount of space specified via the DATA and INDEX parameters and determines the appropriate proportions for each component.

If the definition of the master catalog was successful, a second DEFINE command defines a user catalog that resides on volume VSER02.

1. The UCAT parameter is required and NAME specifies the user catalog being defined.
2. The MRPW parameter specifies the master password for this catalog.
3. The UPDPW parameter specifies the update password of this catalog.
4. The RDPW parameter specifies the read password of this catalog.
5. The FOR parameter specifies the retention period for this file—in this case, one year.
6. The CYL parameter specifies the amount of space to be allocated to the data space being defined. A space parameter is required.
7. The VOL parameter is required and specifies the volume containing this catalog.
8. The CLASS parameter assigns the data space (specified in the CYL parameter) to class 7. The user catalog is suballocated from this space
9. The DATA parameter specifies the amount of space to be allocated to the catalog's data component.
10. The INDEX parameter specifies the amount of space to be allocated to the catalog's index component. VSAM adds together the amount of space specified via the DATA and INDEX parameters and determines the appropriate proportions for each component.
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.

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 DEFINE command defines VSAM data space on a volume owned by the master catalog. This allows files to be suballocated onto this volume.

This example can be viewed as a logical follow-on to the previous example.

```
// JOB      EXAMPL02
// ASSGN    SYS005,231
// ASSGN    SYS006,232
// ASSGN    SYS007,233
// DLBL     UCAT1,'D27UCAT1',,VSAM
// EXTENT   SYS005,VSER02
// DLBL     VOL3,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03,1,0,20,300
// DLBL     VOL4,,,VSAM
// EXTENT   SYS007,VSER04,1,0,60,300
// DLBL     VOL5,,,VSAM
// EXTENT   SYS007,VSER04,1,0,20,40
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   UCAT -
            ( NAME(D27UCAT2) -
              MODEL(D27UCAT1/UCATMRPW D27UCAT1 UCAT1) -
              FILE(VOL3) -
              ORIGIN(20) -
              VOL(VSER03) -
              CYL(15,5) -
              CLASS(0) -
              NOIMBED -
            ) -
            CATALOG(AMASTCAT/MCATUPPW)
     DEFINE  SPACE -
            ( ORIGIN(20) -
              FILE(VOL4) -
              VOL(VSER04) -
              TRK(40) -
              CLASS(1) -
            ) -
            CATALOG(AMASTCAT/MCATUPPW)
     DEFINE  SPACE  -
             ( ORIGIN(60) -
               FILE(VOL4) -
               VOL(VSER04) -
               CYL(10,1) -
             ) -
            CATALOG(AMASTCAT/MCATUPPW)

/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The UCAT1 DLBL statement describes the user catalog defined in Example 1. This enables using D27UCAT1 as a model for the new user catalog named D27UCAT2 when explicitly referenced. Since an IJSYSUC job catalog DLBL statement is not coded, all other catalog references in the example are to the master catalog.

The VOL3 DLBL statement describes the volume on which the new user catalog is to be defined.

The VOL4 DLBL statement is required for the first space definition (default to class 0) on volume VSER04.

The VOL5 DLBL statement is required for the second space definition, class 1 on volume VSER04.

The DEFINE UCAT MODEL parameter specifies the name of the catalog to be modeled and the master password of that catalog so as to enable the

catalog's security attributes to be modeled. The catalog will be opened in response to the UCAT1 *dname*.

The DEFINE UCAT FILE parameter is required and names the *filename* of the DLBL statement describing this user catalog being defined.

The FILE parameter in both DEFINE SPACE commands is required and names the *filename* of the DLBL statement that describes the volume in which space is to be defined.

Explanation of Commands

The first DEFINE command defines a new user catalog modeled with the same self-describing attributes of the user catalog defined in Example 1. (The IMBED attribute of the model is overridden.) The VOL and space (CYL) parameters are required even though modeling is performed.

1. The UCAT parameter is required and NAME specifies the user catalog being defined.
2. The MODEL parameter specifies the name of the catalog to be modeled and the master password of that catalog so as to enable the catalog's security attributes to be modeled.
3. The ORIGIN parameter indicates 20 is the beginning track number.
4. The VOL parameter is required and specifies the volume containing this catalog.
5. The CYL parameter specifies the amount of space to be allocated to the catalog's data space. A space parameter is required.
6. The CLASS parameter specifies the space class for the new user catalog. The useclass for the catalog is always the same as the class specified (in this case, 0), overriding the useclass 7 of the model. If the class were not explicitly specified, the model useclass would be used for both the class of the space defined and the useclass of the catalog.
7. The NOIMBED parameter overrides the imbed attribute of the model and specifies that the index sequence set is not to be imbedded in the catalog's data component.
8. 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 VSAM data space on volume VSER04 and establishes volume ownership by the master catalog, AMASTCAT. All subsequent VSAM objects are required to be cataloged in the volume's catalog, AMASTCAT. The space is to be assigned to class 1. Class 1 space is normally assigned to the fixed head area of a direct-access storage device, although this is not an enforced requirement. In this example, the device type is not specifically illustrated, so no particular attempt has been made to align the class 1 with the fixed head area.

1. The SPACE parameter is required.
2. The ORIGIN parameter indicates that 20 is the beginning track number.
3. The VOL parameter is required and specifies the volume containing this data space.
4. The TRK parameter specifies the amount of space to be allocated on this volume. A space parameter is required.

5. The CLASS parameter specifies that the space is to be assigned to class 1.

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.

The third DEFINE command defines a second VSAM space on VSER04, making possible the suballocation of files onto this volume. The space is assigned to class 0 by default.

1. The SPACE parameter is required.

2. The ORIGIN parameter indicates that 60 is the beginning track number.

3. The VOL parameter is required and specifies the volume containing this data space.

4. The CYL parameter specifies the amount of space to be allocated on the volume. The space is assigned to class 0 by default. A space parameter is required.

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.

Example 3: Define VSAM Files

This example defines a unique key-sequenced file into the master catalog, a suballocated relative-record file into the master catalog, and a suballocated entry-sequenced file into the user catalog D27UCAT2. Execution of the LISTCAT commands is dependent upon successful definition of the specified file in the preceding DEFINE command.

Example 3 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL03
// ASSGN    SYS006,232
// ASSGN    SYS007,233
// DLBL     IJSYSUC,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03
// DLBL     DFILE,,,VSAM
// EXTENT   SYS007,VSER04,1,0,220,40
// DLBL     XFILE,,,VSAM
// EXTENT   SYS007,VSER04,1,0,260,20
// DLBL     VOL4,,,VSAM
// EXTENT   SYS007,VSER04
```

```
// EXEC    IDCAMS,SIZE=AUTO
   DEFINE  CLUSTER -
           ( NAME(EXAMPLE.KSDS1) -
             RDPW(KSD1PSWD) -
             FOR(365) -
             VOL(VSER04) -
             BUFSP(12288) -
             RECORDSIZE(400,475) -
             KEYS(12 4) -
             FREESPACE(40,40) -
           ) -
           DATA -
           ( NAME(EXAMPLE.KSDS1.DATA) -
             UNIQUE -
             CYLINDERS(2 1) -
             FILE(DFILE) -
           ) -
           INDEX -
           ( NAME(EXAMPLE.KSDS1.INDEX) -
             UNIQUE -
             FILE(XFILE) -
             CYLINDERS(1 1) -
           ) -
           CATALOG(AMASTCAT/MCATUPPW)
   IF LASTCC = 0 -
       THEN -
   LISTC   ENTRIES(EXAMPLE.KSDS1/KSD1PSWD) ALL -
           CATALOG(AMASTCAT)
   DEFINE  CLUSTER -
           ( NAME(EXAMPLE.RRDS1) -
             VOL(VSER04) -
             TRK(10 5) -
             RECORDSIZE(100 100) -
             FILE(VOL4) -
             NUMBERED -
           ) -
           CATALOG(AMASTCAT/MCATUPPW)
   IF LASTCC=0 -
       THEN -
   LISTCAT ENTRIES(EXAMPLE.RRDS1) CLUSTER ALL -
           CATALOG(AMASTCAT)
   DEFINE  CLUSTER -
           ( NAME(EXAMPLE.ESDS1) -
             MRPW(ESD1MRPW) -
             UPDPW(ESD1UPPW) -
             VOL(VSER03) -
             SPANNED -
             REUSE -
             NONINDEXED -
             CYL(1 1) -
             RECORDSIZE(2500 3000) -
           ) -
           CATALOG(D27UCAT2/UCATMRPW)
   IF LASTCC = 0 -
       THEN -
   LISTC   ENTRIES(EXAMPLE.ESDS1) ALLOC
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The IJSYSUC DLBL statment describes the user catalog into which the entry-sequenced file is to be defined. IJSYSUC is used to designate a user catalog as a job catalog. All references will be to the job catalog unless otherwise directed.

The DFILE and XFILE DLBL statements describe the data components of the data and index components, respectively, of the unique key-sequenced file being defined and their corresponding EXTENT statements specify the area of disk allocation to be assigned to each component.

The VOL4 DLBL statement identifies the volume containing the catalog recovery area for the relative-record file.

The FILE parameter is required in the second DEFINE command since the file is being defined in a recoverable catalog. It names the *filename* of the DLBL statement describing the volume containing the catalog recovery area.

Explanation of Job Control Statements

1. The IJSYSUC DLBL statement describes the user catalog into which the entry-sequenced file is to be defined. IJSYSUC is used to designate a user catalog as a job catalog. All references will be to the job catalog unless otherwise directed.
2. The DFILE and XFILE DLBL statements describe the data components of the data and index components, respectively, of the unique key-sequenced file being defined and their corresponding EXTENT statements specify the area of disk allocation to be assigned to each component. Note that the EXTENT statements do not require a symbolic unit specification.

Explanation of Commands

The first DEFINE command defines a unique key-sequenced file on volume VSER04.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The RDPW parameter specifies the read password of this cluster. Since no master password is defined, this password will be propagated up to the master level.
3. The FOR parameter specifies the retention period for this file—in this case, one year.
4. The VOL parameter is required and specifies the volume containing this file.
5. The BUFSP parameter is specified for increased performance.
6. The RECORDSIZE parameter specifies the average and maximum record sizes.
7. The KEYS parameter specifies the key length and offset from the beginning of the record.
8. The FREESPACE parameter specifies the percentage of space within control intervals and control areas, respectively, that is originally free.
9. 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.

10. The UNIQUE parameter specifies that this portion of the file is the only one that occupies the data space allocated to it.
11. A space, in this example CYLINDERS, parameter is required for a unique file and the amount of space specified to be allocated to the data component must match the space parameters in the corresponding EXTENT statement. The secondary allocation quantity is ignored.
12. The FILE parameter is required for a unique file and names the *filename* of the DLBL statement for the data component.
13. 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.
14. The UNIQUE parameter specifies that this portion of the file is the only one that occupies the data space allocated to it.
15. The FILE parameter is required for a unique file and names the *filename* of the DLBL statement for the index component. The DLBL statement also provides access to the catalog recovery area on VSER04.
16. A space, in this example CYLINDERS, parameter is required for a unique file and the amount of space specified to be allocated to the index component must match the space parameters in the corresponding EXTENT statement. The secondary allocation quantity is ignored.
17. 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 DLBL statement for the job catalog (IJSYSUC) appears in the job control and this define must therefore be explicitly directed to the master catalog.

If the define of the unique key-sequenced file was successful, then the following LISTCAT command is executed.

1. The ENTRIES and ALL parameters cause the entire catalog description of the file just defined to be listed.
2. The CATALOG parameter directs the LISTCAT to the master catalog.

The second DEFINE command defines a suballocated relative-record file into the VSAM class 0 data space on volume VSER04 which is owned by the master catalog. Class 0 is selected by default because of the absence of the USECLASS parameter.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The VOL parameter is required and specifies the volume containing this file.
3. The TRK parameter specifies the amount of space allocated to this file. A space parameter is required.
4. The RECORDSIZE parameter specifies the average and maximum record sizes which, in the case of a relative-record file, must be equal.
5. The NUMBERED parameter is required to override the default (INDEXED).
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. This parameter is also required because a DLBL statement for the job

catalog (IJSYSUC) appears in the job control and this define is directed to the master catalog.

If the define of the suballocated relative-record file 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 object.
2. The CATALOG parameter directs the LISTCAT to the master catalog.

The third DEFINE command defines a suballocated entry-sequenced file into the job catalog (IJSYSUC). Note that no read password was specified so that the cluster can be read without a password.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The MRPW parameter specifies the master password of this cluster.
3. The UPDPW parameter specifies the update password of this cluster.
4. The VOL parameter is required and specifies the volume containing this file.
5. The SPANNED parameter specifies that records may span control interval boundaries.
6. The REUSE parameter specifies that the file can be reused, that is, reloaded without being deleted and redefined.
7. The NONINDEXED parameter is required to override the default (INDEXED).
8. The CYL parameter specifies the amount of space to be allocated to this file. Space is allocated from class 0 space because of the absence of the USECLASS parameter. A space parameter is required.
9. The RECORDSIZE parameter specifies the average and maximum record sizes.
10. The CATALOG parameter specifies the *file-ID* of the job catalog. It is required only because the catalog is password protected and its update password is required for the define.

If the define of the suballocated entry-sequenced file was successful, then the following LISTCAT command is executed.

1. The ENTRIES and ALLOC parameters cause the file entry just defined to be listed, limited, however, to only volume and allocation information.
2. The absence of a CATALOG parameter implicitly directs the LISTCAT to the job catalog (IJSYSUC).

Example 4: Define NonVSAM and VSAM Files

This example defines a nonVSAM file into a user catalog, and VSAM key-sequenced and entry-sequenced files into the master catalog. Attributes for the entry-sequenced file are modeled from those of the entry-sequenced file defined in Example 3.

Example 4 can be viewed as a logical follow-on to the previous examples.

```
// JOB     EXAMPL04
// ASSGN   SYS005,231
// ASSGN   SYS006,232
// ASSGN   SYS007,233
// DLBL    IJSYSUC,'D27UCAT2',,VSAM
// EXTENT  SYS006,VSER03
// DLBL    UCAT1,'D27UCAT',,VSAM
// EXTENT  SYS005,VSER02
// DLBL    VOL4,,,VSAM
// EXTENT  SYS007,VSER04
// EXEC    IDCAMS,SIZE=AUTO
   DEFINE  NONVSAM -
           ( NAME(EXAMPLE.NONVSAM1) -
             VOL(231401) -
             DEVT(2314) -
           ) -
           CATALOG(D27UCAT1/UCATUPPW UCAT1)
   IF LASTCC = 0 -
       THEN -
   LISTCAT NONVSAM ALL -
           CATALOG(D27UCAT1 UCAT1)
   DEFINE  CLUSTER -
           ( NAME(EXAMPLE.KSDS2) -
             FILE(VOL4) -
             MRPW(KSD2MRPW) -
             UPDPW(KSD2UPPW) -
             RDPW(KSD2RDPW) -
           ) -
           DATA -
           ( NAME(EXAMPLE.KSDS2.DATA)  -
             RECORDS(500 100) -
             USECLASS(0 0) -
             EXCEPTIONEXIT(DATEXIT) -
             ERASE -
             FREESPACE(20 10) -
             KEYS(6 4) -
             RECORDSIZE(80 100) -
             ORDERED -
             VOL(VSER04 VSER01) -
           ) -
           INDEX -
           ( NAME(EXAMPLE.KSDS2.INDEX) -
             USECLASS(1 P) -
             IMBED -
             VOL(VSER04) -
           ) -
           CATALOG(AMASTCAT/MCATUPPW)
   IF LASTCC = 0 -
       THEN -
   LISTCAT DATA ALL ENT(EXAMPLE.KSDS2/KSD2MRPW) -
           CATALOG(AMASTCAT)
   DEFINE  CLUSTER -
           ( NAME(EXAMPLE.ESDS2) -
             MODEL(EXAMPLE.ESDS1/ESD1MRPW) -
             VOLUMES(VSER04) -
             FILE(VOL4) -
             USECLASS(1 0) -
             MRPW(ESD2MRPW) -
             CTLPW(ESD2CTPW) -
             UPDPW(ESD2UPPW) -
             RDPW(ESD2RDPW) -
           ) -
           CATALOG(AMASTCAT/MCATUPPW)
```

```
   IF LASTCC = 0 -
       THEN -
       DO
   LISTC    ENTRIES(EXAMPLE.ESDS2/ESD2MRPW) ALL -
            CATALOG(AMASTCAT)
   LISTC    NAME CATALOG(AMASTCAT)
       END
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The UCAT1 DLBL statement describes the user catalog in which the nonVSAM entry is to be defined. This catalog can only be referenced by a command which explicitly names it.

The VOL4 DLBL statement identifes the volume containing the catalog recovery area for both the key-sequenced and entry-sequenced files.

The CATALOG parameter specifying the name of the user catalog is required because job catalog also appears in the job control, so the parameter must explicitly direct the define to the user catalog. The *dname* UCAT1 will cause the catalog to be opened.

The FILE parameter in both DEFINE CLUSTER commands is required since the catalog is recoverable and names the *filename* of the DLBL statement describing the volume containing the catalog recovery area for the cluster.

Explanation of Job Control Statements

1. The IJSYSUC DLBL statement describes the user catalog D27UCAT2 as a job catalog. It contains the model to be used when defining the entry-sequenced file in this example. All references will be to this job catalog unless otherwise directed.

Explanation of Commands

The first DEFINE command defines an existing nonVSAM file into user catalog D27UCAT1.

1. The NONVSAM parameter is required and NAME specifies the nonVSAM object being defined.
2. The VOL parameter is required and specifies the volume containing the file.
3. The DEVT 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 so the parameter must explicitly direct the define to the user catalog.
   - The catalog is protected and its update password is required for the define.

If the define 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 and opens that catalog.

The second DEFINE command defines a key-sequenced file into VSAM data space owned by the master catalog. Note that attributes are specified at the data and index level rather than the cluster level.

1. The CLUSTER parameter is required and NAME specifies the name of the cluster being defined.
2. The MRPW, UPDW, and RDPW parameters specify the master, update, and read passwords, respectively, of the cluster.
3. The DATA component is explicitly named via the NAME parameter; it will not have a name generated for it by VSAM.
4. The RECORDS parameter specifies the amount of space to be allocated to the data component. Note that no space allocation parameter is given for the index component. VSAM will calculate a suitable value based on the data component specification.
5. The USECLASS parameter specifies that the data component is to be allocated from class 0 data space.
6. The EXCEPTIONEXIT parameter specifies the name of the routine to be given control if an exception occurs while processing the data component.
7. The ERASE parameter specifies that the data component is to be overwritten with binary zeros when it is deleted.
8. The FREESPACE parameter specifies the percentage of space within control intervals and control areas, respectively, that is originally free.
9. The KEYS parameter specifies the key length and offset.
10. The RECORDSIZE parameter specifies the average and maximum record sizes.
11. The ORDERED parameter specifies that VSER04 is to be used first (Primary). Volume VSER01 is to be used only for secondary extensions when class 0 space is not available on VSER04.
12. The VOL parameter specifies the volume (VSER04) containing this data component and the volume used for extending the file (VSER01).
13. The INDEX component is explicitly named via the NAME parameter; it will not have a name generated for it by VSAM.
14. The USECLASS parameter specifies that the index component is to be allocated from class 1 data space and secondary extensions are also to be allocated from class 1. (Class 1 space was established on volume VSER04 in Example 2.)
15. The IMBED parameter specifies that the index sequence set is to be placed with the data component.
16. The VOL parameter specifies the volume containing this index component.
17. 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 DLBL statement for the job catalog (IJSYSUC) appears in the job control and this define must, therefore, be explicitly directed to the master catalog.

If the define of the key-sequenced file was successful, then the following LISTCAT command is executed.

1. The DATA, ALL, and ENT 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 file that uses the entry-sequenced file (EXAMPLE.ESDS1) defined in Example 3 as a model. The protection attributes of the model, the containing volume, and the class of space to be used are, however, overridden by explicit specification. The define is directed to the master catalog.

1. The CLUSTER parameter is required and NAME specifies the cluster being defined.
2. The MODEL parameter specifies the name of the file to be used as a model. Note that the new entry-sequenced file is being defined into the master catalog, but the entry-sequenced file to be used as a model is defined in job catalog D27UCAT2.
3. The VOLUMES parameter specifies a volume (VSER04) different from the volume (VSER03) which contains the file being modeled.
4. The USECLASS parameter overrides the model useclass specification of 0 and specifies that the primary allocation for the file is to be taken from class 1 space on volume VSER04. However, secondary extensions are to be allocated from class 0 space.
5. The MRPW, CTLPW, UPDPW, and RDPW parameters specify passwords different from the passwords specified for the file being modeled.
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. This parameter is also required because a DLBL statement for the job catalog (IJSYSUC) appears in the job control and this define must, therefore, be explicitly directed to the master catalog.

If the define of the entry-sequenced file was successful, then the following LISTCAT commands are executed.

1. The ENTRIES and ALL parameters of the first LISTC command cause all the cataloged information in the entry just defined to be listed.
2. The CATALOG parameter directs the LISTCAT to the master catalog.
3. The NAME parameter of the second LISTC command causes only the names of the objects cataloged in the master catalog to be listed.
4. The CATALOG parameter directs the LISTCAT to the master catalog.

Example 5: Loading and Printing Files

This example shows various techniques which can be used to load and print files using the REPRO and PRINT commands.

Example 5 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL05
// ASSGN    SYS013,234
// ASSGN    SYS007,233
// ASSGN    SYS006,232
*      NONVSAM INDEXED SEQUENTIAL FIXED FILE
// DLBL     INDSET1,'BADVA101.TEST.ISAM1',,ISE
// EXTENT   SYS013,231401,4,0,020,20
// EXTENT   SYS013,231401,4,1,040,20
// EXTENT   SYS013,231401,1,2,060,20
*      NONVSAM VARIABLE SAM FILE
// DLBL     INDSET2,'BADVA101.TEST.SAM2'
// EXTENT   SYS013,231401,1,0,122,2
*      NONVSAM FIXED SAM FILE
// DLBL     INDSET3,'BADVA101.TEST.SAM1'
// EXTENT   SYS013,231401,1,0,120,2
*      NONVSAM FIXED SAM FILE
// DLBL     INDSET4,'EXAMPLE.NONVSAM1'
// EXTENT   SYS013,231401,1,0,124,14
*      VSAM FILES
// DLBL     VSDSET1,'EXAMPLE.KSDS1',,VSAM
// EXTENT   SYS007,VSER04
// DLBL     VSDSET2,'EXAMPLE.ESDS1',,VSAM,CAT=UCAT2
// EXTENT   SYS006,VSER03
// DLBL     VSDSET3,'EXAMPLE.RRDS1',,VSAM
// EXTENT   SYS007,VSER04
// DLBL     VSDSET4,'EXAMPLE.KSDS2',,VSAM
// EXTENT   SYS007,VSER04
// DLBL     UCAT2,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER04
// EXEC     IDCAMS,SIZE=AUTO
   /* LOAD A VSAM KEY-SEQUENCED FILE */
   /* FROM AN ISAM FILE */
   REPRO    INFILE(INDSET1,ENV(RECFM(F), -
              BLKSZ(100),RECSZ(100))) -
            OUTFILE(VSDSET1/KSD1PSWD)
   IF MAXCC = 0 -
      THEN -
      PRINT  INFILE(VSDSET1/KSD1PSWD) -
             FROMKEY(ABC)
   /* LOAD A VSAM ENTRY-SEQUENCED FILE FROM AN */
   /* EXISTING VARIABLE UNBLOCKED SAM FILE */
   REPRO      INFILE(INDSET2, ENV(RECFM(V), -
                RECSZ(132),BLKSZ(136))) -
              OUTFILE(VSDSET2/ESD1UPPW) -
              COUNT(030)
   IF LASTCC =0 -
      THEN -
      PRINT  INFILE(VSDSET2) -
             FADDR(65) -
             HEX
   /* LOAD A VSAM RELATIVE-RECORD FILE FROM AN */
   /* EXISTING SAM FILE                  */
   REPRO      INFILE(INDSET3,ENV(RECFM(F), -
                BLKSZ(100))) -
              OUTFILE(VSDSET3) -
              SKIP(10)
```

```
   IF LASTCC = 0 -
      THEN -
      PRINT  INFILE(VSDSET3) -
             TNUM(25)
   /* PRINT THE CONTENTS OF THE SAM FILE */
       PRINT INFILE(INDSET3,ENV(RECFM(F), -
               BLKSZ(100))) -
             COUNT(20) -
             CHAR
   /* LOAD A VSAM KEY-SEQUENCED FILE */
   /* FROM A NONVSAM FILE */
   REPRO     INFILE(INDSET4,ENV(RECFM(F), -
               BLKSZ(80))) -
             OUTFILE(VSDSET4/KSD2UPPW)
   IF LASTCC = 0 -
      THEN -
      PRINT  INFILE(VSDSET4/KSD2RDPW) -
             FROMKEY(AAAAJA) -
             TOKEY(AAAAJ9)
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control inroduced in VSE/VSAM Release 2, ignore this section.

Explanation of Job Control Statements

1. The INDSET1 DLBL statement describes the ISAM file which is to be copied into the VSAM file 'EXAMPLE.KSDS1'.
2. The INDSET2 DLBL statement describes the variable-length SAM file which is to be copied into the VSAM file 'EXAMPLE.ESDS1'.
3. The INDSET3 DLBL statement describes the fixed SAM file which is to be copied into the VSAM file 'EXAMPLE.RRDS1'.
4. The INDSET4 DLBL statement describes the fixed SAM file which is to be copied into the VSAM file 'EXAMPLE.KSDS2'.
5. The VSDSET1 DLBL statement describes a VSAM key-sequenced file which is to be loaded and printed.
6. The VSDSET2 DLBL statement describes a VSAM entry-sequenced file which is to be loaded and printed and indicates that it is in the catalog identified by the UCAT2 DLBL statement.
7. The VSDSET3 DLBL statement describes a VSAM relative-record file which is to be loaded and printed.
8. The VSDSET4 DLBL statement describes a VSAM key-sequenced file which is to be loaded and printed.
9. The UCAT2 DLBL statement describes the user catalog in which the file 'EXAMPLE.ESDS1' is cataloged. Specification of the user catalog by the CAT parameter on the DLBL statement with *filename* VSDSET2 enables reference to cataloged information describing the file which is required to open it. The VSAM master catalog need not be specified since it can always be referenced in the absence of a DLBL statement for the job catalog (IJSYSUC).

Explanation of Commands

The first REPRO command causes a VSAM key-sequenced file to be loaded from an ISAM file.

1. The INFILE parameter is required and names the file containing the source data. The *filename* of the DLBL statement for this file must be identical to this name. The ENV subparameter is required for all nonVSAM files.
2. The OUTFILE parameter is required and names the target file. The *filename* of the DLBL statement for this file must be identical to this name. The update or higher password of the VSAM file is required.

If the REPRO operation was successfully executed, then the contents of the VSAM key-sequenced file just loaded is printed. The format of the listing is DUMP, since the default is taken.

1. The INFILE parameter is required and names the file to be printed. The *filename* of the DLBL statement for this file must be identical to this name. The read or higher password is required to print the file.
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 'ABC'.

The second REPRO command causes a VSAM entry-sequenced file to be loaded from an existing variable unblocked SAM file.

1. The INFILE parameter is required and names the file containing the source data. The *filename* of the DLBL statement for this file must be identical to this name. The ENV subparameter is required for all nonVSAM files. For a variable SAM file, BLKSZ must equal the largest block, and RECSZ must be 4 less than BLKSZ.
2. The OUTFILE parameter is required and names the target file. The *filename* of the DLBL statement for this file must be identical to this name. The update or higher password of the VSAM file is required.
3. The COUNT parameter specifies that the first 30 records of the SAM file are to be loaded.

If the REPRO operation was successfully executed, then the contents of the VSAM entry-sequenced file just loaded is printed in hexadecimal format.

1. The INFILE parameter is required and names the file to be printed. The *filename* of the DLBL statement for this file must be identical to this name. Note that a password is not required since the file does not have a read password.
2. The FADDR parameter specifies that the first record printed is that record whose relative byte address is exactly equal to 65.
3. The HEX parameter specifies that the listing is to be in hexadecimal format.

The third REPRO command causes a VSAM relative-record file to be loaded from an existing fixed SAM file. The relative-record file can receive only fixed length records that equal its defined record length.

1. The INFILE parameter is required and names the file containing the source data. The *filename* of the DLBL statement for this file must be identical to this name. The ENV subparameter is required for all nonVSAM files.

2. The OUTFILE parameter is required and names the target file which is not password protected. The *filename* of the DLBL statement for this file must be identical to this name.
3. The SKIP parameter specifies that the first 10 records of the SAM file are to be bypassed.

If the REPRO operation was successfully executed, then the contents of the VSAM relative-record file just loaded are printed.

1. The INFILE parameter is required and names the file to be printed. The *filename* of the DLBL statement for this file must be identical to this name. A password is not necessary when a file is not password protected.
2. The TNUM parameter limits the output to the first 25 relative records.

The PRINT command causes the contents of the SAM file to be printed.

1. The INFILE parameter is required and names the file to be printed. The *filename* of the DLBL statement for this file must be identical to this name. The ENV subparameter is required for all nonVSAM files.
2. The COUNT parameter specifies that only 20 records are to be printed.
3. The CHAR parameter specifies that the listing is to be in character format.

The last REPRO command causes a VSAM key-sequenced file to be loaded from a nonVSAM file.

1. The INFILE parameter is required and names the file containing the source data. The *filename* of the DLBL statement for this file must be identical to this name. The ENV subparameter is required for all nonVSAM files.
2. The OUTFILE parameter is required and names the target file. The *filename* of the DLBL statement for this file must be identical to this name. Since the VSAM file is password protected, its update or higher-level password is required.

If the REPRO operation was successfully executed, then the contents of the VSAM key-sequenced file just loaded are printed. The FROMKEY and TOKEY parameters are used to limit the output to a specific range of keys.

1. The INFILE parameter is required and names the file containing the source data. The *filename* of the DLBL statement for this file must be identical to this name. Since the VSAM file is password protected, its read or higher-level password is required.
2. The FROMKEY and TOKEY parameters specify the keys at which printing is to begin and end, respectively.

Example 6: Modifying and Printing the Contents of VSAM Files

This example shows techniques for modifying the contents of VSAM files using the REPRO command. It can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL06
// ASSGN    SYS006,232
// ASSGN    SYS007,233
// ASSGN    SYS013,234
// DLBL     UCAT2,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03
// DLBL     INDSET4,'EXAMPLE.NONVSAM2'
// EXTENT   SYS013,231401,1,0,138,2
// DLBL     VSDSET2,'EXAMPLE.ESDS1',,VSAM,CAT=UCAT2
// EXTENT   SYS006,VSER03
// DLBL     VSDSET3,'EXAMPLE.RRDS1',,VSAM
// EXTENT   SYS007,VSER04
// DLBL     VSDSET4,'EXAMPLE.KSDS2',,VSAM
// EXTENT   SYS007,VSER04
// EXEC     IDCAMS,SIZE=AUTO
   REPRO    INFILE(INDSET4, -
              ENV(RECFM(F),BLKSZ(80))) -
            OUTFILE(VSDSET4/KSD2UPPW) -
            REPLACE
   IF LASTCC = 0 -
       THEN -
   PRINT    INFILE(VSDSET4/KSD2RDPW) -
            FROMKEY(AAAAJA) -
            TOKEY(AAAAJ9)
   REPRO    INFILE(VSDSET3) -
            OUTFILE(VSDSET2/ESD1UPPW) -
            REUSE
   IF LASTCC = 0 -
      THEN -
   PRINT    INFILE(VSDSET2)
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

Explanation of Job Control Statements

1. The UCAT2 DLBL statement describes the user catalog D27UCAT2. This DLBL statement will be explicitly referenced by means of the CAT parameter of the DLBL statement for the file cataloged in D27UCAT2. Since this job contains files cataloged in both the master and user catalogs, a DLBL statement with the *filename* IJSYSUC cannot be used as there is then no way to explicitly reference the master catalog by means of the REPRO command.
2. The INDSET4 DLBL statement describes the nonVSAM file to be copied into the VSAM file EXAMPLE.KSDS2.
3. The VSDSET2 DLBL statement describes a VSAM entry-sequenced file which is to be reloaded and indicates that it is in the catalog identified by the UCAT2 DLBL statement.

4. The VSDSET3 DLBL statement describes a VSAM relative-record file which is to be copied into the VSAM entry-sequenced file EXAMPLE.ESDS2.
5. The VSDSET4 DLBL statement describes a VSAM key-sequenced file which is to receive replacement records.

Explanation of Commands

The first REPRO command causes records in the VSAM key-sequenced file to be replaced with input from a nonVSAM file.

1. The INFILE parameter is required and names the file containing the source data. The *filename* of the DLBL statement for this file must be identical to this name. The ENV subparameter is required for all nonVSAM files.
2. The OUTFILE parameter is required and names the target file. The *filename* of the DLBL statement for this file must be identical to this name. Since the VSAM file is password protected, its update or higher-level password is required.
3. The REPLACE parameter causes a record in the output file having the same key as a record in the input file to be replaced. Records in the input file whose key is not already contained in the output file will be inserted in the output file.

If the REPRO operation was successfully executed, then the contents of the VSAM key-sequenced file just changed are printed.

1. The INFILE parameter is required and names the file containing the source data. The *filename* of the DLBL statement for this file must be identical to this name. Since the VSAM file is password protected, its read or higher-level 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 file to be loaded from the VSAM relative-record file.

1. The INFILE parameter is required and names the file containing the source data. The *filename* of the DLBL statement for this file must be identical to this name. Since this file is unprotected, no password is required.
2. The OUTFILE parameter is required and names the target file. The *filename* of the DLBL statement for this file must be identical to this name. The update or higher password of the VSAM file is required.
3. The REUSE parameter specifies that any records already in the entry-sequenced file output are to be overwritten since the entry-sequenced file was defined with the REUSE attribute.

If the REPRO operation was successfully executed, then the entire contents of the reloaded VSAM entry-sequenced file are printed.

1. The INFILE parameter is required and names the file containing the source data. The *filename* of the DLBL statement for this file must be identical to this name. Since no read password was specified for this file, no password is required.

Example 7: Modifying and Listing the Cataloged Attributes of a File.

This example shows how the cataloged attributes of a file may be modified. It can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL07
// ASSGN    SYS006,232
// ASSGN    SYS007,233
// DLBL     IJSYSUC,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03
// DLBL     VOL4,,,VSAM
// EXTENT   SYS007,VSER04
// EXEC     IDCAMS,SIZE=AUTO
   ALTER    EXAMPLE.KSDS1.DATA/KSD1PSWD -
            FREESPACE(10,10) -
            FILE(VOL4) -
            CATALOG(AMASTCAT)
   IF LASTCC = 0 -
       THEN -
   LISTC    ENT(EXAMPLE.KSDS1.DATA) -
            ALL -
            CATALOG(AMASTCAT/MCATMRPW)
   ALTER    EXAMPLE.ESDS1/ESD1MRPW -
            MRPW(ESD1PWMR) -
            CTLPW(ESD1PWCT) -
            UPDPW(ESD1PWUP) -
            RDPW(ESD1PWRD) -
            AUTH(ESD1AUTH)
   IF LASTCC = 0 -
       THEN -
   LISTC    ENT(EXAMPLE.ESDS1/ESD1PWMR) -
            CLUSTER -
            ALL
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The VOL4 DLBL statement describes the location of the catalog recovery area where the altered entry of EXAMPLE.KSDS1.DATA will be recorded.

The FILE parameters in the first ALTER command is required as the entry being altered is defined in a recoverable catalog.

Explanation of Job Control Statements

1. The IJSYSUC DLBL statement describes the user catalog D27UCAT2 as a job catalog.

Explanation of Commands

The first ALTER command (of EXAMPLE.KSDS1.DATA) shows a way to tune space management for a file. The data object was originally defined (see Example 3) with 40% free space for the data component. After initial loading, this percentage is reduced because further activity against this file will not be of the mass insert type.

1. The name of the data object is required as all alterations must name the object which contains the attributes to be altered. The cluster password is required to alter the data component.
2. The FREESPACE parameter specifies the new percentages of free space to be allowed in control intervals and control areas, respectively.
3. The CATALOG parameter specifies the name of the master catalog. Since a DLBL statement with the *filename* IJSYSUC appears in the job control, the master catalog must be explicitly specified in order to alter an object cataloged in it.

If the ALTER operation was successfully executed, then the data object altered is listed.

1. The ENT parameter specifies the name of the data object to be listed.
2. The ALL parameter specifies that all data object fields are to be listed.
3. The CATALOG parameter is required to name the master catalog. Since a DLBL statement with the *filename* IJSYSUC appears in the job control but the entry is in the master catalog, the master catalog must be specified. The master password is required since LISTCAT ALL displays protection attributes and the entry is password protected (and no password was supplied with entryname).

The second ALTER command (of EXAMPLE.ESDS1) shows a way to change the security scheme of a file. Establishing a security scheme for an existing nonprotected file would be done in the same manner.

1. The name of the cluster object is required as all alterations must name the object which contains the attributes to be altered. The master password of the cluster is required to alter a security-protected object.
2. The MRPW, CTLPW, UPDPW, and RDPW parameters respectively specify the new master, control, update, and read passwords for the cluster object.
3. The AUTH parameter specifies the name of the user authorization verification module, which is invoked for access authorization of this file when the master password is not provided.

If the ALTER operation was successfully executed, then the cluster object just altered is listed.

1. The ENT parameter specifies the name of the cluster object to be listed. The master password of the object allows its security information to be listed.
2. The CLUSTER parameter specifies that only the attributes of the cluster object will be listed.
3. The ALL parameter specifies that all cluster object attributes are to be listed.

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.

Example 8 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL08
// ASSGN    SYS010,230
// ASSGN    SYS007,233
// DLBL     BASEDD,'EXAMPLE.KSDS2',,VSAM
// EXTENT   SYS007,VSER04
// DLBL     AIXDD,'EXAMPLE.AIX',,VSAM
// EXTENT   SYS007,VSER04
// DLBL     IDCUT1,'SORT.WORK.ONE',,VSAM
// EXTENT   SYS010,VSER01
// DLBL     IDCUT2,'SORT.WORK.TWO',,VSAM
// EXTENT   SYS010,VSER01
// DLBL     VOL4,,,VSAM
// EXTENT   SYS007,VSER04
// DLBL     PATHDD,'EXAMPLE.PATH',,VSAM
// EXTENT   SYS007,VSER04
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   AIX -
            ( NAME(EXAMPLE.AIX) -
              RELATE(EXAMPLE.KSDS2/KSD2MRPW) -
              MRPW(AIXMRPW) -
              UPDPW(AIXUPPW) -
              READPW(AIXRDPW) -
              KEYS(3 0) -
              RECORDSIZE(40 50) -
              VOL(VSER04) -
              CYL(3 1) -
              FILE(VOL4) -
              NONUNIQUEKEY -
              UPGRADE -
            ) -
            CATALOG(AMASTCAT/MCATUPPW)
   /* DEFINE A PATH OVER THE ALTERNATE INDEX */
   DEFINE   PATH -
            ( NAME(EXAMPLE.PATH) -
              PATHENTRY(EXAMPLE.AIX/AIXMRPW) -
              READPW(PATHRDPW) -
              FILE(VOL4) -
            ) -
            CAT(AMASTCAT/MCATUPPW)
   /* BUILD THE ALTERNATE INDEX */
   BLDINDEX -
            INFILE(BASEDD) -
            OUTFILE(AIXDD/AIXUPPW) -
            INDATASET(EXAMPLE.KSDS2/KSD2RDPW) -
            OUTDATASET(EXAMPLE.AIX/AIXUPPW) -
            CATALOG(AMASTCAT/MCATUPPW)
            WORKVOLUMES(VSER01)
   /* PRINT THE BASE CLUSTER VIA THE ALTERNATE KEY */
   /* USING THE PATH DEFINED TO CREATE THIS */
   /* RELATIONSHIP */
   PRINT INFILE(PATHDD/PATHRDPW)
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control starements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 1, ignore this section.

The BASEDD DLBL statement describes the base cluster.

The AIXDD DLBL statement describes the alternate index.

The IDCUT1 and IDCUT2 DLBL statements describe the *file-IDs* and a volume containing VSAM data space to be made available to BLDINDEX for defining and using two sort work files 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 VOL4 DLBL statement describes the volume containing the catalog recovery area for the objects being defined.

The FILE parameter is required because the alternate index and path are being defined in a recoverable catalog.

The INFILE parameter is required and names the base cluster. The *filename* of the DLBL statement for this object must be identical to this name. Note that a password is not required since the base cluster is identified with the cluster name which is not protected. The INFILE and OUTFILE parameters are mutually exclusive with the INDATASET and OUTDATASET parameters.

The OUTFILE parameter is required and names the alternname index. The *filename* of the DLBL statement for this object must be identical to this name. The update password of the alternate index is also required.

Explanation of Job Control Statements

1. The PATHDD DLBL statement describes the path relating the alternate index and base cluster. It is required for the PRINT command.
2. No job catalog job control is required because all functions will use the master catalog.

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 and provides the base cluster master password.
3. The MRPW, UPDPW, and READPW parameters specify the master, update, and read 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 VOL parameter specifies the volume containing the alternate index.
7. The CYL parameter specifies the amount of space to be allocated to the alternate index.
8. The NONUNIQUE parameter specifies that the base cluster can contain multiple occurrences of any one alternate key.
9. 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.
10. 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 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.

1. The INDATASET parameter is required and names the base cluster, and supplies the base cluster's master password which is required for defining an alternate index over it.
2. The OUTDATASET parameter is required and names the alternate index. The update password of the alternate index is also required.
3. The CATALOG parameter is required because the master catalog is password protected. It specifies the name of the master catalog and its update password. If it is necessary for BLDINDEX to use external sort work files, they will be defined in and deleted from the master catalog. The update password will permit these actions.
4. The WORKVOLUMES parameter is required except if an entry-sequenced default model exists in the pertinent catalog. Space for work files will be allocated on volume VSER01.

The PRINT command causes the base cluster to be printed by means of the alternate key using the path defined to create this relationship.

1. The INFILE parameter is required and names the path object. The *filename* of the DLBL statement for this object must be identical to this name. The password required is the read or higher-level password of the object being opened, in this case, the path.

Example 9: Defining a VSAM Data Space and Cluster on an FBA Volume.

This example defines a VSAM data space and cluster on an FBA volume owned by the master catalog.

Example 9 can be viewed as a logical follow-on to the previous examples.

```
// JOB     EXAMPLE09
// ASSGN   SYS007,2C1
// DLBL    VOL5,,,VSAM
// EXTENT  SYS007,VSER05,1,0,1600,12000
// EXEC    IDCAMS,SIZE=AUTO
  /* DEFINE A SPACE ON A VOLUME ASSIGNED TO AN FBA DEVICE */
  DEFINE SPACE -
         ( ORIGIN(1600) -
           FILE(VOL5) -
           VOL(VSER05) -
           BLOCKS(12000) -
           CLASS(0)) -
         CATALOG(AMASTCAT/MCATUPPW)
   /* DEFINE A KEY-SEQUENCED FILE IN THE CLASS 0 SPACE */
   /* ON THE FBA VOLUME */
   DEFINE CLUSTER -
          ( NAME(EXAMPLE.KSDS3)-
            RDPW(KSD3PSWD) -
            FILE(VSER05) -
            VOL(VSER05) -
            RECORDSIZE(250,250) -
            KEYS(8 2) -
            FREESPACE(10,15)) -
          DATA-
         ( NAME(EXAMPLE.KSDS3.DATA) -
           RECORDS(800))-
          INDEX-
          ( NAME(EXAMPLE.KSDS3.INDEX) -
            BLOCKS (20)) -
          CATALOG(AMASTCAT/MCATUPPW)
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The VOL5 DLBL/EXTENT statements describe the volume and space on an FBA device which requires allocations in blocks or records. The space defined is expressed in the EXTENT statement as 12000 blocks beginning at block 1600 on volume VSER05.

The DEFINE SPACE FILE parameter is required and names the *filename* of the DLBL statement describing the volume on which space is being defined.

The FILE parameter in the DEFINE CLUSTER command is required since the file is being defined in a recoverable catalog.

Explanation of Commands

The DEFINE SPACE command creates space on an FBA volume.

1. The ORIGIN parameter indicates 1600 is the beginning block number.
2. The VOL parameter is required and specifies the volume that is to contain the data space.

3. The BLOCKS parameter specifies the amount of space to be allocated on this volume. An amount of space, the size of the max-CA (device-dependent), will be taken from the 12000 blocks for the CRA because this is the first space defined for the volume which is owned by a recoverable catalog.
4. The CLASS parameter specifies that the space is to be classified as class 0 (same as the default).
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 DEFINE CLUSTER command defines a key-sequenced file into the class 0 space on the FBA volume.

1. The CLUSTER parameter is required and NAME specifies the cluster beng defined.
2. The RDPW parameter specifies the read password of this cluster. Since no master password is defined, this password will be propagated up to the master level.
3. The VOL parameter specifies the volume containing this file.
4. The RECORDSIZE parameter specifies the average and maximum record sizes.
5. The KEYS parameter specifies the key length and offset from the beginning of the record.
6. The FREESPACE parameter specifies the percentage of space within control intervals and control areas, respectively, that is originally free.
7. 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.
8. The RECORDS parameter specifies space suballocation in records (RECORDS or BLOCKS are the only valid parameters for an FBA device).
9. 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 by VSAM.
10. The BLOCKS parameter specifies space suballocation in blocks (RECORDS or BLOCKS are the only valid parameters for an FBA device).
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.

Example 10: Exporting a VSAM File

This example exports an entry-sequenced file from a user catalog named D27UCAT2 to the second file of a labeled tape. Following the successful creation of the portable copy, the temporary export flag is set for the file in D27UCAT2 as well as the inhibit update flag, which will prevent any access other than read access to the file. When the file is imported, the inhibit update flag will be set for the file in the target catalog.

Example 10 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL10
// ASSGN    SYS005,382
// ASSGN    SYS006,232
// DLBL     IJSYSUC,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03
// DLBL     SOURCE,'EXAMPLE.ESDS1',,VSAM
// EXTENT   SYS006,VSER03
// TLBL RECEIVE,'PORTABLE.TAPE2',,TAPE01
// EXEC     IDCAMS,SIZE=AUTO
   EXPORT   EXAMPLE.ESDS1/ESD1PWMR -
            INFILE(SOURCE) -
            OUTFILE(RECEIVE, -
              ENV(PDEV(2400) NOREWIND STDLABEL)) -
            RECORDMODE -
            TEMP -
            INHIBITSOURCE -
            INHIBITTARGET
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The SOURCE DLBL statement describes the entry-sequenced file to be exported from the job catalog onto a tape.

The INFILE parameter is required and identifies the file to be exported. The *filename* of the DLBL statement for this object must be identical to this name.

Explanation of Job Control Statements

1. SYS005 must be used in the ASSIGN statement for magnetic tape output (OUTFILE).
2. The IJSYSUC DLBL statement describes the job catalog that contains the entry-sequenced file to be exported.
3. The RECEIVE TLBL statement describes the portable file that will contain the exported file. Note that it specifies the second file on the volume.

Explanation of Command

The EXPORT command causes an entry-sequenced file from a user catalog named D27UCAT2 to be exported to the second file of a labeled tape.

1. The name of the cluster object is required. The master password is required to export a password-protected object.

2. The OUTFILE parameter is required and identifies the portable file. The *filename* of the TLBL statement for this file must be identical to this name. The PDEV subparameter must be specified for a tape device. The BLKSZ subparameter defaults to 2048. The NOREWIND subparameter is specified because the file is on a multifile volume. The STDLABEL subparameter specifies standard label processing for the tape.
3. The RECORDMODE parameter overrides the default (CIMODE) and causes EXPORT processing using record mode operations.
4. The TEMP parameter causes the temporary export attribute to be set and prevents the file from being deleted.
5. The INHIBITSOURCE parameter causes the inhibit update flag to be in the catalog that contained the entry for the file being exported.
6. The INHIBITTARGET parameter causes the inhibit update flag to be set in the target catalog when the file is imported.

Example 11: Exporting an Alternate Index and Base Cluster

This example exports the alternate index and base cluster 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 as the delete of the base cluster will cause deletion of all objects defined over it.

Example 11 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL11
// ASSGN    SYS005,382
// ASSGN    SYS007,233
// DLBL     SOURCE,'EXANOKE.AIX',,VSAM
// EXTENT   SYS007,VSER04
// TLBL RECEIVE,'PORTABLE.TAPE3',,TAPE01
// EXEC     IDCAMS,SIZE=AUTO
   EXPORT   EXAMPLE.AIX/MCATMRPW -
            INFILE(SOURCE) -
            OUTFILE(RECEIVE, -
              ENV(PDEV(2400) BLKSZ(2056)NOREWIND))
/*
// DLBL     SOURCE,'EXAMPLE.KSDS2',,VSAM
// EXTENT   SYS007,VSER04
// TLBL RECEIVE,'PORTABLE.TAPE4',,TAPE01
// EXEC     IDCAMS,SIZE=AUTO
   EXPORT   EXAMPLE.KSDS2/MCATMRPW -
            INFILE(SOURCE) -
            OUTFILE(RECEIVE, -
              ENV(PDEV(2400) BLKSZ(2056)NOREWIND))
/*
// MTC      RUN,SYS005
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The SOURCE DLBL statements describe the files being exported.

The INFILE parameter in both EXPORT commands is required and identifies the alternate index being exported. It also provides access to the catalog recovery area. The *filename* of the DLBL statement for this object must be identical to this name.

Explanation of Job Control Statements

1. SYS005 must be used in the ASSIGN statement for magnetic tape output (OUTFILE).
2. The RECEIVE TLBL statements describe the portable files. Note that these are the third and fourth files on a multifile volume.
3. The MTC statement rewinds and unloads the portable copy tape after the EXPORT commands are completed.

Explanation of Commands

The first 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.
2. The OUTFILE parameter is required and identifies the portable file. The *filename* of the TLBL statement for this object must be identical to this name. The PDEV subparameter must be specified for a tape device. Normally, the BLKSZ subparameter defaults to 2048; however, for improved performance it is recommended that you specify a value at least 8 bytes larger than the file's data-component control-interval size. The NOREWIND subparameter is specified because the file is on a multifile volume.
3. Control interval processing (CIMODE) is effective as the default.
4. PERMANENT export is effective as the default.

The second 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.
2. The OUTFILE parameter is required and identifies the portable file. The *filename* of the TLBL statement for this object must be identical to this name. The PDEV subparameter must be specified for a tape device. Normally the BLKSZ subparameter defaults to 2048; however, for improved performance it is recommended that you specify a value at least 8 bytes larger than the file's data-component control-interval size. The NOREWIND subparameter is specified because the file is on a multifile volume.
3. The RECORDMODE parameter is not specified so processing defaults to CIMODE processing.
4. PERMANENT export is effective as the default.

Example 12: Disconnecting a User Catalog from a Master Catalog

This example disconnects user catalog D27UCAT1 from the master catalog. The user catalog volume need not be mounted; therefore, no DLBL statement to describe it is required.

Example 12 can be viewed as a logical follow on to the previous examples.

```
// JOB      EXAMPL12
// EXEC     IDCAMS,SIZE=AUTO
   EXPORT   D27UCAT1/MCATUPPW -
            DISCONNECT
/*
/&
```

Explanation of Command

1. The name of the user catalog is required. The update or higher-level password of the master catalog is required to disconnect the user catalog.
2. The DISCONNECT parameter is required to signal the disconnect action.

Example 13: Importing a Base Cluster and Alternate Index

This example will import the base cluster and alternate index exported in Example 11 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.

Example 13 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL13
// ASSGN    SYS004,382
// MTC      REW,SYS004
// ASSGN    SYS007,233
// TLBL SOURCE,'PORTABLE.TAPE4',,TAPE01
// DLBL   RECEIVE,'EXAMPLE.KSDS2',,VSAM
// EXTENT  SYS007,VSER04
// EXEC     IDCAMS,SIZE=AUTO
   IMPORT   INFILE(SOURCE,ENV(PDEV(2400)BLKSZ(2056))) -
            OUTFILE(RECEIVE) -
            OBJECTS( -
            (EXAMPLE.KSDS2.INDEX -
            USECLASS(0 0))) -
            CAT(AMASTCAT/MCATUPPW)
/*
// MTC      REW,SYS004
// TLBL SOURCE,'PORTABLE.TAPE3',,TAPE01
// DLBL    RECEIVE,'EXAMPLE.AIX',,VSAM
// EXTENT  SYS007,VSER04
// EXEC     IDCAMS,SIZE=AUTO
   IMPORT   INFILE(SOURCE,ENV(PDEV(2400)BLKSZ(2056))) -
            OUTFILE(RECEIVE) -
            CAT(AMASTCAT/MCATMRPW)
/*
// MTC      RUN,SYS004
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The RECEIVE DLBL statements describe the files being imported. Since the name is not being changed, the original name is used as *file-ID*.

The OUTFILE parameters in both IMPORT commands is required and identifies the file being imported. It also provides access to the catalog recovery area. The *filename* of the DLBL statement for this object must be identical to this name.

Explanation of Job Control Statements

1. SYS004 must be used in the ASSIGN statement for magnetic tape input (INFILE).
2. The SOURCE TLBL statements describe the portable files. Note that file sequence numbers four and three are specified.
3. The first two MTC statements position the tape at load point so that the desired file can be located. The third MTC rewinds and unloads the tape when the import operation is completed.

Explanation of Commands

The first IMPORT command causes the base cluster to be imported into the master catalog. The class of space from which the index component is to be allocated is changed from that contained on the portable file.

1. The INFILE parameter is required and identifies the portable file. The PDEV subparameter must be specified for a tape device. The BLKSZ subparameter must be specified because the IMPORT uses the same value (2056) that the EXPORT used. The *filename* of the TLBL statement for this object must be identical to this name. NOREWIND and STDLABEL are default values.
2. The OBJECTS parameter identifies the index component of the file being imported. The USECLASS parameter specifies that the index component is to be allocated from class 0 space. This specification overrides the useclass attributes in the portable file.
3. The CATALOG parameter is required because 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 identifies the portable file. The PDEV subparameter must be specified for a tape device. The BLKSZ subparameter must be specified because the IMPORT uses the same value (2056) that the EXPORT used. The *filename* of the TLBL statement for this object must be identical to this name. NOREWIND and STDLABEL are default values.
2. The CATALOG parameter is required since the master catalog is password protected. It specifies the name of the master catalog and its master password which is required to define an alternate index over a password protected base cluster.

Example 14: Importing an Entry-Sequenced File

This example imports the entry-sequenced file exported in Example 10. The exported copy of the file will replace the original copy of the file. IMPORT will find the duplicate name, EXAMPLE.ESDS1, in catalog D27UCAT2, delete it since the temporary export attribute will have been set for EXAMPLE.ESDS1, then redefine it using the catalog information from the portable file.

Example 14 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL14
// ASSGN    SYS004,382
// MTC      REW,SYS004
// ASSGN    SYS006,232
// DLBL     IJSYSUC,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03
// TLBL SOURCE,'PORTABLE.TAPE2',,TAPE01
// DLBL     RECEIVE,'EXAMPLE.ESDS1',,VSAM
// EXTENT   SYS006,VSER03
// EXEC     IDCAMS,SIZE=AUTO
   IMPORT   INFILE(SOURCE,ENV(PDEV(2400))) -
            OUTFILE(RECEIVE) -
            CATALOG(D27UCAT2/UCATUPPW)
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The IJSYSUC DLBL statement describes the user catalog that contains the entry-sequenced file to be imported as a job catalog.

The RECEIVE DLBL statement describes the VSAM file to be replaced.

The OUTFILE parameter is required and identifies the VSAM cluster. The *filename* of the DLBL statement for this file must be identical to this name.

Explanation of Job Control Statements

1. SYS004 must be used in the ASSIGN statement for magnetic tape input (INFILE).
2. The SOURCE TLBL statement describes the portable file. Note that the second file on the volume is specified.

Explanation of Command

The IMPORT command causes an entry-sequenced file to be imported from the second file of a labeled tape into a user catalog named D27UCAT2.

1. The INFILE parameter is required and identifies the portable file. The *filename* of the TLBL statement for this file must be identical to this name. The PDEV subparameter must be specified for a tape device. The BLKSZ subparameter need not be specified since IMPORT will use the same default (2048) that EXPORT used. NOREWIND and STDLABEL are default values.
2. The CATALOG parameter is required since the target catalog is not the default catalog and is password protected. It specifies the name of the

user catalog. The update password of the catalog permits IMPORT to delete the duplicate entry as well as to redefine the imported file.

Example 15: Importing a Unique File

This example imports a key-sequenced file with UNIQUE component allocation named EXAMPLE.KSDS1 from a tape to a user catalog. The names of the cluster and each of its components is changed by means of the NEWNAME subparameter.

```
// JOB      EXAMPL15
// ASSGN    SYS004,382
// ASSGN    SYS006,232
// DLBL     IJSYSUC,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03
// TLBL SOURCE,'PORTABLE.TAPE1',,TAPE01
// DLBL     FILDATA,'EXAMNEW.KSDS1',,VSAM
// EXTENT   SYS006,VSER03,1,0,320,60
// DLBL     RECDATA,'EXAMNEW.KSD1.DATA',,VSAM
// EXTENT   SYS006,VSER03,1,0,320,40
// DLBL     RECIX,EXAMNEW.KSDS1.INDEX,,VSAM
// EXTENT   SYS006,VSER03,1,0,360,20
// EXEC     IDCAMS,SIZE=AUTO
   IMPORT   INFILE(SOURCE, -
              ENV(PDEV(2400)BLKSZ(6000) UNLOAD)) -
            OUTFILE(FILDATA) -
            CATALOG(D27UCAT2/UCATUPPW) -
            OBJECTS ( -
              (EXAMPLE.KSDS1 -
              VOL(VSER03) -
              NEWNAME(EXAMNEW.KSDS1)) -
              (EXAMPLE.KSDS1.DATA -
              NEWNAME(EXAMNEW.KSDS1.DATA) -
              FILE(RECDATA)) -
              (EXAMPLE.KSDS1.INDEX -
              NEWNAME(EXAMNEW.KSDS1.INDEX) -
              FILE(RECIX)))
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The IJSYSUC DLBL statement describes the user catalog into which the cluster is to be imported as a job catalog.

The FILDATA DLBL statement and its associated EXTENT statement identifies the volume and extent on the disk into which the unique file will be imported.

The RECDATA DLBL statement and its associated EXTENT statement defines space on the disk into which the unique data component will be imported. The *file-ID* specifies the changed name of the data component which is required to open the file.

The RECIX DLBL statement and its associated EXTENT statement define space on the disk into which the unique index component will be imported. The file-ID specifies the changed name of the index component.

The OUTFILE parameter is required and describes the VSAM file that will be imported. The *filename* of the DLBL statement for this file must be identical to this name. The changed *file-ID* on the DLBL statement is required to open the file; its associated EXTENT statement is used to define the space that the unique data space will occupy.

Explanation of Job Control Statements

1. SYS004 must be used in the ASSGN statement for magnetic tape input (INFILE).
2. The SOURCE TLBL statement describes the portable file created by the export which occurred in Example 9.
3. The RECDATA DLBL statement and its associated EXTENT statement defines space on the disk into which the unique data component will be imported. Note that the symbolic unit on the EXTENT statement is not required.
4. The RECIX DLBL statement and its associated EXTENT statement defines space on the disk into which the unique index component will be imported. Note that the symbolic unit on the EXTENT statement is not required.

Explanation of Command

The IMPORT command causes a key-sequenced file to be imported from a tape into a user catalog named D27UCAT2. Note that the key-sequenced file was originally exported from the master catalog and originally resided on a different volume.

1. The INFILE parameter is required and identifies the portable file. The *filename* of the TLBL statement for this file must be identical to this name. The PDEV subparameter must be specified for a tape device. The BLKSZ subparameter must be specified so that it is the same as that specified for the export operation. UNLOAD causes the tape to be rewound and unloaded when IMPORT completes. STDLABEL is the default value
2. The CATALOG parameter identifies the user catalog and supplies the update password of the catalog which is required when importing into a protected catalog.
3. The OBJECTS parameter is used to change attributes of the cluster and its components.
4. The VOL parameter specifies that the cluster is to reside on a new volume.
5. The NEWNAME subparameters successively specify new names for the cluster, its data component, and its index component.
6. The FILE subparameters identify the DLBL statement describing the unique space for the data and index components.

Example 16: Connecting a User Catalog to the Master Catalog

This example connects user catalog D27UCAT1 to the master catalog. The volume containing the user catalog need not be mounted. Example 16 can be viewed as a logical follow-on to the previous examples.

```
// JOB     EXAMPL16
// EXEC    IDCAMS,SIZE=AUTO
   IMPORT  OBJECTS ( -
             (D27UCAT1 -
             VOLUME(VSER02) -
             DEVICETYPE(2314)) -
           ) -
           CONNECT -
           CATALOG(AMASTCAT/MCATUPPW)
/*
/&
```

Explanation of Command

1. The name of the user catalog is required.
2. The VOLUME subparameter is required to identify the volume containing the user catalog.
3. The DEVICETYPE subparameter is required to identify the device type of the volume containing the user catalog.
4. The CONNECT parameter is required to signal the connect action.
5. The CATALOG parameter is required because the master catalog's update or higher-level password is required if the master catalog is password protected.

Example 17: Using REPRO to Unload a User Catalog to Tape

This example shows how a VSAM user catalog can be unloaded to tape using the catalog unload/reload feature of the REPRO command.

Example 17 can be viewed as a logical follow-on to the previous examples.

```
// JOB     EXAMPL17
// ASSGN   SYS005,382
// MTC     REW,SYS005
// ASSGN   SYS006,232
// DLBL    IJSYSUC,'D27UCAT2',,VSAM
// EXTENT  SYS006,VSER03
// DLBL    CATIN,'D27UCAT2',,VSAM
// EXTENT  SYS006,VSER03
// TLBL    CATOUT,'PORTABLE.TAPE1',,TAPE01
// EXEC    IDCAMS,SIZE=AUTO
   REPRO   INFILE(IJSYSUC/UCATMRPW) -
           OUTFILE(CATOUT -
           ENVIRONMENT -
           ( PDEV(2400) -
            RECFM(VARBLK) -
            REW -
            BLKSZ(5164) -
            RECSZ(516) -
            ))
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The CATIN DLBL statement describes the user catalog as a VSAM file to be opened and used by REPRO as the source for the unload operation.

Explanation of Job Control Statements

1. SYS005 must be used in the ASSGN statement for magnetic tape output (OUTFILE).
2. The IJSYSUC DLBL statement is required and describes the catalog as the job catalog for the job (and as a VSAM file to be opened and used by REPRO as the source for the unload operation).
3. The CATOUT TLBL statement describes a tape file which is to hold the unloaded copy of the catalog.

Explanation of Command

The REPRO command causes a VSAM user catalog to be unloaded to tape.

1. The INFILE parameter is required and identifies the job catalog as an input VSAM file. The master password allows opening the catalog as a file.
2. The OUTFILE parameter is required and describes the tape file which is to hold the unloaded copy of the catalog. The *filename* of the TLBL statement for this file must be identical to this name. The PDEV subparameter must be specified for a tape device. The RECFM subparameter must be specified to designate a variable blocked format.
3. The BLKSZ subparameter, in the case of a catalog file, must be a multiple of 516 plus 4. The RECSZ subparameter must be specified since the default is block size minus 4 and 516 is the only acceptable value (length of the catalog record +4).

Example 18: Using REPRO to Reload a User Catalog

This example shows how a VSAM user catalog can be reloaded from the backup copy created in Example 17 using the catalog unload/reload feature of the REPRO command.

Example 18 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL18
// ASSGN    SYS004,382
// MTC      REW,SYS004
// ASSGN    SYS006,232
// DLBL     IJSYSUC,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03
// DLBL     CATOUT,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03
// TLBL CATIN,'PORTABLE.TAPE1',,TAPE01
// EXEC     IDCAMS,SIZE=AUTO
   REPRO    INFILE(CATIN -
              ENVIRONMENT -
            ( PDEV(2400) -
              RECFM(VARBLK) -
              BLKSZ(5164) -
              RECSZ(516) -
            )) -
           OUTFILE(IJSYSUC/UCATMRPW)
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The CATOUT DLBL statement describes the user catalog as a VSAM file to be opened and used by REPRO as the target for the reload operation.

Explanation of Job Control Statements

1. SYS004 must be used in the ASSGN statement for magnetic tape input (INFILE).
2. The IJSYSUC DLBL statement is required and describes the user catalog to be reloaded as the job catalog (and as a VSAM file to be opened and used by REPRO as the target for the reload operation).
3. The CATIN TLBL statement describes the backup copy of the user catalog as a tape file.

Explanation of Command

The REPRO command causes a VSAM user catalog to be reloaded from the backup copy created in Example 17.

1. The INFILE parameter is required and identifies the tape file which holds the backup copy of the user catalog. The *filename* of the TLBL statement for this file must be identical to this name. The PDEV subparameter must be specified for a tape device. The RECFM subparameter must be specified to designate a variable blocked format. The BLKSZ subparameter must be specified to ensure that it is the same as that specified when the catalog was unloaded. The RECSZ subparame-

ter must be specified since the default is block size minus 4 and 516 is the only acceptable value (length of the catalog record +4).

2. The OUTFILE parameter is required and identifies the job catalog as a VSAM file to be opened. The master password allows opening the catalog as a file.

Example 19: Listing Catalog Recovery Areas

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.

Example 19 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL19
// ASSGN    SYS010,230
// ASSGN    SYS007,233
// DLBL     CRAVOL1,,,VSAM
// EXTENT   SYS010,VSER01
// DLBL     CRAVOL2,,,VSAM
// EXTENT   SYS007,VSER04
// DLBL     CATVOL,'AMASTCAT',,VSAM
// EXTENT   SYS010,VSER01
// EXEC     IDCAMS,SIZE=AUTO
   LISTCRA  INVOLUMES(VSER01 VSER04) -
            IFILE(CRAVOL1 CRAVOL2) -
            CATALOG(AMASTCAT/MCATMRPW CATVOL) -
            MASTERPW(MCATMRPW) -
            COMPARE -
            DUMP
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The CRAVOL1 DLBL statement identifies the first catalog recovery area to be listed.

The CRAVOL2 DLBL statement identifies the second catalog recovery area to be listed.

The CATVOL DLBL statement identifies the catalog to be compared against and makes it available as a file.

The IFILE parameter is required and specifies the catalog recovery areas to be listed by identifying the pertinent DLBL statements.

The CATALOG parameter is required if the COMPARE option is selected. It specifies the name of the catalog to be compared against, and its master password, and identifies its associate DLBL statement.

Explanation of Command

The LISTCRA command causes the catalog recovery areas on volumes VSER01 and VSER04 to be listed and their contents compared with the VSAM master catalog AMASTCAT.

1. The INVOLUMES parameter is required and specifies the catalog recovery areas to be listed by identifying the pertinent volumes.

2. The CATALOG parameter is required if the COMPARE option is selected. It specifies the name of the catalog to be compared against, and its master password.
3. The MASTERPW password specifies the master password of the master catalog which is necessary in order to open the catalog recovery areas.
4. The COMPARE parameter specifies that catalog records are to be compared with the records in the catalog recovery areas.
5. The DUMP parameter specifies the dump format for output.

Example 20: Using the EXPORTRA Command

This example performs the EXPORTRA function against all of the volumes owned by the VSAM master catalog AMASTCAT. All of the files listed in the catalog recovery areas on the catalog volume and on the other volumes owned by AMASTCAT are exported to a SAM file on another disk volume. Using the FORCE option causes EXPORTRA to ignore time stamp mismatches between the volumes and the catalog.

Example 20 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL20
// ASSGN    SYS010,230
// ASSGN    SYS007,233
// ASSGN    SYS013,234
// DLBL     CRAVOL1,,,VSAM
// EXTENT   SYS010,VSER01
// DLBL     CRAVOL2,,,VSAM
// EXTENT   SYS007,VSER04
// DLBL     PORT,'PORT'
// EXTENT   SYS013,231401,1,0,140,200
// EXEC     IDCAMS,SIZE=AUTO
   EXPORTRA -
            CRAVOLUMES((VSER01 ALL) (VSER04 ALL)) -
            CRA((CRAVOL1 ALL) (CRAVOL2 ALL)) -
            FORCE -
            OUTFILE(PORT ENV(BLKSZ(4000))) -
            CIMODE -
            MASTERPW(MCATMRPW)
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The CRAVOL1 DLBL statement identifies the first volume whose files are to be exported.

The CRAVOL2 DLBL statement identifies the second volume whose files are to be exported.

The CRA parameter is required and identifies the catalog recovery areas and volumes from which the export is to take place. The *filenames* of the DLBL statement 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.

Explanation of Job Control Statements

1. The PORT DLBL statement identifies the SAM file that is to receive the files being exported and their associated catalog information.

Explanation of Command

The EXPORTRA command exports everything appearing in the catalog recovery areas of volumes VSER01 and VSER04.

1. The CRAVOLUMES parameter is required and identifies the volumes from which the export is to take place. The ALL subparameter specifies that everything is to be exported from each catalog recovery area.
2. The FORCE parameter specifies that time stamp mismatches are to be ignored.
3. The OUTFILE parameter is required and identifies the SAM file that is to receive the exported information. The *filename* of the DLBL statement for this object must be identical to this name. The BLKSZ parameter specifies 4000-byte output records and is larger than any data control interval on the volumes being exported. The PDEV parameter is not required because OUTFILE is not on a tape device.
4. The CIMODE parameter specifies control interval processing. (CIMODE does not have to be specified because it is the default.) This causes processing by control intervals rather than on a record basis.
5. The MASTERPW parameter specifies the master password of the master catalog. This password is required in order to export catalog information.

Example 21: Using the IMPORTRA Command

This example imports all of the files that were exported using EXPORTRA in Example 20. The receiving catalog is the VSAM master catalog, and the CATALOG parameter is used to supply its master password.

Example 21 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL21
// ASSGN    SYS010,230
// ASSGN    SYS007,233
// ASSGN    SYS013,234
// DLBL     PORT,'PORT'
// EXTENT   SYS013,231401,1,0,140,200
// DLBL     VSAMIN,'DUMMY.NAME',,VSAM
// EXTENT   SYS010,VSER01
// EXTENT   SYS007,VSER04
// EXEC     IDCAMS,SIZE=AUTO
   IMPORTRA -
            INFILE(PORT ENV(BLKSZ(4000))) -
            OUTFILE(VSAMIN) -
            CATALOG(AMASTCAT/MCATMRPW)
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The VSAMIN DLBL statement and its accompanying EXTENT statements identify the volumes of the objects to be imported. This permits them to be reloaded from the portable file. The *file-ID* appearing on the DLBL statement is required. It is a dummy name that must not appear either in the catalog or among the names on the portable file.

The OUTFILE parameter is required and identifies the VSAM volumes involved in the import and the dummy *file-ID*. The *filename* of the DLBL statement for these volumes must be identical to this name.

Explanation of Job Control Statements

1. The PORT DLBL statement identifies the portable file created by EXPORTRA in Example 20.

Explanation of Command

The IMPORTRA command imports the files written on the portable file by EXPORTRA in Example 20.

1. The INFILE parameter is required and identifies the portable file. The *filename* of the DLBL statement for this file must be identical to this name. The BLKSZ parameter specifies 4000 bytes to match the block size specification in the XPRA command. The PDEV parameter is not required because INFILE is not on a tape device.
2. The CATALOG parameter is required because the catalog's master password is needed to import catalog information.

Example 22: Deleting Entries in a User Catalog and the User Catalog Itself

This example deletes entries in the VSAM user catalog and deletes the user catalog, D27UCAT2. The result of this job and that shown by Example 23 is to remove all VSAM files and catalogs defined in this set of examples.

Example 22 can be viewed as a logical follow-on to the previous examples.

```
// JOB      EXAMPL22
// ASSGN    SYS005,231
// ASSGN    SYS006,232
// ASSGN    SYS013,234
// DLBL     VSDSET2,'EXAMNEW.KSDS1',,VSAM
// EXTENT   SYS006,VSER03
// DLBL     UCAT1,'D27UCAT1',,VSAM
// EXTENT   SYS005,VSER02
// DLBL     NONVSAM,'EXAMPLE.NONVSAM1'
// EXTENT   SYS013,231401
// DLBL     IJSYSUC,'D27UCAT2',,VSAM
// EXTENT   SYS006,VSER03
// EXEC     IDCAMS,SIZE=AUTO
   /* DELETE THE NONVSAM FILE EXAMPLE.NONVSAM1 */
   DELETE   EXAMPLE.NONVSAM1 -
            SCRATCH -
            FILE(NONVSAM) -
            NONVSAM -
            CATALOG(D27UCAT1 UCAT1)
    /* DELETE VSAM FILES CATALOGED IN */
    /* USER CATALOG D27UCAT2 */
    DELETE  (EXAMPLE.ESDS1/ESD1PWMR -
            EXAMNEW.KSDS1/KSD1PSWD) -
            PURGE -
            FILE(VSDSET2) -
            CLUSTER -
            CATALOG(D27UCAT2)
    /* DELETE THE VSAM USER CATALOG D27UCAT2 */
    DELETE  D27UCAT2/UCATMRPW -
            USERCATALOG -
            PURGE
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The VSDSET2 DLBL statement identifies the VSAM key-sequenced file to be deleted. This DLBL statement is requires since the file was defined with the UNIQUE attribute.

The UCAT1 DLBL statement identifies the catalog from which the nonVSAM file is to be deleted. The catalog can only be referenced if the CATALOG parameter specifies the *dname* UCAT1.

The NONVSAM DLBL statement identifies the nonVSAM file to be deleted. This DLBL statement is required to enable the SCRATCH opinion of the DELETE command to be executed.

The IJSYSUC DLBL statement is required to identify the job catalog that is to be deleted.

The FILE parameter in the first DELETE command is required if the SCRATCH option is specified because the volume containing the VTOC

entry must be mounted. It identifies the file to be deleted. The *filename* of the DLBL statement for this object must be identical to this name.

The CATALOG parameter is required in this example. It identifies the catalog containing the entry to be deleted. This parameter must be explicitly specified since a DLBL statement for the job catalog (IJSYSUC) appears in the job control. A password is not required to delete a nonVSAM entry but specification of the *dname* is required to enable the catalog to be opened.

The FILE parameter in the second DELETE command is required since EXAMNEW.KSDS1 was defined with the UNIQUE attribute. The parameter is not required for EXAMPLE.ESDS1 because it was not defined with the UNIQUE attribute. If both files were defined with the UNIQUE attribute, a separate DELETE command would have to be specified for each file.

Explanation of Commands

The first DELETE command deletes the nonVSAM file named EXAMPLE.NONVSAM1.

1. The name of the nonVSAM file is required.
2. The SCRATCH parameter specifies that the VTOC entry of the object being deleted is to be removed.
3. The NONVSAM parameter insures that the entry being deleted is a nonVSAM file.
4. The CATALOG parameter is required in this example. It identifies the catalog containing the entry to be deleted. A password is not required to delete a nonVSAM entry.

The second DELETE command deletes the VSAM files cataloged in the user catalog D27UCAT2.

1. The names of the VSAM entry-sequenced and key-sequenced files together with the master password for each are required. File EXAMNEW.KSDS1 was originally defined as EXAMPLE.KSDS1, but when Example 11 caused the file to be imported, the name was changed.
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 file.
4. The CATALOG parameter is required in this example. It identifies the catalog containing the entry to be deleted. A password is not required to delete a nonVSAM entry.

The third 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.

Example 23: Deleting Entries in the Master Catalog and the Master Catalog Itself

This example deletes the second user catalog, the entries in the master catalog, and, finally, the master catalog itself.

Example 23 can be viewed as a logical follow-on to the previous examples.

```
// JOB     EXAMPL23
// ASSGN   SYS005,231
// ASSGN   SYS007,233
// DLBL    IJSYSUC,'D27UCAT1',,VSAM
// EXTENT  SYS005,VSER02
// DLBL    VSDSET3,'EXAMPLE.ESDS2',,VSAM,CAT=IJSYSCT
// EXTENT  SYS007,VSER04
// DLBL    VOL4,'EXAMPLE.KSDS2',,VSAM,CAT=IJSYSCT
// EXTENT  SYS007,VSER04
// EXEC    IDCAMS,SIZE=AUTO
   /* DELETE THE VSAM USER CATALOG D27UCAT1 */
   DELETE  D27UCAT1/UCATMRPW -
           USERCATALOG -
           PURGE
   /* DELETE VSAM ENTRY-SEQUENCED FILE NAMED */
   /* EXAMPLE.ESDS2 FROM THE MASTER CATALOG  */
   DELETE  EXAMPLE.ESDS2/ESD2MRPW -
           PURGE -
           FILE(VSDSET3) -
           ERASE
   /* DELETE VSAM KEY-SEQUENCED FILE NAMED */
   /* EXAMPLE.KSDS2 FROM THE MASTER CATALOG */
   DELETE  EXAMPLE.KSDS2 -
           FILE(VOL4) -
           CATALOG(AMASTCAT/MCATMRPW)
   /* DELETE VSAM RELATIVE-RECORD FILE NAMED */
   /* EXAMPLE.RRDS1 FROM THE MASTER CATALOG  */
   DELETE  EXAMPLE.RRDS1 -
           FILE(VOL4) -
           CLUSTER -
   /* DELETE VSAM SPACE FROM VOLUME VSER04 */
   DELETE  VSER04 -
           FILE(VOL4) -
           SPACE -
           CATALOG(AMASTCAT/MCATUPPW)
   /* DELETE THE VSAM MASTER CATALOG */
   DELETE  AMASTCAT/MCATMRPW -
           MASTERCATALOG -
           PURGE
/*
/&
```

Explanation of Shaded Items

The shaded parameters and job control statements that appear in this example are provided only for the convenience of users who are using the old (Release 1) method of job control. If you are using the job control introduced in VSE/VSAM Release 2, ignore this section.

The IJSYSUC DLBL statement is required to identify the job catalog being deleted.

The VSDSET3 DLBL statement identifies the VSAM entry-sequenced file to be deleted. This DLBL statement is required since the ERASE option is specified in the DELETE command for this file. The CAT parameter will override the DLBL statement for the job catalog (IJSYSUC) so that the master catalog can be accessed to open the file for the ERASE operation.

This DLBL statement also provides access to the catalog recovery area for this file.

The VOL4 DLBL statement identifies the VSAM key-sequenced file to be deleted, identifies the volume containing the catalog recovery area for both files being deleted, and identifies the volume (i.e., VSAM data space) to be deleted. The CAT parameter will override the DLBL statement for the job catalog (IJSYSUC) so that the master catalog can be accessed to open EXAMPLE.KSDS2 in order to overwrite the data with binary zeros, since the file was defined with the ERASE attribute.

The FILE parameter in the second DELETE command is required when the ERASE option is specified to enable the file to be opened. In addition, it is required since the file is being deleted from a recoverable catalog.

The FILE parameter in the third DELETE command is required since EXAMPLE.KSDS2, defined with the ERASE attribute, must be opened to allow the data to be overwritten with binary zeros. The *filename* of the DLBL statement for this object must be identical to this name. In addition, it is required since the file is being deleted from a recoverable catalog.

The FILE parameter in the fourth DELETE command is required since the catalog is recoverable and names the volume that contains the catalog recovery area for the object to be deleted. The *filename* of the DLBL statement for the object must be the same as this name.

The FILE parameter in the fifth DELETE command is required and identifies the volume to be deleted. The *filename* of the DLBL statement for the object must be the same as this name.

Explanation of Commands

The first DELETE command deletes the VSAM user catalog D27UCAT1.

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.

The second DELETE command deletes the VSAM entry-sequenced file EXAMPLE.ESDS2 from the master catalog.

1. The name of the VSAM file and its master password are required.
2. The PURGE parameter causes the entry to be deleted without regard for the retention period.
3. The ERASE parameter causes the data component to be overwritten with binary zeros. As a result, sensitive data is not left as a residue after the file is deleted.

The third DELETE command deletes the VSAM key-sequenced file EXAMPLE.KSDS2 from the master catalog. This command will cause the deletion of all objects defined over this base cluster, that is, the alternate index named EXAMPLE.AIX, which, in turn, will cause the deletion of paths over that object, that is, EXAMPLE.PATH.

1. The name of the VSAM file being deleted is required.
2. The CATALOG parameter identifies the catalog containing the file. The master password of the catalog covers the password requirements for all objects being deleted.

The fourth DELETE command deletes the VSAM relative-record file EXAMPLE.RRDS1 from the master catalog.

1. The name of the VSAM file being deleted is required. No password is required since the file is not password protected.
2. The CLUSTER parameter insures that the catalog object being deleted is a VSAM file.

The fifth DELETE command deletes the VSAM data space from volume VSER04. Since this volume no longer has VSAM files, it will cease to belong to the master catalog. If VSAM files still required this volume, the deletion would only free unused VSAM space on the volume.

1. The volume serial number of the volume is required.
2. The SPACE parameter is required to specify that the catalog object being deleted is a VSAM data space.
3. The CATALOG parameter identifies the catalog containing the VSAM data space entry. The update or higher password of the catalog is required to delete a volume or space.

The last DELETE command deletes the VSAM master catalog itself.

1. The name of the master catalog and its master password are required.
2. The MASTERCATALOG parameter is required to specify that the object being deleted is a master catalog.
3. The PURGE parameter causes the catalog to be deleted without regard for the retention period.

Example 24: Defining an Entry Sequence Default Model

This example creates a default model for a VSAM entry-sequence file.

```
// JOB     EXAMPL24
// EXEC    IDCAMS,SIZE=AUTO
   DEFINE  CLUSTER -
           (NAME((DEFAULT.MODEL.ESDS) -
            NOALLOCATION -
            CYLINDERS(1 1) -
            RECORDSIZE(2500 3000) -
            SPANNED -
            VOLUMES(VSER01) -
            NONINDEXED)
           CATALOG(AMASTCAT/MCATPPW)
/*
/&
```

Explanation of Command

1. The cluster parameter is required and NAME specifies the name of the cluster being defined.
2. The NOALLOCATION parameter indicates that no space allocation is to take place for the define.
3. The CYLINDERS parameter is required because the MODEL parameter is not specified. One cylinder is specified for both the primary and secondary allocations.
4. The RECORDSIZE parameter specifies the average record size as 2500 bytes and the maximum size as 3000 bytes.

5. The VOLUMES parameter specifies that the cluster will reside on VSER01.
6. The NONINDEXED parameter specifies that the cluster is for an entry-sequenced file.
7. The CATALOG parameter specifies the name and password of the catalog in which the cluster will be defined.

Example 25: Defining a Dynamic File

This example defines a VSAM entry-sequenced file using the default volume capability and the REUSE and NOALLOCATION (dynamic file) options.

```
// JOB      EXAMPL25
// EXEC     IDCAMS,SIZE=AUTO
   DEFINE   CLUSTER -
            ( NAME(EXAMPLE.ESDS3) -
              RECORDSIZE(100 200) -
              RECORDS(1000 200) -
              NONINDEXED -
              REUSE -
              NOALLOCATION)
/*
/&
```

Explanation of Command

1. The CLUSTER parameter is required and NAME specifies the name of the cluster being defined.
2. The RECORDSIZE parameter specifies the average and maximum record sizes.
3. The RECORDS parameter specifies the amount of space to be allocated to the file when the file is accessed.
4. The NONINDEXED parameter is required to override the default (INDEXED).
5. The REUSE and NOALLOCATION parameters indicate that the file has the dynamic file capability.
6. The default volumes capability is effective as the default because VOLUMES was not specified. (In this case, a default model for an entry-sequenced file must exist in the catalog.)

Example 26: Accessing A Dynamic File

This example accesses the reusable VSAM entry-sequenced file defined in Example 25 using the DLBL statement's DISP= operand. The file is allocated at OPEN, loaded and kept (passed) at CLOSE in the first step of the job. The second step accesses the passed data and then deallocates (that is, sets the file back to the NOALLOCATION state) the file at CLOSE.

```
// JOB EXAMPL26
// DLBL FILE1,'EXAMPLE.ESDS3',,VSAM,DISP=(NEW,KEEP)
// EXEC PROG1,SIZE=AUTO
// DLBL FILE2,'EXAMPLE.ESDS3',,VSAM,DISP=(OLD,DELETE)
// EXEC PROG2,SIZE=AUTO
/&
```

Example 27: Defining a Partition Independent File

This example creates a VSAM entry-sequence file. The file name will be 'PART.INDEP.xx" where xx is the identifier of the partition in which the job is executed, that is, BG, F1, F2, etc. As specified in this example, the command does not provide all the parameters necessary for successful file definition. These parameters will be filled in by VSAM according to the values specified in the default model (DEFAULT.MODEL.ESDS) specified in Example 24.

```
// JOB      EXAMPLE@&
// EXEC     IDCAMS,SIZE=AUTO)
   DEFINE   CLUSTER -
            (NAME(%PART.INDEP) -
             NONINDEXED)
            CATALOG(AMASTCAT/MCATUPPW)
/*
/&
```

Explanation of Command

1. The CLUSTER parameter is required and NAME specifies the name of the cluster being defined.
2. The NONINDEXED parameter specifies that the cluster is for an entry-sequenced file.
3. The CATALOG parameter specifies the name and password of the catalog in which the cluster will be defined.

Appendix B: Interpreting LISTCAT Output Listings

The various options available in the LISTCAT command may be employed in numerous combinations, thus permitting you to tailor the LISTCAT output contents. This appendix provides information on the structure of LISTCAT output: the order in which entries are listed and the meanings of the listed fields.

In all cases, each entry listed is identified both by the *type* of entry (cluster, nonVSAM, etc.) and the *name* of the entry. Entries are usually listed in alphabetic order, according to the name of the entry. There are two exceptions to strict alphabetic ordering:

1. When the ENTRIES parameter is employed, the entries are listed in the order in which they are specified.
2. When the entry types are not restricted by use of one or more of the options (cluster, space, data, etc.), an entry that has *associated* entries is immediately followed by listings of the associated entries–thus a cluster's data, and possibly index, component is listed immediately following the cluster. A path is listed immediately following its pathentry.

This appendix is organized in eight parts:

- Entry type code used with Field Group Descriptions.
- Description of Keyword Fields
- LISTCAT and Access Method Services Output Messages
- LISTCAT Output Listing
- LISTCAT VOLUME Output Listing
- LISTCAT SPACE ALL Output Listing
- LISTCAT ALL Output Listing
- LISTCAT ALLOCATION Output Listing

Entry Type Code Used with Field Group Descriptions

Although the different entry types do not use all of the fields that may be listed (for example, a cluster entry does not have any allocation fields), there is a large set of fields, and groupings of fields, that are common to most entry types. Accordingly, the fields are discussed as a complete set. Each field, or field grouping, indicates the entry types to which they apply, as follows:

Code	Meaning
C	Cluster
D	Data
I	Index
G	Alternate Index
R	Path
U	User Catalog
V	Space (that is, a *volume* entry)
A	NonVSAM

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:

Group Names

Allocation Group
Associations Group
Attributes Group
Data Space Group
History Group
NonVSAM Entry, Special Field for
Protection Group
Statistics Group
Volumes Group
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.

The Allocation Group (D,I)

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. This is the highest of the high-used RBAs in the "Volumes Groups," with a possible exception in the case that the file was not successfully closed and has not been verified.

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 candidate volume.

SPACE-SEC — Gives the number of units (indicated under TYPE) of space to be allocated whenever a file (or key range within it) is extended beyond the primary space on this volume.

SPACE-TYPE — Indicates the unit of space allocation:

BLOCK—Blocks

CYLINDER—Cylinders

TRACK—Tracks

USECLASS-PRI—Indicates the primary useclass of a catalog, cluster, or alternate index.

USECLASS-SEC—Indicates the secondary useclass of a catalog, cluster, or alternate index.

The Associations Group (G,R,C,D,I)

This grouping lists the types (cluster, data,etc.) and entry names of each entry associated with the present entry. A cluster or alternate index entry will indicate its associated path entries and data and index (if a key-sequenced file) entries. Similarly, an index or data entry will indicate its associated cluster or alternate index of which it is a component.

- 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 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 establishes the connection to the base cluster) 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.

AIX — Identifies an alternate-index entry.

CLUSTER — Identifies a cluster entry.

DATA — Identifies a data entry.

INDEX — Identifies an index entry.

NONVSAM — Identifies a nonVSAM file entry.

PATH — Identifies a path entry.

UCAT — Identifies a user catalog entry.

The Attributes Group (D,I,G,R)

The fields in this group describe the miscellaneous attributes of 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 — Indicates, for an AIX, the alternate index Relative Key Position. The offset, from the beginning of the base cluster's data record, at which the alternate-key field begins.

BUFSPACE — The minimum buffer space in virtual storage to be provided by a processing program.

CIFORMAT — See *Using the VSE/VSAM Space Management for SAM Feature.*

CI/CA — The number of control intervals per control area. (If this field contains 0, see *Using the VSE/VSAM Space Management for SAM Feature.)*

CISIZE — The size, in bytes, of the entry's control intervals (see the CONTROLINTERVALSIZE parameter—this listed value is the size which either was specified in the parameter or was computed when the entry was defined). (If this field contains 0, see *Using the VSE/VSAM Space Management for SAM Feature.)*

ERASE — Records are to be erased (set to binary 0s) when deleted.

EXCPEXIT — The name of the object's exception exit routine.

EXP-DEFINE — See *Using the VSE/VSAM Space Management for SAM Feature.*

IMBED — The sequence-set index record is stored along with its associated data control area.

IMP-DEFINE — See *Using the VSE/VSAM Space Management for SAM Feature.*

INH-UPDATE — The data component cannot be updated. Either the data component was exported with INHIBITSOURCE specified, or exported and imported with INHIBITTARGET 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-possible relative-record number that can be addressed for this relative-record file. This number depends on the control-interval size and record size specified for the file.

NOALLOC — No space is allocated to the file, (that is, the file has the potential to have no space; it may have space if it is dynamic).

NOCIFORMAT — See *Using the VSE/VSAM Space Management for SAM Feature.*

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 — For an AIX more than one data record in the base cluster can contain the same alternate-key value.

NOREPLICAT — Index records are not replicated.

NOREUSE — The file cannot be reused.

NOTUSABLE — The entry is not usable either because the catalog could not be correctly recovered (see RESETCAT IGNORE option) or because space or critical volumes allocated for the file has been deleted by the DELETE FORCE option or by ALTER REMOVEVOLUMES.

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.

NOWRITECHK — Write operations are not checked for media recording correctness.

NUMBERED — The cluster is a relative-record file.

ORDERED — Volumes are used for space allocation in the order they were specified when the cluster/alternate index was defined.

RECFORMAT — See *Using the VSE/VSAM Space Management for SAM Feature.*

RECOVERY — A temporary CLOSE is issued as each control area of the file is loaded, so the whole file 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 file.

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 file can be reused (that is, its contents are temporary and its high-used RBA can be reset to 0 each time it is opened).

RKP — Relative Key Position, the offset from the beginning of the logical record, for this entry itself, at which the key field begins. Implicitly defined by VSAM for an alternate index. For a cluster, see the KEYS parameter.

SAMDATASET — See *Using the VSE/VSAM Space Management for SAM Feature.*

SAMLRECL — See *Using the VSE/VSAM Space Management for SAM Feature.*

SHROPTNS — *(n,m)* The numbers n and m identify the types of sharing permitted. See SHAREOPTIONS in the DEFINE CLUSTER or DEFINE ALTERNATEINDEX sections 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 file has been loaded.

SUBALLOC — More than one VSAM cluster or alternate index can share the data space. A VSAM catalog might also occupy the data space.

TEMP-EXP — Indicates that the TEMPORARY option was employed when this file was exported (see EXPORT command). This field is omitted should the file not be temporarily exported.

UNIQUE — Only one VSAM cluster or alternate index can occupy the data space—the cluster or alternate index is unique.

UNIQUEKEY — For an AIX 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.

VSAMDATSET — Indicates that this is a VSAM entry-sequenced file (not a SAM ESDS file).

WRITECHECK — Write operations are checked for correctness.

The Data Space Group (V)

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 by a secondary allocation operation. If a VSAM file must be given additional space on a volume and all existing data spaces are full, an existing data space will be extended, or a new one allocated on the volume, using catalog DADSM. A new data space so allocated is known as an implicit or automatic data space (this attribute is not utilized by DOS/VSE).

CLASS—Indicates the class of space on the storage volume.

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, or by a DEFINE SPACE.

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 file component.

DATASET DIRECTORY — Lists the VSAM files that can be stored (see CANDIDATE below) or actually are stored (in whole or in part) in the data space.

ATTRIBUTES — Describes the relation between the named file and the data space.

CANDIDATE—The volume is a candidate for storing the file

(NULL)—The file is stored (in whole or in part) in the data space

DSN — The file or data set name of the object that can be stored or is stored on the volume.

EXTENTS — The number of file suballocated extents for the file within the data space.

DATASETS — The number of VSAM files stored (in whole or in part) in the data space. (The number does not include files for which the volume is a candidate.)

EXTENT-DESCRIPTOR — Describes the data space extent.

BEG-BLOCK—The device address (in decimal) of the extents (FBA devices only).

BEG-CCHH — The device address (that is, CC = cylinder and HH = track) of the extent.

BLOCKS-TOTAL—The total number of blocks (in decimal) allocated to the data space (FBA devices only).

BLOCKS-USED—The number of blocks (in decimal) allocated to files and catalogs (FBA devices only).

SPACE-MAP (for CKD devices) — 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).

SPACE-MAP (for FBA devices)—A hexadecimal number that tells what blocks are used and what blocks are free in the extent. Each RLC (run length code) is always four bytes long.

TRACKS-TOTAL— The total number of tracks (in decimal) allocated to the data space.

TRACKS-USED — The number of tracks (in decimal) allocated to files and catalogs.

EXTENTS — The number of file suballocated extents in the data space.

FORMAT-1-LABEL — Identifies the Format-1 label that describes the data space.

BLOCK—The address (in decimal) of the Format 1 label in the VTOC (FBA devices only).

CCHHR — The device address (that is, CC = cylinder, HH = track, and R = record number) of the Format-1 label in the Volume Table of Contents.

TIMESTAMP — The time the data space was allocated (time-of-day clock value). The Format-1 label contains the timestamp, as a part of the VSAM generated name.

SEC-ALLOC — Gives the number of units (indicated under TYPE) of space to be allocated whenever the data space is extended (not utilized by DOS/VSE).

TYPE — Indicates the unit of space allocation:

BLOCK—Blocks

CYLINDER—Cylinders

TRACK—Tracks

The History Group (C,D,G,I,R)

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) on 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 = DOS/VS 28, 29, and 30

2 = DOS/VS 31 to DOS/VS 34 and VSE

The NonVSAM Entry, Special Field For (A)

The special field for a nonVSAM file describes a nonVSAM file stored on magnetic tape.

FSEQN — The sequence number (for the tape volume indicated under the "VOLUMES group" keyword VOLSER) of the file on which the nonVSAM file is stored.

The Protection Group (C,D,G,I,R)

The fields in this group describe how the alternate index, cluster, data component, index component, or path defined by the entry is protected. NULL or SUPPRESSED might be listed under PROTECTION:

NULL indicates that the object defined by the entry has no passwords.

SUPP indicates that the master password of neither the catalog nor the entry was specified, so authority to list protection information is not granted.

ATTEMPTS — Gives the number of times (zero through seven) the console operator is allowed to attempt to enter a correct password.

CODE — Gives the one to eight character 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). This is the information specified in the *string* subparameter of the AUTH subparameter of the DEFINE command.

USVR — The name of the user-written program which is to be invoked to verify authorization of any access to the entry; known as the USVR-User Security Verification Routine.

The Statistics Group (D,I)

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 — For the data component of a KSDS or RRDS — actual number of bytes of unused control areas in the total amount of allocated space.

For the data component of an ESDS — actual number of the bytes of unused control intervals in the total amount of allocated space.

For the index component — actual number of bytes of unused control intervals in the total amount of allocated space.

INDEX — This field appears only in an index entry. The fields under it describe activity in the index component.

ENTRIES/SECT — The desired number of entries in each section of entries in an index record. Used by VSAM for further calculations.

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 file to which the index belongs.

SEQ-SET-RBA — A decimal field which contains the RBA of the first sequence-set record. The sequence set might be separated from the index set by some quantity of RBA space.

The remaining fields in the Statistics Group, except for the SYSTEM-TIMESTAMP field, are updated only when the file is closed.

Note: If the CLOSE disposition of the file results in the resetting or deallocation of the file, the statistics group will be zero.

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 inserted into the data component that have not been added to the end of the file.

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. Note that for relative record files this is the number of slots that have been formatted.

REC-UPDATED — The number of records that have been retrieved for update and rewritten. This value does not reflect those records which were deleted. A record that is updated and then deleted is still counted in the update statistics.

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 were deleted from the old control interval.

SYSTEM-TIMESTAMP — The time (time-of-day clock value) the data or index component was last closed (after being opened for operations that might have changed its contents).

The Volumes Group (D,I)

The fields in this group identify the volume(s) on which a data component, index component, user catalog, or nonVSAM file 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. The index component of a key-sequence file with the imbed attribute will have a minimum of two Volumes Groups, one for the imbedded sequence set and one for a high-level index. The extents for the imbedded sequence set will be the same as those for the data component.

BLOCKS/CA — The number of blocks (in decimal) which comprise a control area. (Applies to FBA devices only.)

BLKS/MIN-CA — The number of blocks (in decimal) that VSAM can write in a minimum control area unit on the volume. (Applies only to FBA devices.)

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. The extents can be contiguous.

80—A sequence set occupies a track adjacent to a control area.

EXTENTS — Gives the physical and relative-byte addresses of each extent. If extent-number is 0 (NOALLOC), these fields that follow (BLOCKS to TRACKS) are not printed.

BLOCKS — The number of blocks (in decimal) in the extent. (Applies to FBA devices only.)

HIGH-BLOCK — The device address of the end of the extent. (Applies to FBA devices only).

HIGH-CCHH — The device address (that is, CC = cylinder and HH = track) of the end of the extent.

HIGH-RBA — A decimal field containing the RBA (relative byte address) of the end of the extent.

LOW-BLOCK — The device address of the beginning of the extent. (Applies to FBA devices only).

LOW-CCHH — The device address (that is, CC = cylinder and HH = track) of the beginning of the extent.

LOW-RBA — A decimal field containing 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 file with the KEYRANGE attribute, the highest hexadecimal value allowed in the key field of a record in the key range. A maximum of 64 bytes can appear in HIGH-KEY.

HI-KEY-RBA[1] — For a key-sequenced file, the RBA (relative byte address) in decimal of the control interval on the volume that contains the highest-keyed record in the file or key range.

LOW-KEY[1] — For a key-sequenced file with the KEYRANGE attribute, the lowest hexadecimal value allowed in the key field of a record in the file 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

For the data component of a KSDS or RRDS — The highest RBA of a control area (plus 1) within space allocated to the component (or to a keyrange) that actually contains data.

For the data component of an ESDS — The highest RBA of a control interval (plus 1) that actually contains data.

For the index component — The highest RBA of a control interval (plus 1) within space allocated to the component (or to sequence set records of a keyrange) that actually contains index records.

TRACKS/CA — The number of tracks which comprise a control area. (This value is computed when the entry is defined, and reflects the optimum size of the control area for the given device type and the nature of the entry--whether indexed, non-indexed, or numbered.) For a key-sequenced file with the imbedded attribute, this value includes the sequence set track.

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, but which as yet has no space allocated upon it for the entry.

OVERFLOW — The volume is an overflow volume on which data records in a key range are stored. The keyrange begins on another (a PRIME) volume.

PRIME — The volume is the first volume on which data records in a key range are stored.

VOLSER — The one through six character volume serial.

---

[1] Multiple keyranges might reside on a single volume—the volumes group is repeated for each such keyrange field.

The Volume Entry, Special Fields For (V)

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.

BLKS/MAX-CA — The number of blocks (in decimal) in the largest control area on the volume (FBA devices only).

BLKS/MIN-CA — The number of blocks (in decimal) that VSAM can use in a minimum control area unit on the volume (FBA devices only).

BLOCKS/VOL — The number of blocks (in decimal) that VSAM can use on the volume. This does not include alternate blocks reserved for error recovery (FBA devices only).

BYTES/TRK — The number of bytes that VSAM can use on each track on the volume, including alternate track cylinders.

CYLS/VOL — The number of cylinders that VSAM can use on the volume, including alternate track cylinders.

DATASETS-ON-VOL — The number of VSAM clusters that reside, in whole or in part, on the volume. If the number of data spaces is zero, then this is a candidate volume only and the Data Space grouping is omitted. (The number includes files for which the volume is a candidate.)

DATASPCS-ON-VOL — The number of VSAM data spaces on the volume. If the number of data spaces is zero, then this is a candidate volume only and the data space grouping is omitted.

DEVTYPE — The type of device to which this volume belongs.

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 suballocated on the volume for a single file.

TRKS/CYL — The number of tracks in each cylinder on the volume.

VOLUME-TIMESTAMP — The time (time-of-day clock value) VSAM last changed the contents of the volume. The Format-4 label 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.

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 *VSE/VSAM Messages and Codes.* When your LISTCAT job completes successfully, Access Method Services provides messages that follow the entry listing (see Figure B-1).

The first line identifies the catalog which contains the listed entries.

The next group of lines specifies 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.

```
            LISTING FROM CATALOG -- MJKCAT
    THE NUMBER OF ENTRIES PROCESSED WAS:
          AIX ---------------------1
          CLUSTER -----------------2
          DATA --------------------3
          INDEX -------------------3
          NONVSAM -----------------1
          PATH --------------------1
          SPACE -------------------2
          USERCATALOG -------------0
          TOTAL -------------------13
    THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure B-1. Messages that Follow the Entry Listing

LISTCAT Output Listing

When you specify LISTCAT without any parameters, the entryname and type of each entry is listed (see Figure B-2). 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.

```
/* A:   LIST ENTRYNAMES OF THE JOB CATALOG */
*   LISTCAT -
    CATALOG (MJKCAT)

             LISTING FROM CATALOG -- MJKCAT
VOLUME -------- FBA001
AIX ------------ MJK.ALT.INDEX1
   DATA ------- T1E628A0.VSAMDSET.DFD79226.T8ED1AE4.T1E628A0
   INDEX ------ T1E6F2A0.VSAMDSET.DFD79226.T8ED1AE4.T1E6F2A0
   PATH ------- MJK.AIX1.PATH
CLUSTER ------- MJK.CLUSTER1
   DATA ------- TBA10D20.VSAMDSET.DFD79226.T8ED1ADE.TBA10D20
   INDEX ------ TBAA8330.VSAMDSET.DFD79226.T8ED1ADE.TBAA8330
NONVSAM ------- MJK.NONVSAM1
CLUSTER ------- MJKCAT
   DATA ------- VSAM.CATALOG.BASE.DATA.RECORD
   INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
VOLUME -------- 111111
             LISTING FROM CATALOG -- MJKCAT
     THE NUMBER OF ENTRIES PROCESSED WAS:
           AIX --------------------1
           CLUSTER ----------------2
           DATA -------------------3
           INDEX ------------------3
           NONVSAM ----------------1
           PATH -------------------1
           SPACE ------------------2
           USERCATALOG ------------0
           TOTAL ------------------13
     THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure B-2. 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 B-3).

```
/* B:   LIST VOLUMES FOR SELECTED ENTRIES  */
   LISTCAT -
    VOLUME

              LISTING FROM CATALOG -- MJKCAT

VOLUME --------------- FBA001
   HISTORY
     RELEASE----------------2
   VOLUMES
     VOLSER------------FBA001   DEVTYPE--------------FBA
AIX ---------- MJK.ALT.INDEX1
   HISTORY
     OWNER-IDENT-------(NULL)   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------00.000

   DATA ------- T1E628A0.VSAMDSET.DFD79226.T8ED1AE4.T1E628A0
   HISTORY
     OWNER-IDENT-------(NULL)   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------00.000
   VOLUMES
     VOLSER------------FBA001   DEVTYPE--------------FBA

   INDEX ------ T1E6F2A0.VSAMDSET.DFD79226.T8ED1AE4.T1E6F2A0
   HISTORY
     OWNER-IDENT-------(NULL)   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------00.000
   VOLUMES
     VOLSER------------FBA001   DEVTYPE--------------FBA

   PATH ------- MJK.AIX1.PATH
   HISTORY
     OWNER-IDENT-------(NULL)   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------00.000

CLUSTER ------- MJK.CLUSTER1
   HISTORY
     OWNER-IDENT-----OWNCLUST   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------79.326

   DATA ------- TBA10D20.VSAMDSET.DFD79226.T8ED1ADE.TBA10D20
   HISTORY
     OWNER-IDENT-------(NULL)   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------79.326
   VOLUMES
     VOLSER------------FBA001   DEVTYPE--------------FBA

   INDEX ------ TBAA8330.VSAMDSET.DFD79226.T8ED1ADE.TBAA8330
   HISTORY
     OWNER-IDENT-------(NULL)   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------79.326
   VOLUMES
     VOLSER------------FBA001   DEVTYPE--------------FBA

NONVSAM ------- MJK.NONVSAM1
   HISTORY
     RELEASE----------------2
   VOLUMES
     VOLSER------------335001   DEVTYPE-------------3350

CLUSTER ------- MJKCAT
   HISTORY
     OWNER-IDENT-------(NULL)   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------99.999

   DATA ------- VSAM.CATALOG.BASE.DATA.RECORD
   HISTORY
     OWNER-IDENT-------(NULL)   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------99.999
   VOLUMES
     VOLSER------------111111   DEVTYPE-------------3330

   INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
   HISTORY
     OWNER-IDENT-------(NULL)   CREATION----------79.226
     RELEASE----------------2   EXPIRATION--------99.999
   VOLUMES
     VOLSER------------111111   DEVTYPE-------------3330

VOLUME -------- 111111
   HISTORY
     RELEASE----------------2
   VOLUMES
     VOLSER------------111111   DEVTYPE-------------3330
```

Figure B-3. An Example of LISTCAT VOLUME Output (Part 1 of 2)

```
            LISTING FROM CATALOG -- MJKCAT
     THE NUMBER OF ENTRIES PROCESSED WAS:
           AIX --------------------1
           CLUSTER ----------------2
           DATA -------------------3
           INDEX ------------------3
           NONVSAM ----------------1
           PATH -------------------1
           SPACE ------------------2
           USERCATALOG ------------0
           TOTAL ------------------13
     THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure B-3. An Example of LISTCAT VOLUME Output (Part 2 of 2)

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 B-4). You can use this type of listing to determine how much 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 ------- FBA001
   HISTORY
     RELEASE----------------2
   CHARACTERISTICS
     BLK/MIN-CA------------11   DEVTYPE -------------FBA                              DATASETS-ON-VOL-------4
     BLK/MAX-CA-----------132   VOLUME-TIMESTAMP           MAX-EXT/ALLOC-----------5   DATASPCS-ON-VOL-------1
     BLOCKS/VOL---------91608      X'8ED1AD13DF304000'
   DATASPACE
     DATASETS---------------4   FORMAT-1LABEL              ATTRIBUTES
     EXTENTS----------------1   RRNUM-----------------3    SUBALLOC
     SEC-ALLOC--------------0   TIMESTAMP                  EXPLICIT
     TYPE---------------BLOCK      X'8ED1AD13DF304000'
     CLASS------------------1
     EXTENT-DESCRIPTOR
     BLOCKS-TOTAL--------4180   BEG-BLOCK------------506   SPACE-MAP-----0000006E00000FE6
     BLOCKS-USED----------110
     DATASET-DIRECTORY
      DSN---TBA10D20.VSAMDSET.DFD79226.T8ED1ADE.TBA10D20   ATTRIBUTES--------(NULL)   EXTENTS-----------------1
      DSN---TBAAB330.VSAMDSET.DFD79226.T8ED1ADE.TBAA8330   ATTRIBUTES--------(NULL)   EXTENTS-----------------1
      DSN---T1E628A0.VSAMDSET.DFD79226.T8ED1AE4.T1E628A0   ATTRIBUTES--------(NULL)   EXTENTS-----------------1
      DSN---T1E6F2A0.VSAMDSET.DFD79226.T8ED1AE4.T1E6F2A0   ATTRIBUTES--------(NULL)   EXTENTS-----------------1
VOLUME -------- 111111
   HISTORY
     RELEASE----------------2
   CHARACTERISTICS
     BYTES/TRK----------13165   DEVTYPE-------------3330   MAX-PHYREC-SZ------13030   DATASETS-ON-VOL--------1
     TRKS/CYL--------------19   VOLUME-TIMESTAMP           MAX-EXT/ALLOC----------5   DATASPCS-ON-VOL--------1
     CYLS/VOL-------------411      X'8ED1AC2FB51C4000'
   DATASPACE
     DATASETS---------------1   FORMAT-1-LABEL             ATTRIBUTES
     EXTENTS----------------1   CCHHR------X'0027000003'   SUBALLOC
     SEC-ALLOC-------------19   TIMESTAMP                  EXPLICIT
     TYPE---------------TRACK      X'8ED1AC2FB51C4000'     USERCAT
     CLASS------------------1
     EXTENT-DESCRIPTOR
     TRACKS-TOTAL---------190   BEG-CCHH-----X'00030000'   SPACE-MAP---------------4B73
     TRACKS-USED-----------75
     DATASET-DIRECTORY
     DSN----MJKCAT                                         ATTRIBUTES--------(NULL)   EXTENTS-----------------3
              LISTING FROM CATALOG -- MJKCAT
     THE NUMBER OF ENTRIES PROCESSED WAS
           AIX ------------------0
           CLUSTER --------------0
           DATA -----------------0
           INDEX ----------------0
           NONVSAM --------------0
           PATH -----------------0
           SPACE ----------------2
           USERCATALOG ----------0
           TOTAL ----------------2
     THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure B-4. 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 B-5). 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, then 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
                LISTING FROM CATALOG -- MJKCAT
VOLUME-----------FBA001
   HISTORY
     RELEASE----------------2
   CHARACTERISTICS
     BLKS/MIN-CA------------11   DEVTYPE--------------FBA                              DATASETS-ON-VOL--------4
     BLKS/MAX-CA----------132    VOLUME-TIMESTAMP          MAX-EXT/ALLOC-----------5   DATASPCS-ON-VOL--------1
     BLOCKS/VOL---------91608       X'8ED1AD13DF304000'
   DATASPACE
     DATASETS---------------4    FORMAT-1-LABEL            ATTRIBUTES
     EXTENTS----------------1    RRNUM-------------------3 SUBALLOC
     SEC-ALLOC--------------0    TIMESTAMP                 EXPLICIT
     TYPE---------------BLOCK       X'8ED1AD13DF304000'
     CLASS------------------1
     EXTENT-DESCRIPTOR
     BLOCKS-TOTAL--------4180    BEG-BLOCK------------506  SPACE-MAP--------0000006E00000FE6
     BLOCKS-USED----------110
     DATASET-DIRECTORY
      DSN----TBA10D20.VSAMDSET.DFD79226.T8ED1ADE.TBA10D20  ATTRIBUTES---------(NULL)  EXTENTS----------------1
      DSN----TBAA8330.VSAMDSET.DFD79226.T8ED1ADE.TBAA8330  ATTRIBUTES---------(NULL)  EXTENTS----------------1
      DSN----T1E628A0.VSAMDSET.DFD79226.T8ED1AE4.T1E628A0  ATTRIBUTES---------(NULL)  EXTENTS----------------1
      DSN----T1E6F2A0.VSAMDSET.DFD79226.T8ED1AE4.T1E6F2A0  ATTRIBUTES---------(NULL)  EXTENTS----------------1
AIX ----------- MJK.ALT.INDEX1
   HISTORY
     OWNER-IDENT--------(NULL)   CREATION-----------79.226
     RELEASE----------------2    EXPIRATION--------00.000
   PROTECTION----------(NULL)
   ASSOCIATIONS
     DATA-----T1E628A0.VSAMDSET.DFD79226.T8ED1AE4.T1E628A0
     INDEX----T1E6F2A0.VSAMDSET.DFD79226.T8ED1AE4.T1E6F2A0
     CLUSTER--MJK.CLUSTER1
     PATH-----MJK.AIX1.PATH
   ATTRIBUTES
     UPGRADE
   DATA ------- T1E628A0.VSAMDSET.DFD79226.T8ED1ADE.T1E628A0
   HISTORY
     OWNER-IDENT--------(NULL)   CREATION-----------79.226
     RELEASE----------------2    EXPIRATION--------00.000
   PROTECTION----------(NULL)
   ASSOCIATIONS
     AIX-------MJK.ALT.INDEX1
   ATTRIBUTES
     KEYLEN-----------------5    AVGLRECL------------4086  BUFSPACE------------8704   CISIZE--------------4096
     RKP--------------------5    MAXLRECL------------32600 EXCPEXIT----------(NULL)   CI/CA-----------------8
     AXRKP-----------------45
     SHROPTNS(1,3)   RECOVERY    SUBALLOC          NOERASE INDEXED        NOWRITECHK   NOIMBED        NOREPLICAT
     UNORDERED        NOREUSE    SPANNED        NONUNIQKEY
   STATISTICS
     REC-TOTAL--------------1    SPLITS-CI--------------0  EXCPS-------------------4
     REC-DELETED------------0    SPLITS-CA--------------0  EXTENTS-----------------1
     REC-INSERTED-----------0    FREESPACE-%CI----------0  SYSTEM-TIMESTAMP:
     REC-UPDATED------------0    FREESPACE-%CA----------0     X'8ED1AE52BED16000'
     REC-RETRIEVED----------0    FREESPC-BYTES----------0
   ALLOCATION
     SPACE-TYPE---------BLOCK
     SPACE-PRI-------------66    USECLASS-PRI-----------1  HI-USED-RBA--------32768
     SPACE-SEC-------------66    USECLASS-SEC-----------1  HI-USED-RBA--------32768
   VOLUME
     VOLSER------------FBA001                             HI-ALLOC-RBA-------32768   EXTENT-NUMBER----------1
     DEVTYPE--------------FBA    BLOCKS/MIN-CA---------11  HI-USED-RBA--------32768   EXTENT-TYPE--------X'00'
     VOLFLAG-----------PRIME     BLOCKS/CA-------------66
     EXTENTS
     LOW-BLOCK------------539    LOW-RBA----------------0  BLOCKS------------------66
     HIGH-BLOCK-----------604    HIGH-RBA-----------32767
```

Figure B-5. An Example of LISTCAT ALL Output (Part 1 of 4)

```
   INDEX ------ T1E6F2A0.VSAMDSET.DFD79226.T8ED1AE4.T1E6F2A0
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------00.000
   PROTECTION----------(NULL)
   ASSOCIATIONS
     AIX------MJK.ALT.INDEX1
   ATTRIBUTES
     KEYLEN-----------------5    AVGLRECL---------------0    BUFSPACE---------------0    CISIZE---------------512
     RKP--------------------5    MAXLRECL-------------505    EXCPEXIT----------(NULL)    CI/CA-----------------11
     SHROPTNS(1,3)    RECOVERY   SUBALLOC          NOERASE   NOWRITECHK        NOIMBED   NOREPLICAT      UNORDERED
     NOREUSE
   STATISTICS
     REC-TOTAL--------------1    SPLITS-CI--------------0    EXCPS------------------4    INDEX
     REC-DELETED------------0    SPLITS-CA--------------0    EXTENTS----------------1    LEVELS----------------1
     REC-INSERTED-----------0    FREESPACE-%CI----------0    SYSTEM-TIMESTAMP            ENTRIES/SECT----------2
     REC-UPDATED------------0    FREESPACE-%CA----------0       X'8ED1AE532123C000'      SEQ-SET-RBA-----------0
     REC-RETRIEVED----------0    FREESPC-BYTES-------5120                                HI-LEVEL-RBA----------0
   ALLOCATION
     SPACE-TYPE---------BLOCK
     SPACE-PRI-------------11    USECLASS-PRI-----------1    HI-ALLOC-RBA--------5632
     SPACE-SEC-------------11    USECLASS-SEC-----------1    HI-USED-RBA----------512
   VOLUME
     VOLSER------------FBA001                               HI-ALLOC-RBA--------5632    EXTENT-NUMBER----------1
     DEVTYPE--------------FBA    BLKS/MIN-CA-----------11    HI-USED-RBA----------512    EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    BLOCKS/CA-------------11
     EXTENTS
     LOW-BLOCK------------605    LOW-RBA----------------0    BLOCKS----------------11
     HIGH-BLOCK-----------615    HIGH-RBA------------5631
   PATH ------- MJK.AIX1.PATH
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------00.000
   PROTECTION----------(NULL)
   ASSOCIATIONS
     AIX------MJK.ALT.INDEX1
     DATA-----T1E628A0.VSAMDSET.DFD79226.T8ED1AE4.T1E628A0
     INDEX----T1E6F2A0.VSAMDSET.DFD79226.T8ED1AE4.T1E6F2A0
     DATA-----TBA10D20.VSAMDSET.DFD79226.T8ED1ADE.TBA10D20
     INDEX----TBAA8330.VSAMDSET.DFD79226.T8ED1ADE.TBAA8330
   ATTRIBUTES
     UPDATE
CLUSTER ------- MJK.CLUSTER1
   HISTORY
     OWNER-IDENT-----OWNCLUST    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------79.326
   PROTECTION
     MASTERPW----------MASTCL    UPDATEPW----------UPDCL    CODE--------------CODECL
     CONTROLPW---------CNTLCL    READPW------------READCL   ATTEMPTS----------------3    USVR--------------(NULL)
     USAR--------------(NONE)
   ASSOCIATIONS
     DATA-----TBA10D20.VSAMDSET.DFD79226.T8ED1ADE.TBA10D20
     INDEX----TBAA8330.VSAMDSET.DFD79226.T8ED1ADE.TBAA8330
     AIX------MJK.ALT.INDEX1
   DATA ------- TBA10D20.VSAMDSET.DFD79226.T8ED1ADE.TBA10D20
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------79.326
   PROTECTION----------(NULL)
   ASSOCIATIONS
     CLUSTER--MJK.CLUSTER1
   ATTRIBUTES
     KEYLEN-----------------5    AVGLRECL-------------80    BUFSPACE------------4608    CISIZE--------------2048
     RKP--------------------5    MAXLRECL------------100    EXCPEXIT--------EXITCLUS    CI/CA-----------------5
     SHROPTNS(1,3)    RECOVERY   SUBALLOC          NOERASE   INDEXED       NOWRITECHK    NOIMBED       NOREPLICAT
     UNORDERED         NOREUSE   NONSPANNED
   STATISTICS
     REC-TOTAL-------------20    SPLITS-CI--------------0    EXCPS------------------5
     REC-DELETED------------0    SPLITS-CA--------------0    EXTENTS----------------1
     REC-INSERTED-----------0    FREESPACE-%CI---------15    SYSTEM-TIMESTAMP
     REC-UPDATED------------0    FREESPACE-%CA---------20       X'8ED1AE07E536D000'
     REC-RETRIEVED---------20    FREESPC-BYTES----------0
   ALLOCATION
     SPACE-TYPE---------BLOCK
     SPACE-PRI-------------22    USECLASS-PRI-----------1    HI-ALLOC-RBA-------10240
     SPACE-SEC-------------22    USECLASS-SEC-----------1    HI-USED-RBA--------10240
   VOLUME
     VOLSER------------FBA001                               HI-ALLOC-RBA-------10240    EXTENT-NUMBER----------1
     DEVTYPE--------------FBA    BLKS/MIN-CA-----------11    HI-USED-RBA--------10240    EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    BLOCKS/CA-------------22
     EXTENTS
     LOW-BLOCK------------506    LOW-RBA----------------0    BLOCKS----------------22
     HIGH-BLOCK-----------527    HIGH-RBA-----------10239
```

Figure B-5. An Example of LISTCAT ALL Output (Part 2 of 4)

```
   INDEX ------ TBAA8330.VSAMDSET.DFD79226.T8ED1ADE.TBAA8330
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------79.326
   PROTECTION----------(NULL)
   ASSOCIATIONS
     CLUSTER--MJK.CLUSTER1
   ATTRIBUTES
     KEYLEN-----------------5    AVGLRECL----------------0   BUFSPACE----------------0   CISIZE----------------512
     RKP--------------------5    MAXLRECL-------------505   EXCPEXIT--------EXITCLUS   CI/CA----------------11
     SHROPTNS(1,3)   RECOVERY    SUBALLOC          NOERASE   NOWRITECHK        NOIMBED   NOREPLICAT      UNORDERED
     NOREUSE
   STATISTICS
     REC-TOTAL--------------1    SPLITS-CI---------------0   EXCPS-------------------5   INDEX
     REC-DELETED------------0    SPLITS-CA---------------0   EXTENTS-----------------1   LEVELS------------------1
     REC-INSERTED-----------0    FREESPACE-%CI-----------0   SYSTEM-TIMESTAMP            ENTRIES/SECT------------2
     REC-UPDATED------------0    FREESPACE-%CA-----------0      X'8ED1AE09502E9000'      SEQ-SET-RBA-------------0
     REC-RETRIEVED----------0    FREESPC-BYTES-------5120                               HI-LEVEL-RBA------------0
   ALLOCATION
     SPACE-TYPE---------BLOCK
     SPACE-PRI-------------11    USECLASS-PRI------------1   HI-ALLOC-RBA--------5632
     SPACE-SEC-------------11    USECLASS-SEC------------1   HI-USED-RBA----------512
   VOLUME
     VOLSER------------FBA001                                HI-ALLOC-RBA--------5632   EXTENT-NUMBER-----------1
     DEVTYPE--------------FBA    BLKS/MIN-CA-------------11  HI-USED-RBA----------512   EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    BLOCKS/CA---------------11
     EXTENTS
     LOW-BLOCK-------------528   LOW-RBA-----------------0   BLOCKS-----------------11
     HIGH-BLOCK------------538   HIGH-RBA-------------5631

NONVSAM -------  MJK.NONVSAM1
   HISTORY
     RELEASE----------------2
   VOLUMES
     VOLSER------------335001    DEVTYPE-------------3350   FSEQN------------------0

CLUSTER-------MJKCAT
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------99.999
   PROTECTION----------(NULL)
   ASSOCIATIONS
     DATA-----VSAM.CATALOG.BASE.DATA.RECORD
     INDEX----VSAM.CATALOG.BASE.INDEX.RECORD

   DATA---------VSAM.CATALOG.BASE.DATA.RECORD
   HISTORY
     OWNER-IDENT--------(NULL)   CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------99.999
   PROTECTION----------(NULL)
   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
   STATISTICS
     REC-TOTAL--------------15   SPLITS-CI---------------0   EXCPS-------------------34
     REC-DELETED------------0    SPLITS-CA---------------0   EXTENTS-----------------2
     REC-INSERTED-----------0    FREESPACE-%CI-----------0   SYSTEM-TIMESTAMP
     REC-UPDATED------------0    FREESPACE-%CA-----------0      X'8ED1ACBC3C2BC000'
     REC-RETRIEVED----------0    FREESPC-BYTES-----450560
   ALLOCATION
     SPACE-TYPE---------TRACK
     SPACE-PRI-------------72    USECLASS-PRI------------1   HI-ALLOC-RBA------491520
     SPACE-SEC-------------18    USECLASS-SEC------------1   HI-USED-RBA-------471040
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512   HI-ALLOC-RBA------450560   EXTENT-NUMBER-----------1
     DEVTYPE-------------3330    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'00030000'    LOW-RBA-----------------0   TRACKS-----------------66
     HIGH-CCHH----X'00060008'    HIGH-RBA----------450559
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512   HI-ALLOC-RBA------491520   EXTENT-NUMBER-----------1
     DEVTYPE-------------3330    PHYRECS/TRK-----------20   HI-USED-RBA-------471040   EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    TRACKS/CA--------------3
     LOW-KEY----------------40
     HIGH-KEY--------------FF
     HI-KEY-RBA--------450560
     EXTENTS
     LOW-CCHH-----X'0006000C'    LOW-RBA-----------450560   TRACKS-----------------6
     HIGH-CCHH----X'00060011'    HIGH-RBA----------491519
```

Figure B-5. An Example of LISTCAT ALL Output (Part 3 of 4)

```
   INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------99.999
   PROTECTION----------(NULL)
   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
   STATISTICS
     REC-TOTAL--------------3    SPLITS-CI--------------0   EXCPS------------------38   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'8ED1ACC5C8F0C000'      SEQ-SET-RBA--------30720
     REC-RETRIEVED----------0    FREESPC-BYTES------41472                               HI-LEVEL-RBA-----------0
   ALLOCATION
     SPACE-TYPE---------TRACK
     SPACE-PRI--------------3    USECLASS-PRI-----------1   HI-ALLOC-RBA-------43008
     SPACE-SEC--------------3    USECLASS-SEC-----------1   HI-USED-RBA--------42496
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512   HI-ALLOC-RBA-------30720   EXTENT-NUMBER----------1
     DEVTYPE-------------3330    PHYRECS/TRK-----------20   HI-USED-RBA----------512   EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    TRACKS/CA--------------1
     EXTENTS
     LOW-CCHH-----X'00060009'    LOW-RBA----------------0   TRACKS-----------------3
     HIGH-CCHH----X'0006000B'    HIGH-RBA-----------30719
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512   HI-ALLOC-RBA-------41984   EXTENT-NUMBER----------1
     DEVTYPE-------------3330    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'00030000'    LOW-RBA------------30720   TRACKS-----------------66
     HIGH-CCHH----X'00060008'    HIGH-RBA-----------41983
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512   HI-ALLOC-RBA-------43008   EXTENT-NUMBER----------1
     DEVTYPE-------------3330    PHYRECS/TRK-----------20   HI-USED-RBA--------42496   EXTENT-TYPE--------X'80'
     VOLFLAG------------PRIME    TRACKS/CA--------------3
     LOW-KEY----------------40
     HIGH-KEY--------------FF
     EXTENTS
     LOW-CCHH-----X'0006000C'    LOW-RBA------------41984   TRACKS-----------------6
     HIGH-CCHH----X'00060011'    HIGH-RBA-----------43007
VOLUME -------- 111111
   HISTORY
     RELEASE----------------2
   CHARACTERISTICS
     BYTES/TRK----------13165    DEVTYPE----------3330      MAX-PHYREC-SZ------13030   DATASETS-ON-VOL--------1
     TRKS/CYL--------------19    VOLUME-TIMESTAMP           MAX-EXT/ALLOC----------5   DATASPCS-ON-VOL--------1
     CYLS/VOL-------------411       X'8ED1AC2FB51C4000'
   DATASPACE
     DATASETS---------------1    FORMAT-1-LABEL             ATTRIBUTES
     EXTENTS----------------1    CCHHR------X'0027000003'   SUBALLOC
     SEC-ALLOC-------------19    TIMESTAMP                  EXPLICIT
     TYPE---------------TRACK       X'8ED1AC2FB51C4000'      USERCAT
     CLASS------------------1
     EXTENT-DESCRIPTOR
     TRACKS-TOTAL----------190   REG-CCHH-----X'00030000'   SPACE-MAP-----------4B73
     TRACKS-USED-----------75
     DATASET-DIRECTORY
     DSN----MJKCAT                                          ATTRIBUTES--------(NULL)   EXTENTS----------------3




              LISTING FROM CATALOG -- MJKCAT
     THE NUMBER OF ENTRIES PROCESSED WAS
           AIX -------------------1
           CLUSTER ---------------2
           DATA ------------------3
           INDEX -----------------3
           NONVSAM ---------------1
           PATH ------------------1
           SPACE -----------------2
           USERCATALOG -----------0
           TOTAL ----------------13
     THE NUMBER OF PROTECTED ENTRIES SUPPRESSED WAS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Figure B-5. An Example of LISTCAT ALL Output (Part 4 of 4)

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 B-6). 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

             LISTING FROM CATALOG -- MJKCAT

AIX ----------- MJK.ALT.INDEX1
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------00.000

   DATA ------- T1E628A0.VSAMDSET.DFD79226.T8ED1AE4.T1E628A0
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------00.000
   ALLOCATION
     SPACE-TYPE---------BLOCK
     SPACE-PRI-------------66    USECLASS-PRI-----------1    HI-ALLOC-RBA-------32768
     SPACE-SEC-------------66    USECLASS-SEC-----------1    HI-USED-RBA--------32768
   VOLUME
     VOLSER------------FBA001                               HI-ALLOC-RBA-------32768    EXTENT-NUMBER----------1
     DEVTYPE--------------FBA    BLLS/MIN-CA-----------11    HI-USED-RBA--------32768    EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    BLOCKS/CA-------------66
     EXTENTS
     LOW-BLOCK------------539    LOW-RBA----------------0    BLOCKS-----------------66
     HIGH-BLOCK-----------604    HIGH-RBA-----------32767

   INDEX ------ T1E6F2A0.VSAMDSET.DFD79226.T8ED1AE4.T1E6F2A0
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------00.000
   ALLOCATION
     SPACE-TYPE---------BLOCK
     SPACE-PRI-------------11    USECLASS-PRI-----------1    HI-ALLOC-RBA--------5632
     SPACE-SEC-------------11    USECLASS-SEC-----------1    HI-USED-RBA----------512
   VOLUME
     VOLSER------------FBA001                               HI-ALLOC-RBA--------5632    EXTENT-NUMBER----------1
     DEVTYPE--------------FBA    BLKS/MIN-CA-----------11    HI-USED-RBA----------512    EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    BLOCKS/CA-------------11
     EXTENTS
     LOW-BLOCK------------605    LOW-RBA----------------0    BLOCKS-----------------11
     HIGH-BLOCK-----------615    HIGH-RBA------------5631
   PATH ------- MJK.AIX1.PATH
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------00.000

CLUSTER ------- MJK.CLUSTER1
   HISTORY
     OWNER-IDENT-----OWNCLUST    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------79.326

   DATA ------- TBA10D20.VSAMDSET.DFD79226.T8ED1ADE.TBA10D20
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------79.326
   ALLOCATION
     SPACE-TYPE---------BLOCK
     SPACE-PRI-------------22    USECLASS-PRI-----------1    HI-ALLOC-RBA-------10240
     SPACE-SEC-------------22    USECLASS-SEC-----------1    HI-USED-RBA--------10240
   VOLUME
     VOLSER------------FBA001                               HI-ALLOC-RBA--------10240   EXTENT-NUMBER----------1
     DEVTYPE--------------FBA    BLKS/MIN-CA-----------11    HI-USED-RBA--------10240    EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    BLOCKS/CA-------------22
     EXTENTS
     LOW-BLOCK------------506    LOW-RBA----------------0    BLOCKS-----------------22
     HIGH-BLOCK-----------527    HIGH-RBA-----------10239

   INDEX -------TBAA8330.VSAMDSET.DFD79226.T8ED1ADE.TBAA8330
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------79.326
   ALLOCATION
     SPACE-TYPE---------BLOCK
     SPACE-PRI-------------11    USECLASS-PRI-----------1    HI-ALLOC-RBA--------5632
     SPACE-SEC-------------11    USECLASS-SEC-----------1    HI-USED-RBA----------512
   VOLUME
     VOLSER------------FBA001                               HI-ALLOC-RBA--------5632    EXTENT-NUMBER----------1
     DEVTYPE--------------FBA    BLKS/MIN-CA-----------11    HI-USED-RBA----------512    EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    BLOCKS/CA-------------11
     EXTENTS
     LOW-BLOCK------------528    LOW-RBA----------------0    BLOCKS-----------------11
     HIGH-BLOCK-----------538    HIGH-RBA------------5631
```

Figure B-6. An Example of LISTCAT ALLOCATION Output (Part 1 of 2)

```
CLUSTER ------- MJKCAT
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------99.999
   DATA ------- VSAM.CATALOG.BASE.DATA.RECORD
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------99.999
   ALLOCATION
     SPACE-TYPE---------TRACK    USECLASS-PRI-----------1    HI-ALLOC-RBA------491520
     SPACE-PRI-------------72    USECLASS-SEC-----------1    HI-USED-RBA-------471040
     SPACE-SEC-------------18
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512    HI-ALLOC-RBA------450560    EXTENT-NUMBER----------1
     DEVTYPE-------------3330    PHYRECS/TRK----------20     HI-USED-RBA--------20480    EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    TRACKS/CA--------------3
     LOW-KEY---------------00
     HIGH-KEY--------------3F
     HIGH-KEY-RBA--------6144
     EXTENTS
     LOW-CCHH-----X'00030000'    LOW-RBA----------------0    TRACKS----------------66
     HIGH-CCHH----X'00060008'    HIGH-RBA----------450559
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512    HI-ALLOC-RBA------491520    EXTENT-NUMBER----------1
     DEVTYPE-------------3330    PHYRECS/TRK----------20     HI-USED-RBA-------471040    EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    TRACKS/CA--------------3
     LOW-KEY---------------40
     HIGH-KEY--------------FF
     HIGH-KEY-RBA------450560
     EXTENTS
     LOW-CCHH-----X'0006000C'    LOW-RBA----------450560     TRACKS----------------6
     HIGH-CCHH----X'00060011'    HIGH-RBA----------491519
   INDEX ------ VSAM.CATALOG.BASE.INDEX.RECORD
   HISTORY
     OWNER-IDENT-------(NULL)    CREATION----------79.226
     RELEASE----------------2    EXPIRATION--------99.999
   ALLOCATION
     SPACE-TYPE---------TRACK
     SPACE-PRI--------------3    USECLASS-PRI-----------1    HI-ALLOC-RBA-------43008
     SPACE-SEC--------------3    USECLASS-SEC-----------1    HI-USED-RBA--------42496
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512    HI-ALLOC-RBA-------30720    EXTENT-NUMBER----------1
     DEVTYPE-------------3330    PHYRECS/TRK----------20     HI-USED-RBA----------512    EXTENT-TYPE--------X'00'
     VOLFLAG------------PRIME    TRACKS/CA--------------1
     EXTENTS
     LOW-CCHH-----X'00060009'    LOW-RBA----------------0    TRACKS----------------3
     HIGH-CCHH----X'0006000B'    HIGH-RBA----------30719
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512    HI-ALLOC-RBA-------41984    EXTENT-NUMBER----------1
     DEVTYPE-------------3330    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'00030000'    LOW-RBA----------30720      TRACKS----------------66
     HIGH-CCHH----X'00060008'    HIGH-RBA----------41983
   VOLUME
     VOLSER------------111111    PHYREC-SIZE----------512    HI-ALLOC-RBA-------43008    EXTENT-NUMBER----------1
     DEVTYPE-------------3330    PHYRECS/TRK----------20     HI-USED-RBA--------42496    EXTENT-TYPE--------X'80'
     VOLFLAG------------PRIME    TRACKS/CA--------------3
     LOW-KEY---------------40
     HIGH-KEY--------------FF
     EXTENTS
     LOW-CCHH-----X'0006000C'    LOW-RBA----------41984      TRACKS----------------6
     HIGH-CCHH----X'00060011'    HIGH-RBA----------43007


            LISTING FROM CATALOG -- MJKCAT
     THE NUMBER OF ENTRIES PROCESSED WAS
           AIX -----------------1
           CLUSTER -------------2
           DATA ----------------3
           INDEX ---------------3
           NONVSAM -------------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 B-6. An Example of LISTCAT ALLOCATION Output (Part 2 of 2)

Appendix C: Interpreting LISTCRA Output Listings

The various options available in the LISTCRA command may be employed in five combinations, thus permitting you to tailor the LISTCRA output contents. This appendix provides information on the structure of LISTCRA output: the order in which entries are listed and the meanings of the listed fields.

Except for SEQUENTIALDUMP, each entry listed is identified both by the *type* of entry (cluster, nonVSAM, etc.) and the *name* of the entry. Entries are usually listed in alphabetic order, according to the name of the entry; however, if insufficient storage is available for the sorting operations, the records are simply listed as they appear in the catalog recovery area (CRA).

Except for SEQUENTIALDUMP, each type of listing is divided into three groups:

- VSAM entries (sorted), for example, clusters (CLUS), alternate indexes (AIX), and their associated entries.
- Other entries (sorted), such as user catalogs (UCAT) and nonVSAM (NONV).
- Unsorted entries.

The following is a list of entry and record abbreviations, types, and definitions:

Abbreviation	Type	Definition
AIX	G	Alternate index entry
CLUS	C	Cluster entry
DATA	D	Data entry of cluster or alternate index
FRSP	F	Free space record
INDX	I	Index entry of cluster or alternate index
NONV	A	NonVSAM entry
OEXT	E	Extension record
PATH	R	Path entry
UCAT	U	User catalog volume pointer entry in master catalog
UPGD	Y	Upgrade relationships to the data entry under which it is listed
VEXT	W	Volume extension record
VOL	V	Volume entry

Four of the various types of listings are (defaults are shown underlined):

1. NAME NOCOMPARE
2. DUMP NOCOMPARE
3. NAME COMPARE
4. DUMP COMPARE

If the defaults of NAME and NOCOMPARE are taken, all entry names, their volumes, and the names and volumes of all their related entries are listed. In the unsorted entries group, any records not yet printed out are printed.

If DUMP and NOCOMPARE are specified, all of the entry records and directly related entry records are listed in dump format. In addition, the names and volumes of indirectly related entries are also listed. In the unsorted entries group, the CRA's self-describing records (control intervals 0-8) are printed out plus any free space records or other records not yet printed out.

When NAME and COMPARE are specified, only the names of those entries that are not identical in the catalog and the CRA are listed. A

MISCOMPARE message is printed identifying the most severe level of comparison.

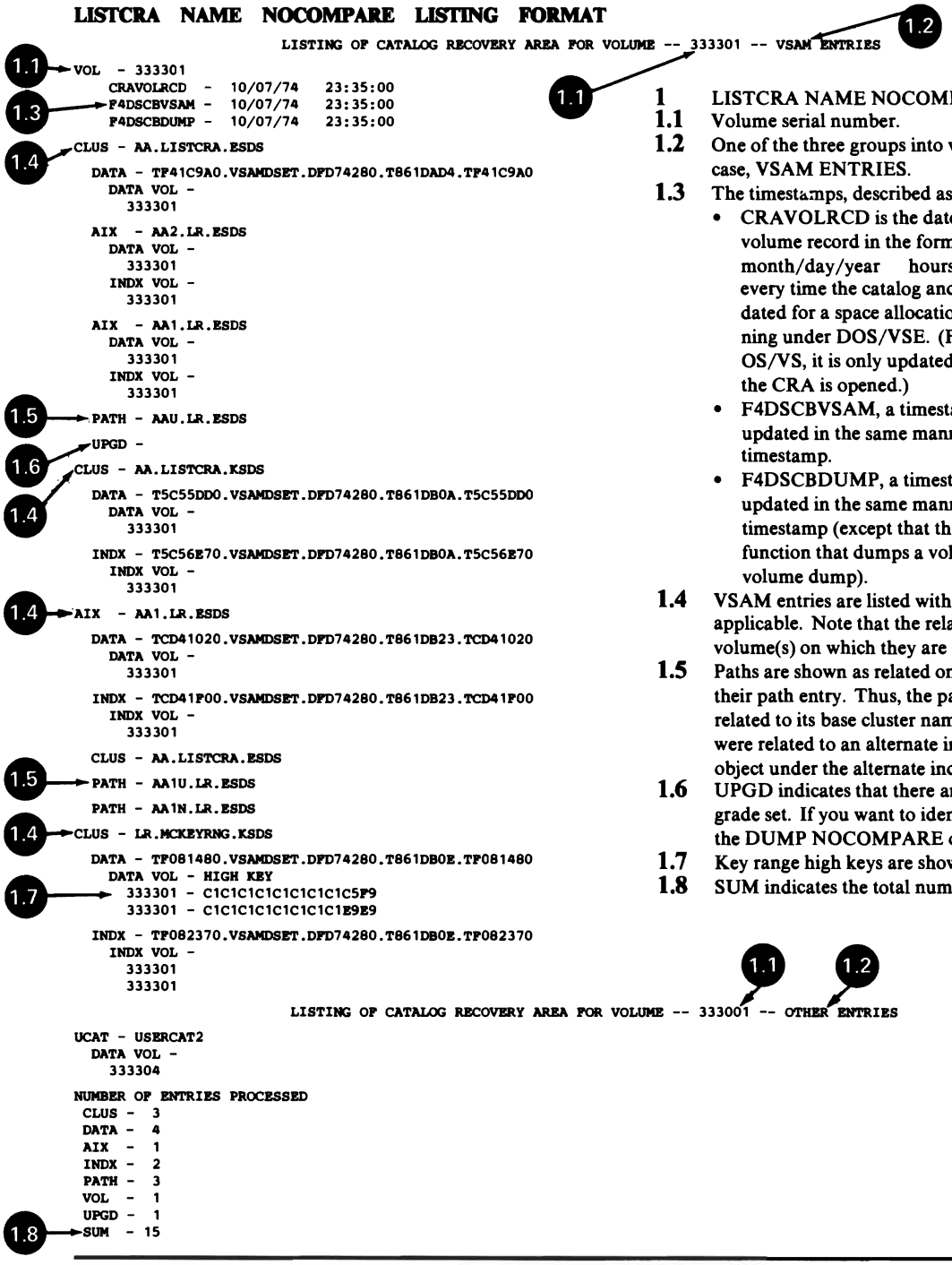
When DUMP and COMPARE are specified, only the records of those entries that are not identical in the catalog and the CRA are listed. The catalog records are listed on lines directly below the CRA records and the bytes that miscompare are identified by underlining. The MISCOMPARE severity message is also printed.

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 miscompares in the same listing and by examining related records in the catalog or CRA.

The fifth type of listing is obtained by specifying SEQUENTIALDUMP. In this case all of the records in the CRAs are listed sequentially in dump format. See the LISTCRA DUMP NOCOMPARE listing for format description.

See *VSE/VSAM VSAM Logic, Volume 1* for a complete description of CRA record formats. The following pages show partial formats for various types of listings that can be requested.

LISTCRA NAME NOCOMPARE LISTING FORMAT



1 LISTCRA NAME NOCOMPARE LISTING NOTES

1.1 Volume serial number.

1.2 One of the three groups into which the listing falls, in this case, VSAM ENTRIES.

1.3 The timestamps, described as follows:

- CRAVOLRCD is the date and time as found in the CRA volume record in the format of month/day/year hours:minutes:seconds. It is updated every time the catalog and CRA volume records are updated for a space allocation change when VSAM is running under DOS/VSE. (For VSAM running under OS/VS, it is only updated for the first CRA update after the CRA is opened.)
- F4DSCBVSAM, a timestamp in the format-4 label, is updated in the same manner as the CRA volume record timestamp.
- F4DSCBDUMP, a timestamp in the format-4 label, is updated in the same manner as the CRA volume record timestamp (except that the OS/VS utility IEHDASDR function that dumps a volume updates it at the time of the volume dump).

1.4 VSAM entries are listed with all of their related entries, where applicable. Note that the related entries are indented and the volume(s) on which they are contained is identified.

1.5 Paths are shown as related only to the entry that serves as their path entry. Thus, the path named AAU.LR.ESDS is related to its base cluster named AA.LISTCRA.ESDS. If it were related to an alternate index, it would appear as a related object under the alternate index (AIX).

1.6 UPGD indicates that there are alternate indexes in the upgrade set. If you want to identify these alternate indexes, use the DUMP NOCOMPARE options.

1.7 Key range high keys are shown where applicable.

1.8 SUM indicates the total number of entries processed.

LISTCRA DUMP NOCOMPARE LISTING FORMAT

```
                    LISTING OF CATALOG RECOVERY AREA FOR VOLUME -- 333301 -- VSAM ENTRIES
VOL  - 333301
000009 00000000 01F3F3F3 F3F0F100 00E  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 40FFFFFF            ...
  0060 FFFFFFFF FF74280F 00000F00 00000000 00070000 00C00000 00000201 0000602   ................................
  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  ..............-..-..............
  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...........
  AIX  - AIX01
    DATA VOL -
      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 00001CC9 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   .....h.....D..`-.`-.....
  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   .....(.....C.....G.0.....&.
  0120 09F3F3F3 F3F0F100 00800100 00000000  00000000 00280000 00020000 14000140   .333301.....(.....
  0140 00190000 00000014 00010003 00060003  00060001 00000000 000027FF 03273050   ..............  .....'0&
  0160 2009F3F3 F3F3F0F1 00004000 00000000  00000000 00000000 00000200 00140001   ..333301.. .....
  0180 40001900 00000000 00000000 00000000  00000000 00000000 00000000 00000000   ................................
```

2 LISTCRA DUMP NOCOMPARE LISTING NOTES

2.1 Three-byte CRA control interval number in hexadecimal.

2.2 Three-byte catalog control interval number in hexadecimal.

2.3 Entry type at X'2C', a character that identifies what type of entry is being listed.

2.4 A 44-byte field at X'31' that contains a volume serial number (for a volume) or an entry name.

2.5 Volume serial number for the recovery volume.

2.6 Two-byte hexadecimal displacement from the second to ending line of each record. Printing is suppressed after the last line containing data in the listing even though all records are 512 bytes long.

2.7 As in the NAME NOCOMPARE type of listing format, the alternate index and the fact that there is an upgrade set is given along with the volume serial numbers for both its data and index (not valid if SDUMP specified).

2.8 The first byte identifies the data object and the next three bytes, the catalog control interval number of the alternate index data component in the upgrade set.

2.9 The first byte identifies the index object and the next three bytes, the catalog control interval number of the alternate-index index component in the upgrade set.

2.10 Since there are two key ranges for this cluster, there are two volume serial numbers.

LISTCRA NAME COMPARE LISTING FORMAT

```
                    LISTING OF CATALOG RECOVERY AREA FOR VOLUME -- 333301 -- VSAM ENTRIES
3.1 ->  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
3.2 ->* MISCOMPARES - CATALOG ENTRY HAS DIFFERENT NAME

CLUS - LR.MCHURBA.ESDS

  DATA - T73FEDA0.VSAMDSET.DFD74280.T861DAE1.T73FEDA0
    DATA VOL -
      333301

* MISCOMPARES - HIGH USED RBA

CLUS - LR.MCKEYRNG.KSDS

  DATA - TF081480.VSAMDSET.DFD74280.T861DB0E.TF081480
    DATA VOL - HIGH KEY
      333301 - C1C1C1C1C1C1C1C1C5F9
      333301 - C1C1C1C1C1C1C1C1E9E9

* 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
3.3 ->IDC0877I NUMBER OF RECORDS THAT MISCOMPARED IN THIS CRA - 4
```

3 LISTCRA NAME COMPARE LISTING NOTES

3.1 An additional time stamp (CATVOLRCD) is listed, the one contained in the catalog volume record. This is updated in the same manner as the CRA volume record time stamp. The remainder of the time stamps are updated as described in Note 1.3.

3.2 The MISCOMPARES message always refers to the record listed above it. See "EXPORTRA/IMPORTRA: Recovering Catalog Entries and Data" and "LISTCRA: Analysis of Recoverable Catalogs" for information about "severity of miscompares" messages.

3.3 The number of records includes the number of extension records that miscompared in addition to the number of entries that miscompared.

LISTCRA DUMP COMPARE LISTING FORMAT

```
                    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 - T73FEDA0.VSAMDSET.DFD74280.T861DAE1.T73FEDA0

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....T73FEDA0.VSAMDS
       00000000 00000000 00000000 C4015700  8FE4F7F3 C6C5C4C1 F04BE5E2 C1D4C4E2   ............D....T73FEDA0.VSAMDS
  0040 C5E34BC4 C6C4F7F4 F2F8F04B E3F8F6F1  C4C1C5F1 4BE3F7F3 C6C5C4C1 F0FFFFFF   ET.DFD74280.T861DAE1.T73FEDA0...
       C5E34BC4 C6C4F7F4 F2F8F04B E3F8F6F1  C4C1C5F1 4BE3F7F3 C6C5C4C1 F0FFFFFF   ET.DFD74280.T861DAE1.T73FEDA0...
  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
```

4.1 → 000020 line

4.2 → CATRCD line

4.3 → * line

4.4 → MISCOMPARES-HIGH USED RBA

4 LISTCRA DUMP COMPARE LISTING NOTES

4.1 The 3-byte CRA control interval number in hexadecimal form.

4.2 Each two lines are, respectively, from the CRA and the catalog for the same displacements.

4.3 Asterisks are printed below the actual bytes in which a miscompare exists and, to the far left, the miscomparing lines are flagged in the margin with a single asterisk.

4.4 See "LISTCRA mismatch messages" for the cause and seriousness of this MISCOMPARES message.

Appendix D: Invoking Access Method Services from a Problem Program

Access Method Services can be invoked by a problem program through the use of the CDLOAD macro instruction.

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 files.

Invoking Macro Instructions

Access Method Services may be invoked from a problem program in DOS/VSE by loading the root segment of IDCAMS into virtual storage and then doing a branch entry to the module. To load IDCAMS, the program should be invoked with the SIZE=AUTO parameter on the EXEC statement, and should issue a CDLOAD macro of the form:

CDLOAD	*address*

where:

address
specifies the address of an 8-byte, left justified character string, 'IDCAMSbb'. CDLOAD returns the starting address of the module in Register 1.

The invoking program should branch to the address returned by CDLOAD plus 6.

Since IDCAMS uses OS/VS linkage conventions, the invoking program must provide an 18 fullword area to be used as a save area by Access Method Services. On entry to IDCAMS, Register 1 should point to the argument list discribed in Figure D-1, Register 13 should point to the save area, Register 14 must contain the return address, and Register 15 must contain the address of IDCAMS plus 6. On return, all registers except Register 15 are restored by Access Method Services. Register 15 contains the final return code from the processor. The following table contains the possible values of Register 15:

Code	Meaning
0	The function was executed as directed 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.

Figure D-1 describes the argument list as it exists in the user's area which is passed to the Access Method Services processor.

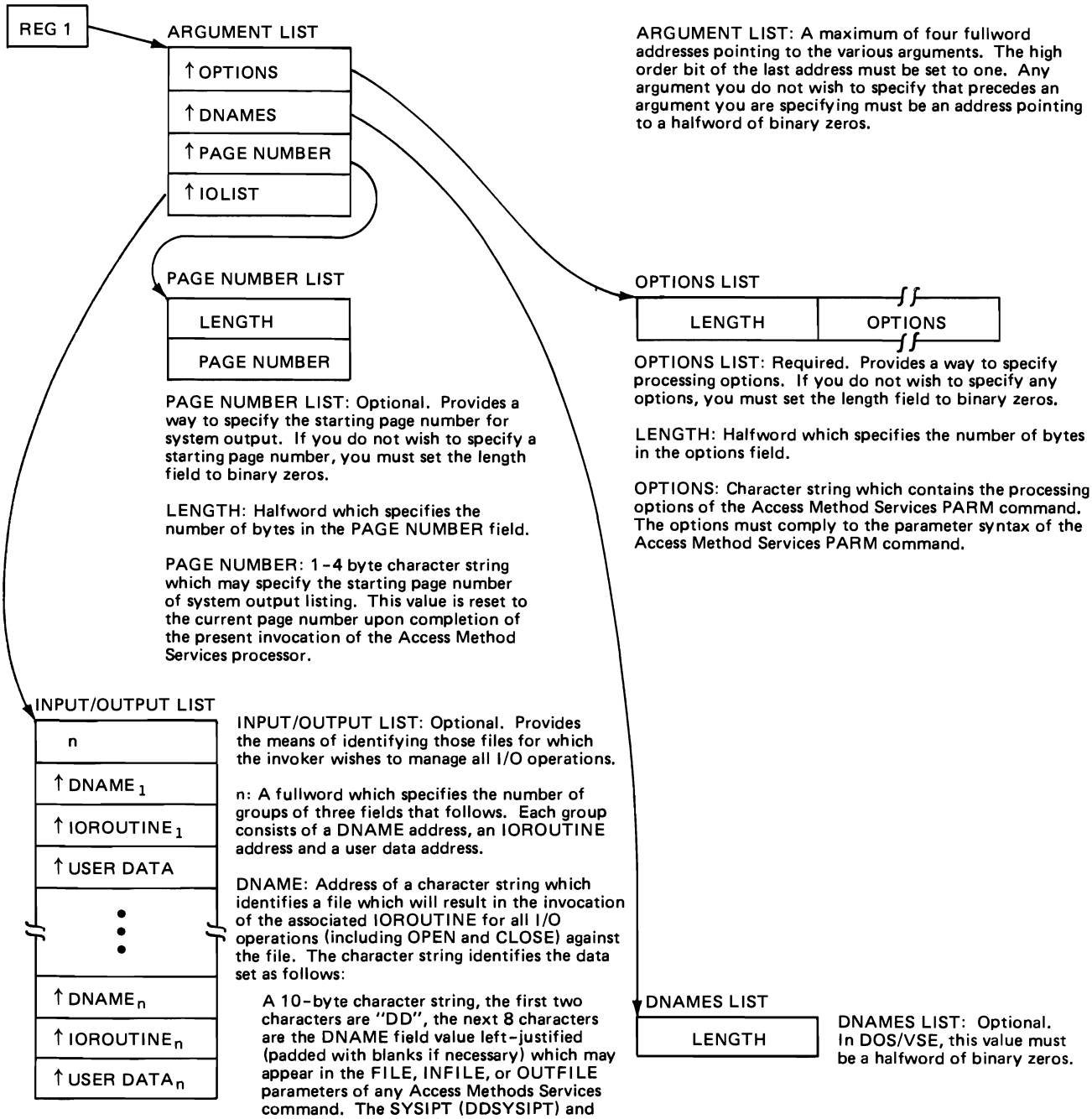


Figure D-1. Processor Invocation Argument List from a Problem Program

User I/O Routines

User I/O Routines enable a user to perform all I/O operations for a file 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 SYSIPT.

A user I/O Routine is invoked by Access Method Services for all operations against the selected files. The identification of the files and their associated I/O routines is via the Input/Output list of the processor invocation parameter list (see Figure D-1).

When writing a user I/O routine, the user must be aware of three things. First, the processor handles the user file as if it were a non-VSAM file that contains undefined records (maximum record length is 32760 bytes) with a physical sequential organization. The processor does not test for the existance of the file. 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 files it is managing and provide to the processor a return code 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 GET/PUT operation, but continue processing
- 12 – do not allow any further calls (except CLOSE) to this routine

Figure D-2 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 filename of the DLBL or TLBL statements for the file; for GET and PUT operations, the IOINFO field is used to communicate the record length and address.

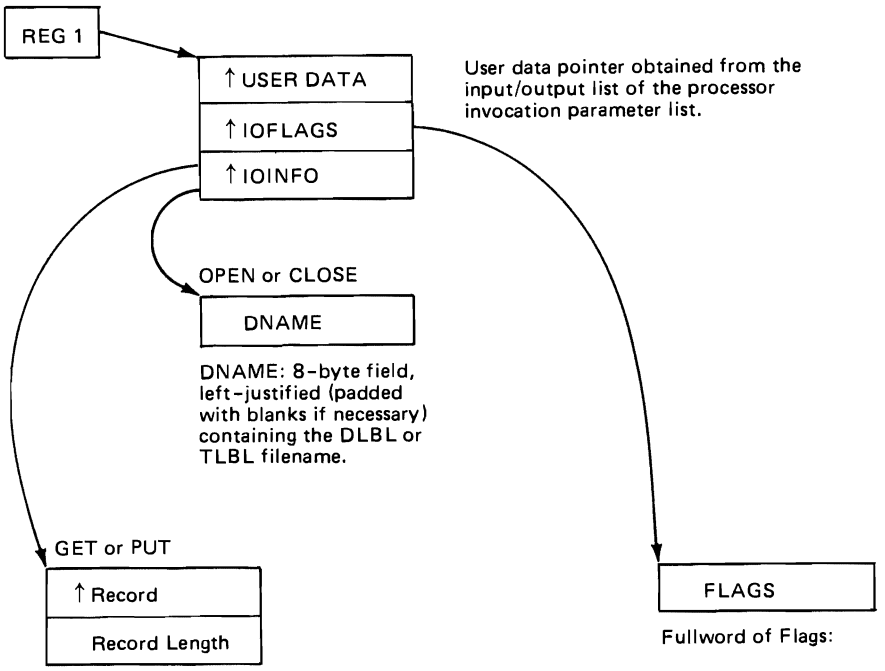


Figure D-2. Arguments Passed to and from a User I/O Routine

Appendix E: 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 parameters can be used with the ALTER command.

Parameter	Abbr	Example	Notes
ALTER	—	ALTER ...	
Required parameter			
entryname	—	OLDDATA/OLDPASSM	6
Optional parameters			
Name parameters			
CATALOG	CAT	CAT(SUPERCAT/SCATM)	
FILE	—	FILE(DD10)	1
NEWNAME	NEWNM	NEWNM(NEWDATA)	
Security and integrity parameters			
ATTEMPTS	ATT	ATT(3)	
AUTHORIZATION	AUTH	AUTH(SECURE CODE10)	
CODE	—	CODE(SWIM)	2
CONTROLPW	CTLPW	CTLPW(FINS)	2
ERASE	ERAS	ERAS	
EXCEPTIONEXIT	EEXT	EEXT(OOPSIO)	2
FOR	—	FOR(756)	
INHIBIT	INH	INH	
MASTERPW	MRPW	MRPW(SCUBA)	2
MODULE	MDLE	NULL(AUTH(MDLE))	4
NOERASE	NERAS	NERAS	
NOWRITECHECK	NWCK	NWCK	
NULLIFY	NULL	NULL(RETN,CODE)	5
OWNER	—	OWNER(BIGCHIEF)	2
READPW	RDPW	RDPW(MASK)	2
RETENTION	RETN	NULL(RETN)	3
SHAREOPTIONS	SHR	SHR(3)	
STRING	STRG	NULL(AUTH(STRG))	4
TO	—	TO(75365)	
UNINHIBIT	UNINH	UNINH	
UPDATEPW	UPDPW	UPDPW(TRUNKS)	2
WRITECHECK	WCK	WCK	

Parameter	Abbr	Example	Notes
Allocation parameters			
ADDVOLUMES	AVOL	AVOL(PCVOL5)	
BUFFERSPACE	BUFSP, BUFSPC	BUFSP(4096)	
FREESPACE	FSPC	FSPC(20 25)	
KEYS	—	KEYS(10 4)	
RECORDSIZE	RECSZ	RECSZ(121 121)	
REMOVEVOLUMES	RVOL	RVOL(PCVOL4)	
Alternate Index and Path parameters			
NONUNIQUEKEY	NUNQK	NUNQK	
NOUPDATE	NUPD	NUPD	
NOUPGRADE	NUPG	NUPG	
UNIQUEKEY	UNQK	UNQK	
UPDATE	UPD	UPD	
UPGRADE	UPG	UPG	

Notes:

1 Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.

2. This parameter can also be a subparameter of NULLIFY

3. RETENTION is a subparameter of NULLIFY.

4. MODULE and STRING are subparameters of NULLIFY (AUTHORIZATION).

5. See notes 2, 3, and 4.

6. This a positional parameter and must follow ALTER.

BLDINDEX Parameters: Summary

The following table shows which parameters can be used with the BLDINDEX command.

Parameter	Abbr	Example	Notes
BLDINDEX	BIX	BIX ...	
Required parameters			
INDATASET	IDS	IDS(PAYROLL.NAMES.PATH)	
INFILE	IFILE	IFILE(DDBASE)	1
OUTDATASET	ODS	ODS(PAYROLL.MASTER)	
OUTFILE	OFILE	OFILE(DDPATH)	1
Optional parameters			
CATALOG	CAT	CAT(UCAT1/MASTPSWD)	
EXTERNALSORT	ESORT	ESORT	2
INTERNALSORT	ISORT	ISORT	2
WORKFILES	WFILE	WFILE(DDWF1 DDWF2)	1
WORKVOLUMES	WVOL	WVOL(VSER01 VSER02)	

Note:

1 Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.

2. When EXTERNALSORT is not specified, INTERNALSORT is the default.

CANCEL Parameters: Summary

The following table shows which parameters can be used with the CANCEL command.

Parameter	Abbr	Example	Notes
CANCEL	—	CANCEL...	
Optional parameters			
JOB	—	JOB	1
STEP	—	STEP	1

Note:

1 When JOB is not specified, STEP is the default.

DEFINE ALTERNATEINDEX Parameters: Summary

The following table shows which parameters 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	—	DEF AIX(...)	
Required parameters				
ALTERNATEINDEX	AIX	—	AIX(...)	
NAME	—	A,D,I	NAME(PAYROLL.NAMES)	19
Allocation parameters				
BLOCKS	BLK BLOCK	A,D,I	BLK(3520 352)	1
CYLINDERS	CYL	A,D,I	CYL(10 2)	1
DEFAULTVOLUMES	DFVOL	A,D,I	DFVOL	22
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)	22
Optional parameters				
CATALOG	CAT	A	CAT(MGMTCAT/CHIEFPW)	
DATA	—	A	DATA(...)	
INDEX	IX	A	IX(...)	
RELATE	REL	A	REL(PAYROLL.MASTER)	4
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)	3
FILE	—	A,D,I	FILE(DDPAY)	2
NOALLOCATION	NAL	A,D,I	NAL	9
NONUNIQUEKEY	NUNQK	A,D	NUNQK	6
NOREUSE	NRUS	A,D,I	NRUS	7
ORDERED	ORD	A,D,I	ORD	8
REUSE	RUS	A,D,I	RUS	7
SUBALLOCATION	SUBAL	A,D,I	SUBAL	9
UNIQUE	UNQ	A,D,I	UNQ	9
UNIQUEKEY	UNQK	A,D	UNQK	6
UNORDERED	UNORD	A,D,I	UNORD	8
USECLASS	USCL	A,D,I	USCL(7 P)	20
• For Data Record Identification and Location				
KEYRANGES	KRNG	A,D	KRNG((A M)(N Z))	
KEYS	—	A,D	KEYS(8 8)	3,17
• For Sequence-Set Record Location				
IMBED	IMBD	A,I	IMBD	14
NOIMBED	NIMBD	A,I	NIMBD	14
NOREPLICATE	NREPL	A,I	NREPL	15
REPLICATE	REPL	A,I	REPL	15
• For Record Processing				
BUFFERSPACE	BUFSP, BUFSPC	A,D	BUFSP(8704)	3
FREESPACE	FSPC	A,D	FSPC(10 10)	21
RECORDSIZE	RECSZ	A,D	RECSZ(121 121)	3,18

Parameter	Abbr	Usage	Example	Notes
Data integrity parameters:				
• When the Alternate Index's Base Cluster Is Opened				
NOUPGRADE	NUPG	A	NUPG	16
UPGRADE	UPG	A	UPG	16
• When Records Are Written Into the Alternate Index				
NOWRITECHECK	NWCK	A,D,I	NWCK	11
RECOVERY	RCVY	A,D	RCVY	12
SPEED	—	A,D	SPEED	12
WRITECHECK	WCK	A,D,I	WCK	11
• When the Alternate Index Is Deleted				
ERASE	ERAS	A,D	ERAS	10
NOERASE	NERAS	A,D	NERAS	10
• When the Alternate Index Is Shared Between Users				
SHAREOPTIONS	SHR	A,D,I	SHR(1)	5
• When an I/O Error Occurs				
EXCEPTIONEXIT	EEXT	A,D,I	EEXT(PAYERROR)	
Protection parameters				
ATTEMPTS	ATT	A,D,I	ATT(4)	13
AUTHORIZATION	AUTH	A,D,I	AUTH(UDE)	
CODE	—	A,D,I	CODE(LES)	
CONTROLPW	CTLPW	A,D,I	CTLPW(UL)	
MASTERPW	MRPW	A,D,I	MRPW(JO)	
READPW	RDPW	A,D,I	RDPW(GO)	
UPDATEPW	UPDPW	A,D,I	UPDPW(GE)	
Ownership/Retention parameters				
FOR	—	A	FOR(380)	
OWNER	—	A,D,I	OWNER(AP)	
TO	—	A	TO(75340)	

Notes:

1. You must specify either CYLINDERS, RECORDS, BLOCKS, or TRACKS as a parameter of ALTERNATEINDEX unless MODEL is specified.
2. You must specify FILE if UNIQUE is specified.
3. 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 and one 512-byte index control interval.
4. RELATE is required except for default models.

Notes 5 through 22 identify the default values when the parameter is not specified as a parameter of ALTERNATEINDEX:

5. SHAREOPTIONS (1 3), when SHR (value reserved) is not specified.
6. NONUNIQUEKEY, when UNIQUEKEY is not specified.
7. NOREUSE, when REUSE is not specified.
8. UNORDERED, when ORDERED is not specified.
9. SUBALLOCATION, when UNIQUE or NOALLOCATION are not specified.
10. NOERASE, when ERASE is not specified.
11. NOWRITECHECK, when WRITECHECK is not specified.
12. RECOVERY, when SPEED is not specified.
13. ATTEMPTS(2), when ATTEMPTS("number") is not specified.
14. NOIMBED, when IMBED is not specified.
15. NOREPLICATE, when REPLICATE is not specified.
16. UPGRADE, when NOUPGRADE is not specified.
17. KEYS(64 0), when KEYS (length offset) is not specified.
18. RECORDSIZE(4086 32600), when RECORDSIZE(average maximum) is not specified.
19. Required at alternate index level; optional at data and index component levels.
20. USECLASS (0 P), when USCL (primary secondary) is not specified.
21. FREESPACE (0 0), when FSPC (cipercent capercent) is not specified.
22. DEFAULTVOLUMES, when VOLUMES is not specified.

DEFINE CLUSTER Parameters: Summary

The following table shows which parameters 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	—	DEF CL	
Required parameters				
CLUSTER	CL	—	CL	
NAME	—	C,D,I	NAME(PAYROLL.MASTER)	22
Allocation parameters				
BLOCKS	BLK BLOCK	C,D,I	BLK(500 100)	1
CYLINDERS	CYL	C,D,I	CYL(10 2)	1
DEFAULTVOLUMES	DFVOL	A,D,I	DFVOL	20
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	14
NONINDEXED	NIXD	C	NIXD	14
NUMBERED	NUMD	C	NUMD	14
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)	21
FILE	—	C,D,I	FILE(DDPAY)	3
NOALLOCATION	NAL	A,D,I	NAL	7
NOREUSE	NRUS	C,D,I	NRUS	5
REUSE	RUS	C,D,I	RUS	5
SUBALLOCATION	SUBAL	C,D,I	SUBAL	7
UNIQUE	UNQ	C,D,I	UNQ	7
USECLASS	USCL	C,D.I	USCL(1 P)	17
• For the Key-Sequenced Cluster As a Whole				
FREESPACE	FSPC	C,D	FSPC(10 10)	18
ORDERED	ORD	C,D,I	ORD	6
UNORDERED	UNORD	C,D,I	UNORD	6
• For Record Processing				
BUFFERSPACE	BUFSP, BUFSPC	C,D	BUFSP(8704)	21
NONSPANNED	NSPND	C,D,	NSPND	4
RECORDSIZE	RECSZ	C,D	RECSZ(121 121)	16,21
SPANNED	SPND	C,D	SPND	4
Data integrity parameters				
• When the Records Are Written Into the Cluster				
NOWRITECHECK	NWCK	C,D,I	NWCK	9
RECOVERY	RCVY	C,D	RCVY	10
SPEED	—	C,D	SPEED	10
WRITECHECK	WCK	C,D,I	WCK	9

Parameter	Abbr	Usage	Example	Notes
• When the Cluster Is Shared Between Users				
SHAREOPTIONS	SHR	C,D,I	SHR(1)	19
• When the Cluster Is Deleted				
ERASE	ERAS	C,D	ERAS	8
NOERASE	NERAS	C,D	NERAS	8
• When an I/O Error Occurs				
EXCEPTIONEXIT	EEXT	C,D,I	EEXT(PAYERROR)	
Protection parameters				
ATTEMPTS	ATT	C,D,I	ATT(4)	11
AUTHORIZATION	AUTH	C,D,I	AUTH(SAR)	
CODE	—	C,D,I	CODE(PER)	
CONTROLPW	CTLPW	C,D,I	CTLPW(COM)	
MASTERPW	MRPW	C,D,I	MRPW(BRI)	
READPW	RDPW	C,D,I	RDPW(AD)	
UPDATEPW	UPDPW	C,D,I	UPDPW(PLAT)	
Ownership/Retention parameters				
FOR	—	C	FOR(380)	
OWNER	—	C,D,I	OWNER(BR)	
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)	15
• For Sequence-Set Record Location				
IMBED	IMBD	C,I	IMBD	12
NOIMBED	NIMBD	C,I	NIMBD	12
NOREPLICATE	NREPL	C,I	NREPL	13
REPLICATE	REPL	C,I	REPL	13

Notes:

1. You must specify either CYLINDERS, RECORDS, BLOCKS, or TRACKS as a parameter of CLUSTER unless MODEL is specified.
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.
3. You must specify FILE if UNIQUE is specified.

Notes 4 through 20 identify the default values when the parameter is not specified as a parameter of CLUSTER:

4. NONSPANNED, when SPANNED is not specified.
5. NOREUSE, when REUSE is not specified.
6. UNORDERED, when ORDERED is not specified.
7. SUBALLOCATION, when UNIQUE or NOALLOCATION are not specified.
8. NOERASE, when ERASE is not specified.
9. NOWRITECHECK, when WRITECHECK is not specified.
10. RECOVERY, when SPEED is not specified.
11. ATTEMPTS(2), when ATTEMPTS (number) is not specified.
12. NOIMBED, when IMBED is not specified.
13. NOREPLICATE, when REPLICATE is not specified.
14. INDEXED, when neither NONINDEXED nor NUMBERED are specified.
15. KEYS(64 0), when KEYS (length offset) is not specified for a key-sequenced file.
16. RECORDSIZE (4086 32600), when RECORDSIZE is not specified and SPANNED is specified; otherwise, RECORDSIZE(4089 4089) when RECORDSIZE is not specified and SPANNED is not specified.
17. USECLASS (0 P) when USCL (primary secondary) is not specified.
18. FREESPACE (0 0), when FSPC (cipercent capercent) is not specified.
19. SHAREOPTIONS (1 3), when SHR (value reserved) is not specified.
20. DEFAULTVOLUMES, when VOLUMES is not specified.
21. 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.
22. Required at the cluster level; optional at the data and index component levels.

DEFINE MASTERCATALOG or USERCATALOG Parameters: Summary

The following table shows which parameters can be used with the DEFINE MASTERCATALOG and 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		DEF MCAT(...)	
Required parameters				
MASTERCATALOG	MCAT, MRCAT	—	MCAT(...)	
USERCATALOG	UCAT	—	UCAT(...)	
NAME	—	U	NAME(USERCAT.TWO5)	
Allocation parameters				
BLOCKS	BLK BLOCK	U,D,I	BLK(500 200)	3
CYLINDERS	CYL	U,D,I	CYL(10 2)	3
DEDICATE	DED	U	DED	3,10
FILE	—	U	FILE(NEWCATDD)	1
RECORDS	REC	U,D,I	REC(800 150)	3
TRACKS	TRK	U,D,I	TRK(100 20)	3
VOLUME	VOL	U	VOL(UVOL25)	
Optional parameters				
CATALOG	CAT	—	CAT(MAS/MWPW)	7
DATA	—	—	DATA(...)	
INDEX	IX	—	IX(...)	
Model Parameter				
MODEL	—	U	MODEL(USCAT)	7
Sequence Set Record Location parameters				
IMBED	IMBD	U,I	IMBD	9
NOIMBED	NIMBD	U,I	NIMBD	9
Allocation parameters				
BUFFERSPACE	BUFSP	U,D	BUFSP(8192)	2
CLASS	—	U	CLASS(1)	8
NOTRECOVERABLE	NRVBL	U,D	NRVBL	5
ORIGIN	ORG	U	ORG(20)	10,11
RECOVERABLE	RVBL	U,D	RVBL	5
• Protection parameters				
ATTEMPTS	ATT	U	ATT(1)	6
AUTHORIZATION	AUTH	U	AUTH(SPR)	
CODE	—	U	CODE(CAN)	
CONTROLPW	CTLPW	U	CTLPW(TOF)	
MASTERPW	MRPW	U	MRPW(CHO)	
READPW	RDPW	U	RDPW(COC)	
UPDATEPW	UPDPW	U	UPDPW(CHU)	
• Data integrity parameters				
NOWRITECHECK	NWCK	U,D,I	NWCK	4
WRITECHECK	WCK	U,D,I	WCK	4
• Ownership/Retention parameters				
FOR	—	U	FOR(9999)	
OWNER	—	U	OWNER(STI)	
TO	—	U	TO(99360)	

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. When no value is specified, the default is BUFFERSPACE(3072).
3. You must specify either CYLINDERS, DEDICATE, RECORDS, BLOCKS, or TRACKS as a subparameter of MASTERCATALOG and USERCATALOG.
4. When WRITECHECK is not specified, the default is NOWRITECHECK.
5. When RECOVERABLE is not specified, the default is NOTRECOVERABLE.
6. When no value is specified, the default is ATTEMPTS(2).
7. Can only be specified with DEFINE USERCATALOG.
8. When no value is specified, the default is CLASS (0).
9. When NOIMBED is not specified, the default is IMBED.
10. Do not specify ORIGIN with DEDICATE.
11. When neither ORIGIN nor DEDICATE is specified, the first available extent on the volume large enough to hold the catalog level space-allocation-value is chosen.

DEFINE NONVSAM Parameters: Summary

The following table shows which parameters can be used with the DEFINE NONVSAM command.

Parameter	Abbr	Example
DEFINE	DEF	DEF NVSAM
Required parameters		
NONVSAM	NVSAM	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)

DEFINE PATH Parameters: Summary

The following table shows which parameters can be used with the DEFINE PATH command.

Parameter	Abbr	Example	Notes
DEFINE	DEF	DEF PATH(...)	
Required parameters			
PATH	—	PATH(...)	
NAME	—	NAME(PAYROLL.NAMES.PATH)	
PATHENTRY	PENT	PENT(PAYROLL.NAMES)	
Optional parameters			
CATALOG	CAT	CAT(USER1CAT)	
Allocation parameter			
FILE	—	FILE(STEPCAT)	1
Model parameter			
MODEL	—	MODEL(BEA)	
Update parameters			
NOUPDATE	NUPD	NUPD	2
UPDATE	UPD	UPD	2
Protection parameters			
ATTEMPTS	ATT	ATT(3)	3
AUTHORIZATION	AUTH	AUTH(PATHCHK)	
CODE	—	CODE(FOL)	
CONTOLPW	CTLPW	CTLPW(UL)	
MASTERPW	MRPW	MRPW(PET)	
READPW	RDPW	RDPW(BAN)	
UPDATEPW	UPDPW	UPDPW(MAR)	
Owner/Retention parameters			
FOR	—	FOR(365)	
OWNER	—	OWNER(ME)	
TO	—	TO(75001)	

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. When NOUPDATE is not specified, UPDATE is the default.
3. When ATTEMPTS is not specified, ATTEMPTS(2) is the default.

DEFINE SPACE Parameters: Summary

The following table shows which parameters can be used with the DEFINE SPACE command.

Parameter	Abbr	Example	Notes
DEFINE	DEF	DEF SPC(...)	
Required parameters			
SPACE	SPC	SPC (...)	
BLOCKS	BLK BLOCK	BLK(500 100)	2
CANDIDATE	CAN	CAN	2
CYLINDERS	CYL	CYL(10 2)	2
DEDICATE	DED	DED	2,5
FILE	—	FILE(DDSPC)	1
RECORDS	REC	REC(1000 200)	2,3
RECORDSIZE	RECSZ	RECSZ(80 80)	3
TRACKS	TRK	TRK(200 40)	2
VOLUMES	VOL	VOL(PSV525)	
Optional parameters			
CATALOG	CAT	CAT(USERCAT1)	
CLASS	CLASS	CLASS(1)	4
ORIGIN	ORG	ORG(40)	5

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. You must specify either CANDIDATE, CYLINDERS, DEDICATE, RECORDS, BLOCKS,or TRACKS as a parameter of SPACE.
3. If RECORDS is specified, you must also specify RECORDSIZE, but, you cannot specify RECORDSIZE without specifying records.
4. If no value is specified, CLASS(0) is the default.
5. Do not specify ORIGIN with DEDICATE.

DELETE Parameters: Summary

The following table shows which parameters can be used with the DELETE command.

Parameter	Abbr	Example	Notes
DELETE	DEL	DEL ...	
Required parameter			
entryname	—	(NEWDATA/BOB)	6
Optional parameters			
ALTERNATEINDEX	AIX	AIX	
CATALOG	CAT	CAT(UCAT57/MASTPW)	
CLUSTER	CL	CL	
ERASE	ERAS	ERAS	
FILE	—	FILE(DDCLU)	1
FORCE	FRC	FORCE	5
MASTERCATALOG	MCAT	MCAT	2
NOERASE	NERAS	NERAS	
NOFORCE	NFRC	NFRC	5
NONVSAM	NVSAM	NVSAM	
NOPURGE	NPRG	NPRG	3
NOSCRATCH	NSCR	NSCR	4
PATH	—	PATH	
PURGE	PRG	PRG	3
SCRATCH	SCR	SCR	4
SPACE	SPC	SPC	2
USERCATALOG	UCAT	UCAT	2

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. 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, MASTERCATALOG, or USERCATALOG.
3. Unless you specify PURGE, the default is NOPURGE.
4. Unless you specify NOSCRATCH, the default is SCRATCH.
5. Unless you specify FORCE, the default is NOFORCE.
6. This is a positional parameter and must follow DELETE.

EXPORT Parameters: Summary

The following table shows which parameters can be used with the EXPORT command.

Parameter	Abbr	Example	Notes
EXPORT	EXP	EXP ...	
Required parameters			
entryname	—	USERCAT1/MCATMPW	11
DISCONNECT	DCON	DCON	2
INFILE	IFILE	IFILE(DD2 ENV(...))	1
OUTFILE	OFILE	OFILE(DD2 ENV(...))	2,12,13,14
Optional parameters			
BLOCKSIZE	BLKSZ	ENV(BLKSZ(6000)...)	12,13
CIMODE	CIM	CIM	17
ENVIRONMENT	ENV	OFILE(DDEXP2 ENV(...))	13
ERASE	ERAS	ERAS	10
INHIBITSOURCE	INHS	INHS	4,5
INHIBITTARGET	INHT	INHT	6
NOERASE	NERAS	NERAS	
NOINHIBITSOURCE	NINHS	NINHS	4
NOINHIBITTARGET	NINHT	NINHT	6
NOLABEL	NLBL	ENV(NLBL)	15
NOPURGE	NPRG	NPRG	7
NOREWIND	NREW	ENV(NREW)	16
PERMANENT	PERM	PERM	9
PRIMEDATADEVICE	PDEV	ENV(...PDEV(2400))	13,14
PURGE	PRG	PRG	7,8
RECORDMODE	RECM	RECM	17
REWIND	REW	ENV(REW)	16
STDLABEL	SLBL	ENV(SLBL)	15
TEMPORARY	TEMP	TEMP	5,8,9,10
UNLOAD	UNLD	ENV(UNLD)	16

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. When ***entryname*** names a user catalog, DISCONNECT is required. Otherwise, DISCONNECT cannot be used.
3. When ***entryname*** names a cluster or alternate index, OUTFILE is required.
4. When INHIBITSOURCE is not specified, NOINHIBITSOURCE is the default.
5. When INHIBITSOURCE is specified, TEMPORARY must also be specified.
6. When INHIBITTARGET is not specified, NOINHIBITTARGET is the default.
7. When PURGE is not specified, NOPURGE is the default.
8. You cannot specify PURGE and TEMPORARY together.
9. When TEMPORARY is not specified, PERMANENT is the default.
10. You cannot specify ERASE and TEMPORARY together.
11. This positional parameter must immediately follow EXPORT
12. When not specified, the default for blocksize is 2048 bytes per block.
13. When you specify OUTFILE, ENVIRONMENT describes BLOCKSIZE and device type - PRIMEDATADEVICE.
14. When not specified, the default for device type is 2314.
15. When NOLABEL is not specified, STDLABEL is the default.
16. When REWIND or UNLOAD is not specified, NOREWIND is the default.
17. When RECORDMODE is not specified, CIMODE is the default.

EXPORTRA Parameters: Summary

The following table shows which parameters can be used with the EXPORTRA command.

Parameter	Abbr	Example	Notes
EXPORTRA	XPRA	XPRA ...	
Required parameters			
CRAVOLUMES	CRAVOL	CRAVOL((...)(...))	
ALL	—	CRAVOL((VSER01 ALL))	2,12
ENTRIES	ENT	CRAVOL((VSER01 ENT(MYDS)))	2,12
INFILE	IFILE	CRA((CRAVOL1 ALL IFILE(CRAVOL2)))	1
NONE	—	CRAVOL((VSER01 NONE))	2,12
OUTFILE	OFILE	OFILE(CRAVOL3 ENV (...))	3,6,8
Optional parameters			
BLOCKSIZE	BLKSZ	ENV(BLKSZ(6000)...)	6
CIMODE	CIM	CIM	11
ENVIRONMENT	ENV	OFILE(CRAVOL3 ENV(...))	7
FORCE	FRC	FRC	4
MASTERPW	MRPW	MRPW(MCATMPW)	5
NOFORCE	NFRC	NFRC	4
NOLABEL	NLBL	ENV(NLBL)	9
NOREWIND	NREW	ENV(NREW)	10
PRIMEDATADEVICE	PDEV	ENV(...PDEV(2400))	8
RECORDMODE	RECM	RECM	11
REWIND	REW	ENV(REW)	10
STDLABEL	SLBL	ENV(SLBL)	9
UNLOAD	UNLD	ENV(UNLD)	10

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. VSER01 identifies the catalog recovery area's volume.
3. CRAVOL3 identifies a DLBL statement that describes a portable copy (that is, a disk pack or magnetic tape reel of an output file.)
4. Unless you specify FORCE, the default option is NOFORCE.
5. When the master catalog is password protected, you must supply its master password.
6. When not specified, the default is 2048 bytes per block.
7. When you specify OUTFILE, ENVIRONMENT describes BLOCKSIZE and device type (PRIMEDATADEVICE).
8. When not specified, the default is 2314.
9. When NOLABEL is not specified, STDLABEL is the default.
10. When REWIND or UNLOAD is not specified, NOREWIND is the default.
11. When RECORDMODE is not specified, CIMODE is the default.
12. When NONE or ENTRIES is not specified, ALL is the default.

IMPORT Parameters: Summary

The following table shows which parameters can be used with the IMPORT command.

Parameter	Abbr	Example	Notes
IMPORT	IMP	IMP ...	
Required parameters			
CONNECT	CON	CON	2
DEVICETYPE	DEVT	DEVT(3340)	2,3
INFILE	IFILE	IFILE(DDBU ENV(...))	7,12
OBJECTS	OBJ	OBJ((...))	11
OUTFILE	OFILE	OFILE(DDOLD)	1
OUTPW	OPW	OPW(OPASSWD)	
Optional parameters			
BLOCKSIZE	BLKSZ	ENV(BLKSZ(600...)	6,7
CATALOG	CAT	CAT(USERCAT1/UPDPW1)	
DEFAULTVOLUMES	DFVOL	DFVOL	
ENVIRONMENT	ENV	(IFILE(DDEXP2 ENV(...))	7
ERASE	ERAS	ERAS	5
FILE	—	FILE(DDKSVOL)	3
KEYRANGES	KRNG	KRNG((A M) (N Z))	3
NEWNAME	NEWNM	NEWNM(NEW.DS)	3
NOERASE	NERAS	NERAS	5
NOLABEL	NLBL	ENV(NLBL)	9
NOREWIND	NREW	ENV(NREW)	10
NOPURGE	NPRG	NPRG	5,13
ORDERED	ORD	ORD	3,4
PRIMEDATADEVICE	PDEV	ENV(...PDEV(2400))	7,8
PURGE	PRG	PRG	5,13
REWIND	REW	ENV(REW)	10
STDLABEL	SLBL	ENV(SLBL)	9
UNLOAD	UNLD	ENV(UNLD)	10
UNORDERED	UNORD	UNORD	3,4
USECLASS	USCL	USCL(7 P)	3
VOLUMES	VOL	VOL(KSVOL1)	3

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. When a user catalog is imported, CONNECT and DEVICETYPE are required. Otherwise, CONNECT and DEVICETYPE cannot be used.
3. This parameter is a subparameter of OBJECTS.
4. When ORDERED is not specified, UNORDERED is the default.
5. ERASE or NOERASE, and PURGE or NOPURGE can be specified only for temporarily-exported alternate indexes and clusters.
6. When not specified, the default for blocksize is 2048 bytes per block.
7. When you specify INFILE, ENVIRONMENT describes BLOCKSIZE and PRIMEDATADEVICE.
8. When not specified, the default device type is 2314.
9. When NOLABEL is not specified, STDLABEL is the default.
10. When REWIND or UNLOAD is not specified, NOREWIND is the default.
11. When a user catalog is imported, OBJECTS is required; otherwise, it is optional.
12. When an alternate index or cluster is imported, INFILE is required; otherwise, it cannot be used.
13. When PURGE is not specified, NOPURGE is the default.

IMPORTRA Parameters: Summary

The following table shows which parameters can be used with the IMPORTRA command.

Parameter	Abbr	Example	Notes
IMPORTRA	MPRA	MPRA ...	
Required parameters			
INFILE	IFILE	IFILE(PORT1 ENV(...))	3
OUTFILE	OFILE	OFILE(PORT2)	1
Optional parameters			
BLOCKSIZE	BLKSZ	ENV(BLKSZ(6000)...)	3
CATALOG	CAT	CAT(MYCAT2)	
DEFAULTVOLUMES	DFVOL	DFVOL	
DEVICETYPE	DEVT	DEVT(3330)	2
ENVIRONMENT	ENV	IFILE(EXP2 ENV(...))	
FILE		FILE(DDUNIQ)	2
NOLABEL	NLBL	ENV(NLBL)	5
NOREWIND	NREW	ENV(NREW)	6
OBJECTS	OBJ	OBJ((MYDS VOL(VOL001) DEVT(3330)))	2
PRIMEDATADEVICE	PDEV	ENV(...PDEV(2400))	4
REWIND	REW	ENV(REW)	6
STDLABEL	SLBL	ENV(SLBL)	5
UNLOAD	UNLD	ENV(UNLD)	6
USECLASS	USCL	USCL(1 1)	2
VOLUMES	VOL	VOL(VOL001)	2

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. This parameter is a subparameter of OBJECTS.
3. When not specified, the default for blocksize is 2048 bytes per block.
4. When not specified, the default device type is 2314.
5. When NOLABEL is not specified, STDLABEL is the default.
6. When REWIND or UNLOAD is not specified, NOREWIND is the default.

LISTCAT Parameters: Summary

The following table shows which parameters 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			
ALL	—	ALL	1
ALLOCATION	ALLOC	ALLOC	1
ALTERNATEINDEX	AIX	AIX	
CATALOG	CAT	CAT(USERCAT1)	
CLUSTER	CL	CL	
DATA	—	DATA	
ENTRIES	ENT	ENT(DS1 DS2)	
INDEX	IX	IX	
NAME	—	NAME	1
NONVSAM	NVSAM	NVSAM	
NOTUSABLE	NUS	NUS	
PATH	—	PATH	
SPACE	SPC	SPC	
USERCATALOG	UCAT	UCAT	
VOLUME	VOL	VOL	1

Notes:

1. NAME is the default, unless ALL, VOLUME, or ALLOCATION is specified.

LISTCRA Parameters: Summary

The following table shows which parameters can be used with the LISTCRA command.

Parameter	Abbr	Example	Notes
LISTCRA	LISTR	LISTR ...	
Required parameters			
INFILE	IFILE	IFILE(DDVOL1)	1
INVOLUMES	IVOL	IVOL(VSER01)	
Optional parameters			
CATALOG	CAT	CAT(UCAT01)	4
COMPARE	CMPR	CMPR	2,4,6
DUMP	—	DUMP	3
MASTERPW	MRPW	MRPW(MCATMRPW)	5
NAME	—	NAME	3
NOCOMPARE	NCMPR	NCMPR	2
SEQUENTIALDUMP	SDUMP	SDUMP	3,6

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. Unless you specify COMPARE, the default is NOCOMPARE.
3. Unless you specify DUMP or SEQUENTIALDUMP, the default is NAME.
4. When you specify COMPARE, you must identify the catalog with the CATALOG parameter.
5. When the master catalog is password protected, you must supply its master password.
6. SEQUENTIALDUMP cannot be specified with COMPARE.

PRINT Parameters: Summary

The following table shows which parameters can be used with the PRINT command.

Parameter	Abbr	Example	Notes
PRINT	—	PRINT INFILE(...)	
Required parameter			
INFILE	IFILE	IFILE(DDMDS)	
Optional parameters			
BLOCKSIZE	BLKSZ	ENV(BLKSZ(800)...)	4
CHARACTER	CHAR	CHAR	1
COUNT	—	COUNT(23)	3
DUMP	—	DUMP	1
ENVIRONMENT	ENV	OFILE(DDEXP1 ENV(...))	5
FROMADDRESS	FADDR	FADDR(122)	2
FROMKEY	FKEY	FKEY(M)	2
FROMNUMBER	FNUM	FNUM(23)	2
HEX	—	HEX	1
NOLABEL	NLBL	ENV(NLBL)	9
NOREWIND	NREW	ENV(NREW)	10
PRIMEDATADEVICE	PDEV	ENV(...PDEV(3330))	6
RECORDFORMAT	RECFM	ENV(REFCM(FB))	7
RECORDSIZE	RECSZ	ENV(RECSZ(100)...))	8
REWIND	REW	ENV(REW)	10
SKIP	—	SKIP(20)	2
STDLABEL	SLBL	ENV(SLBL)	9
TOADDRESS	TADDR	TADDR(442)	3
TOKEY	—	TOKEY(S)	3
TONUMBER	TNUM	TNUM(30)	3
UNLOAD	UNLD	ENV(UNLD)	10

Notes:

1. When CHARACTER or HEX is not specified, DUMP is the default.
2. You can identify the record at which printing is to start with: FROMKEY, FROMADDRESS, FROMNUMBER, or SKIP.
3. You can identify the record at which printing is to end with: TOKEY, TOADDRESS, TONUMBER, or COUNT.
4. This parameter is required for a nonVSAM file. For indexed-sequential or sequential FIXBLK files, substitute the logical record length times the number of records per block. For indexed-sequential FIXUNB files, substitute the logical record length plus the key length.
5. When you are printing a nonVSAM file, this parameter is required. Subparameters BLOCKSIZE, and RECORDFORMAT are required for nonVSAM file processing.
6. When not specified, the default is 2314.
7. This parameter is required when you are printing a nonVSAM file. For format description, substitute one of the following values:

Value	Abbr	Meaning
FIXUNB	F	Fixed, unblocked
FIXBLK	FB	Fixed, blocked
VARUNB	V	Variable, unblocked
VARBLK	VB	Variable, blocked
SPNUNB	S	Variable spanned, unblocked
SPNBLK	SB	Variable spanned, blocked
UNDEF	U	Undefined

8. For indexed-sequential FIXUNB files, this parameter is the logical record length plus the key length. For UNDEF, SPNBLK, or SPUNUB nonVSAM files, this parameter is the maximum record size. Default record size is equal to block size for files that are FIXUNB, FIXBLK or UNDEF. Default record size is equal to block size minus four for files that are VARUNB, VARBLK, SPNUNB, or SPNBLK.
9. When NOLABEL is not specified, STDLABEL is the default.
10. When REWIND or UNLOAD is not specified, NOREWIND is the default.

REPRO Parameters: Summary

The following table shows which parameters can be used with the REPRO command.

Parameter	Abbr	Example	Notes
REPRO	—	REPRO...	
Required parameters			
INFILE	IFILE	IFILE(DDMDS)	
OUTFILE	OFILE	OFILE(DDNDS)	
Optional parameters			
BLOCKSIZE	BLKSZ	ENV(BLKSZ(800)...)	1
COUNT	—	COUNT(25)	4
ENVIRONMENT	ENV	OFILE(DDEXP2(ENV(...)) IFILE(DDEXP1 ENV(...))	2
FROMADDRESS	FADDR	FADDR(122)	3
FROMKEY	FKEY	FKEY(M)	3
FROMNUMBER	FNUM	FNUM(25)	3
HINDEXDEVICE	HDEV	ENV(HDEV(3330)...)	9
NOLABEL	NLBL	ENV(NLBL)	10
NOREPLACE	NREP	NREP	5
NOREWIND	NREW	ENV(NREW)	11
NOREUSE	NRUS	NRUS	
PRIMEDATADEVICE	PDEV	ENV(...PDEV(2319))	6
RECORDSIZE	RECSZ	ENV(RECSZ(80)...)	8
RECORDFORMAT	RECFM	ENV(RECFM(FB)...)	7
REPLACE	REP	REP	5
REUSE	RUS	RUS	
REWIND	REW	ENV(REW)	11
SKIP	—	SKIP(20)	3
STDLABEL	SLBL	ENV(SLBL)	10
TOADDRESS	TADDR	TADDR(442)	4
TOKEY	—	TOKEY(S)	4
TONUMBER	TNUM	TNUM(30)	4
UNLOAD	UNLD	ENV(UNLD)	11

Notes:

1. This parameter is required for a nonVSAM file. For indexed-sequential or sequential FIXBLK files, substitute the logical record length times the number of records per block. For indexed-sequential FIXUNB files, substitute the logical record length plus the key length.
2. When you are printing a nonVSAM file, this parameter is required. Subparameters BLOCKSIZE, and RECORDFORMAT are required for nonVSAM file processing.
3. You can identify the record at which copying is to start with: FROMKEY, FROMADDRESS, FROMNUMBER, or SKIP.
4. You can identify the record at which copying is to end with: TOKEY, TOADDRESS, TONUMBER, or COUNT.
5. When REPLACE is not specified, NOREPLACE is the default.
6. When not specified, the default is 2314.
7. This parameter is required when you are printing a nonVSAM file. For format description, substitute one of the following values:

Value	Abbr	Meaning
FIXUNB	F	Fixed, unblocked
FIXBLK	FB	Fixed, blocked
VARUNB	V	Variable, unblocked
VARBLK	VB	Variable, blocked
SPNUNB	S	Variable spanned, unblocked
SPNBLK	SB	Variable spanned, blocked
UNDEF	U	Undefined

8. For indexed-sequential FIXUNB files, this parameter is the logical record length plus the key length. For UNDEF, SPNBLK, or SPNUNB nonVSAM files, this parameter is the maximum record size. Default record size is equal to block size for files that are FIXUNB, FIXBLK or UNDEF. Default record size is equal to block size minus four for files that are VARUNB, VARBLK, SPNUNB, or SPNBLK.
9. Specify when used with INFILE, for an indexed-sequential file; the default is the device type of the volume on which the prime data records reside.
10. When NOLABEL is not specified, STDALBEL is the default.
11. When REWIND or UNLOAD is not specified, NOREWIND is the default.

RESETCAT Parameter Summary

The following table shows which parameters can be used with the RESETCAT command.

Parameter	Abbr	Example	Notes
RESETCAT	RCAT	RCAT CAT (...)	2
Required Parameters			
CATALOG	CAT	CAT (USERCAT1/UPDPW1)	3
CRAFILES	CRAFILE	CRAFILE((CRA1 ALL) (DDCRA2 ALL))	1
CRAVOLUMES	CRAVOL	CRAVOL((VSER01 ALL) (VSER02 ALL))	4
WORKCAT	WCAT	WCAT(USERCAT2/ UPDPW2)	5
Optional Parameters			
IGNORE	IGN	IGN	7
MASTERPW	MRPW	MRPW(MSTRPW2)	
NOIGNORE	NIGN	NIGH	7
WORKVOLUMES	WVOL	WVOL(VSER03/PDPW2 VSER04)	6

Notes:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.
2. You may use RESETCAT only with recoverable catalogs.
3. USERCAT1 is the catalog to be reset.
4. VSER01 is the volume containing a CRA to be accessed.
5. USERCAT2 is the catalog where the workfiles are defined.
6. VSER03 and VSER04 identify the workfile's volumes.
7. When IGNORE is not specified, NOIGNORE is the default.

VERIFY Parameters: Summary

The following table shows which parameters can be used with the VERIFY command.

Parameter	Abbr	Example	Notes
VERIFY	VFY	VFY DS(...)	
Required Parameters			
DATASET	DS	DS(PAYROLL.MASTER)	
FILE	—	FILE(ACCNTS)	1

NOTES:

1. Beginning with Release 2, this parameter has been replaced or is no longer required because of the simplified method of specifying VSAM's job control.

Appendix F: Operand Notation for VSAM Macros

The addresses, names, numbers, and options required with operands in GENCB, MODCB, SHOWCB, TESTCB, BLDVRP, WRTBFR, and SHOWCAT can be expressed in a variety of ways:

- An absolute numeric expression, for example, RECLEN=400, as in the following sample job stream:

```
          LA      1,RPL                 Set RPL address in reg 1
          MODCB   RPL=(1),RECLEN=400    Set record length field in
*                                       RPL to value of 400
```

- A character string, for example, DDNAME=DATASET
- A code or a list of codes separated by commas and enclosed in parentheses, for example, OPTCD=KEY or OPTCD=(KEY,DIR,IN)
- A register from 2 through 12 that contains an address or numeric value. Equated labels can be used to designate a register, for example, SYNAD=(ERR), where the following equate statement has been included in the program: ERR EQU 3

  An example of register notation for an operand that takes a numeric value is:

```
          LA      6,400                 Set length desired in reg 6
          MODCB   RPL=RPL,RECLEN=(6)    Set record length field in RPL
*                                       to value specified in reg 6
```

  An example of register notation for an operand that takes an address value is:

```
          LA      2,RCDAREA             Set address of record area in
*                                       reg 2
          MODCB   RPL=RPL,AREA=(2)      Set area operand in RPL according
*                                       to contents of reg 2
          .
          .
          .
RCDAREA   DS  CL400
```

- An expression of the form (S,scon), where scon is any expression valid for an S-type address constant, including the base-displacement form.

  An example of S-type address notation for an operand that takes a numeric value follows (the use of the S-type notation, for numeric-value operands is usually equivalent to either absolute-numeric-expression notation or register notation):

```
          MODCB   RPL=RPL,RECLEN=(S,400)    Set record length field
*                                           in RPL to value of 400
```

An example of S-type address notation for an operand that takes an address value is:

```
          MODCB   RPL=RPL,AREA=(S,RCDAREA1)   Set area operand in RPL
*                                             to address of RCDAREA1
          .
          .
          .
RCDAREA1  DS  CL400
```

- An expression of the form (*,scon), where scon is any expression valid for an S-type address constant, including the base-displacement form; the address specified by scon is indirect, that is, it points to the location that contains the value of the keyword.

  If indirect S-type address notation is used, the value it points to must meet either of the following criteria:

  - If the value is a numeric quantity, a bit string representing codes, or a pointer, it must occupy a fullword of storage.
  - If the value is an alphameric character string, it must occupy two words of storage, be left aligned, and be padded on the right with blanks, if necessary.

  An example of indirect S-type address notation for an operand that takes a numeric value is:

```
          MODCB   RPL=RPL,RECLEN=(*,RECLEN1)  Set record length field
*                                             in RPL to value specified
*                                             in the fullword RECLEN1.
          .
          .
          .
RECLEN1   DC  F'400'
```

An example of indirect S-type address notation for an operand that takes a name value is:

```
          MODCB  ACB=ACB,DDNAME=(*,DDNAME1)  Set ddname field in ACB to
*                                            value specified in the
*                                            8-byte field DDNAME1.
          .
          .
          .
DDNAME1   DC  CL8'FILENAME'
```

An example of indirect S-type notation for an operand that takes an address value is:

```
          MODCB   RPL=RPL,AREA=(*,ARCDAREA)  Set area operand in RPL to
*                                            address of area pointed to
*                                            by the pointer ARCDAREA.
          .
          .
          .
ARCDAREA  DC  A(RCDAREA1)
```

- An expression valid for a relocatable A-type address constant, for example:

```
         MODCB  RPL=RPL,AREA=RCDAREA          Set area operand in RPL
*                                             to address of RCDAREA
```

The expressions that can be used depend on the keyword specified. Register and S-type address notations cannot be used when MF=L is specified.

GENCB Macro Operands

Keyword	Absolute Numeric	Code	Character String	Register	S-Type Address	Indirect S-Type Address	A-Type Address
GENCB Keywords							
AM		X					
BLK		X					
COPIES	X			X	X	X	
LENGTH	X			X	X	X	
WAREA				X	X	X	X
ACB Keywords (BLK=ACB)							
BUFND	X			X	X	X	
BUFNI	X			X	X	X	
BUFSP	X			X	X	X	
DDNAME			X			X	
EXLST				X	X	X	X
MACRF		X					
MAREA				X	X	X	X
MLEN	X			X	X	X	
PARMS=CLOSDSP		X					
PASSWD				X	X	X	X
STRNO	X			X	X	X	
EXLST Keywords (BLK=EXLST)							
EODAD				X	X	X	X
EXCPAD				X	X	X	X
JRNAD				X	X	X	X
LERAD				X	X	X	X
SYNAD				X	X	X	X
RPL Keywords (BLK=RPL)							
ACB				X	X	X	X
AREA				X	X	X	X
AREALEN	X			X	X	X	
ARG				X	X	X	X
KEYLEN	X			X	X	X	
NXTRPL				X	X	X	X
OPTCD		X					
RECLEN	X			X	X	X	
TRANSID	X			X	X	X	

MODCB Macro Operands

Keyword	Absolute Numeric	Code	Character String	Register	S-Type Address	Indirect S-Type Address	A-Type Address
MODCB Keyword							
{ACB|EXLST|RPL}				X	X	X	X
AM		X					
ACB Keywords							
BUFND	X			X	X	X	
BUFNI	X			X	X	X	
BUFSP	X			X	X	X	
DDNAME			X			X	
EXLST				X	X	X	X
MACRF		X					
MAREA				X	X	X	X
MLEN	X			X	X	X	
PARMS=CLOSDSP		X					
PASSWD				X	X	X	X
STRNO	X			X	X	X	
EXLST Keywords							
EODAD				X	X	X	X
EXCPAD				X	X	X	X
JRNAD				X	X	X	X
LERAD				X	X	X	X
SYNAD				X	X	X	X
RPL Keywords							
ACB				X	X	X	X
AREA				X	X	X	X
AREALEN	X			X	X	X	
ARG				X	X	X	X
KEYLEN	X			X	X	X	
NXTRPL				X	X	X	X
OPTCD		X					
RECLEN	X			X	X	X	
TRANSID	X			X	X	X	

SHOWCB Macro Operands

Keyword	Absolute Numeric	Code	Character String	Register	S-Type Address	Indirect S-Type Address	A-Type Address
SHOWCB Keywords							
{ACB|EXLST|RPL}				X	X	X	X
AM		X					
AREA				X	X	X	X
FIELDS[1]		X					
LENGTH	X			X	X	X	
OBJECT		X					

1. For a list of the operands you can specify in the FIELDS parameter, see "The SHOWCB Parameter List" in Appendix G.

TESTCB Macro Operands

Keyword	Absolute Numeric	Code	Character String	Register	S-Type Address	Indirect S-Type Address	A-Type Address
TESTCB Keywords							
{ACB|EXLST|RPL}				X	X	X	X
AM		X					
ERET				X	X	X	X
OBJECT		X					
ACB Keywords							
ACBLEN	X			X	X	X	
ATRB		X					
AVSPAC	X			X	X	X	
BUFND	X			X	X	X	
BUFNI	X			X	X	X	
BUFNO	X			X	X	X	
BUFSP	X			X	X	X	
CINV	X			X	X	X	
DDNAME			X				
ERROR	X			X	X	X	
EXLST				X	X	X	X
FS	X			X	X	X	
KEYLEN	X			X	X	X	
LRECL	X			X	X	X	
MACRF		X					
MAREA				X	X	X	X
MLEN	X			X	X	X	
NCIS	X			X	X	X	
NDELR	X			X	X	X	
NEXCP	X			X	X	X	
NEXT	X			X	X	X	
NINSR	X			X	X	X	
NIXL	X			X	X	X	
NLOGR	X			X	X	X	
NRETR	X			X	X	X	
NSSS	X			X	X	X	
NUPDR	X			X	X	X	
OFLAGS		X					
OPENOBJ		X					
PARMS=CLOSDSP		X					
PASSWD				X	X	X	X
RKP	X			X	X	X	
STMST						X	
STRNO	X			X	X	X	

TESTCB Macro Operands (continued)

Keyword	Absolute Numeric	Code	Character String	Register	S-Type Address	Indirect S-Type Address	A-Type Address
EXLST Keywords							
EODAD				X	X	X	X
EXCPAD				X	X	X	X
EXLLEN	X			X	X	X	
JRNAD				X	X	X	X
LERAD				X	X	X	X
SYNAD				X	X	X	X
RPL Keywords							
ACB				X	X	X	X
AIXFLAG		X					
AIXPC	X			X	X	X	
AREA				X	X	X	X
AREALEN	X			X	X	X	
ARG				X	X	X	X
FDBK	X			X	X	X	
FTNCD	X			X	X	X	
KEYLEN	X			X	X	X	
NXTRPL				X	X	X	X
OPTCD		X					
RBA	X			X	X	X	
RECLEN	X			X	X	X	
RPLLEN	X			X	X	X	
TRANSID	X			X	X	X	

BLDVRP Macro Operands

Keyword	Absolute Numeric	Code	Character String	Register	S-Type Address	Indirect S-Type Address	A-Type Address
BUFFERS	X			X			
STRNO	X			X			
KEYLEN	X			X			

SHOWCAT Macro Operands

Keyword	Absolute Numeric	Code	Character String	Register	S-Type Address	Indirect S-Type Address	A-Type Address
DDNAME|NAME|CI				X			X
AREA				X			X
ACB				X			X
CATDSN				X			X
CATFIL				X			X

WRTBFR Macro Operands

Keyword	Absolute Numeric	Code	Character String	Register	S-Type Address	Indirect S-Type Address	A-Type Address
RPL				X			X
TYPE= DS|TRN|ALL|LRU		X					
percent	X			X			

Appendix G: Parameter Lists for VSAM Macros

The VSAM control-block manipulation macros (GENCB, MODCB, SHOWCB, and TESTCB) use an internal parameter list to describe the actions that you specify when you code the macros. The BLDVRP macro (for building a VSAM resource pool) also uses an internal parameter list to indicate the addresses and values that you specify when you code the macros. The **standard** form of these macros builds a parameter list in-line and processes it; the **list** form builds a parameter list in an area specified by the user; the **execute** form processes a previously built parameter list; the **generate** form (not for BLDVRP) builds a parameter list in an area specified by the user and also processes it.

For special purposes, such as developing high-level programming languages, you may want to build and process parameter lists without using the macros. This appendix describes the format of the parameter lists and gives the codes used for the operands of each of the macros. The formats and codes are fixed, so that you can build and alter them by your own methods.

For the control-block manipulation macros, a parameter list contains a variable number of entries of three types:

1. At the beginning of the list, **addresses** of entries of the second and third types; the addresses are fullwords, and the high-order bit of the last fullword is 1.
2. Next, a **header** entry containing general information about the block or list that you want to generate, modify, display or test.
3. At the end of the list, **keyword entries** describing each field that you want to generate, modify, display, or test.

Entries of the second and third types are described separately for GENCB, MODCB, SHOWCB, and TESTCB.

When VSAM receives control, register 1 must point to your parameter list.

The format of the BLDVRP macro is different from the above scheme; see its description at the end of this Appendix.

The GENCB Parameter List

Header Entry:

Offset Dec	Offset (Hex)			
0	(0)	block or list[1]	X'01'	number of copies
4	(4)	address of the area you are providing, or zeros		
8	(8)	length of the area, or zeros		(reserved)

1. X'A0' indicates ACB
   X'B0' indicates EXLST
   X'C0' indicates RPL

Keyword Entries: The parameter list for GENCB contains no keyword entries if you are generating a default ACB, EXLST, or RPL.

Offset Dec	Offset (Hex)		
0	(0)	keyword code[1]	(reserved)
4	(4)	{value|address|option[2]|name} of the keyword	
8	(8)	(required for some keywords[3])	

1. Defined by AL2(value):

Keyword	Value	Keyword	Value
For an ACB:		For an RPL:	
BUFND	4	ACB	43
BUFNI	5	AREA	44
BUFSP	7	AREALEN	45
DDNAME	9	ARG	46
EXLST	12	KEYLEN	48
MAREA	14	NXTRPL	51
MLEN	15	OPTCD	52
MACRF	18	RECLEN	53
PASSWD	30	TRANSID	95
STRNO	32		
CLOSDSP	151		
For an EXLST:			
EODAD	37		
EXCPAD	38		
JRNAD	39		
LERAD	40		
SYNAD	41		

2. Indicate the options for MACRF, OPTCD, and CLOSDSP with a 1 in a bit of the fullword:

MACRF Option	Bit	OPTCD Option	Bit
KEY	0	KEY	0
ADR	1	ADR	1
CNV	2	CNV	2
SEQ	3	SEQ	3
SKP	4	DIR	4
DIR	5	SKP	5
IN	6	NUP	8
OUT	7	UPD	9
NUB	8	NSP	10
UBF	9	KEQ	11
NRM	15	KGE	12
AIX	16	FKS	13
NSR	17	GEN	14
LSR	18	MVE	15
NDF	22	LOC	16
DFR	23	FWD	17
RST	28	BWD	18
NRS	29	ARD	19
		LRD	20

CLOSDSP Option	Bit
KEEP	0
DELETE	1
DATE	2

3. The third fullword is required for the ACB operand DDNAME and for all of the EXLST operands, for which the third fullword indicates [,{A|N}][,L]:

Bit	Meaning When Set to 1
0	Address is active (A)
1	Address is not active (N)
2	Address is of a field containing the name of an exit routine to be loaded (L)
3	Address is specified in the preceding fullword of this entry
4-31	Unused

The MODCB Parameter List

Header Entry:

Offset Dec	Offset (Hex)			
0	(0)	block or list[1]	X'02'	(reserved)
4	(4)	address of the block or list to be modified		

1. X'A0' indicates access-method control block (ACB)
   X'B0' indicates exit list (EXLST)
   X'C0' indicates request parameter list (RPL)

Keyword Entries:

Offset Dec	Offset (Hex)		
0	(0)	keyword code[1]	(reserved)
4	(4)	{value|address|option[2]|name} of the keyword	
8	(8)	(required for some keywords[3])	

1. Defined by AL2(value):

Keyword	Value	Keyword	Value
For an ACB:		For an RPL:	
BUFND	4	ACB	43
BUFNI	5	AREA	44
BUFSP	7	AREALEN	45
DDNAME	9	ARG	46
EXLST	12	KEYLEN	48
MAREA	14	NXTRPL	51
MLEN	15	OPTCD	52
MACRF	18	RECLEN	53
PASSWD	30	TRANSID	95
STRNO	32		
CLOSDSP	151		
For an EXLST:			
EODAD	37		
EXCPAD	38		
JRNAD	39		
LERAD	40		
SYNAD	41		

2. Indicate the options for MACRF, OPTCD, and CLOSDSP with a 1 in a bit of the fullword:

MACRF Option	Bit
KEY	0
ADR	1
CNV	2
SEQ	3
SKP	4
DIR	5
IN	6
OUT	7
NUB	8
UBF	9
NRM	15
AIX	16
NSR	17
LSR	18
NDF	22
DFR	23
RST	28
NRS	29

OPTCD Option	Bit
KEY	0
ADR	1
CNV	2
SEQ	3
DIR	4
SKP	5
NUP	8
UPD	9
NSP	10
KEQ	11
KGE	12
FKS	13
GEN	14
MVE	15
LOC	16
FWD	17
BWD	18
ARD	19
LRD	20

CLOSDSP Option	Bit
KEEP	0
DELETE	1
DATE	2

With the MODCB macro, there are no defaults for these options. When you code a bit for the OPTCD operand, the contrary bit that was previously set is turned off. For example, if KEY was previously set, and you set ADR, KEY is turned off, because a request parameter list can be set for only one type of access.

3. The third fullword is required for the ACB operand DDNAME and for all of the EXLST operands, for which the third fullword indicates [,{A|N}][,L]:

Bit	Meaning When Set to 1
0	Address is active (A)
1	Address is not active (N)
2	Address is of a field containing the name of an exit routine to be loaded (L)
3	Address is specified in the preceeding fullword of this entry
4-31	Unused

The SHOWCB Parameter List

Header Entry:

Offset Dec	(Hex)			
0	(0)	block or list[1]	X'03'	type of object to be displayed[2]
4	(4)	address of the block or list to be displayed		
8	(8)	address of the display area you are providing		
12	(C)	length of the display area		(reserved)

1. X'00' indicates that no block or list is specified to display the standard length of the block(s) or list(s) specified by the keywords ACBLEN, EXLLEN, or RPLLEN
X'A0' indicates ACB
X'B0' indicates EXLST
X'C0' indicates RPL

2. AL2(0) indicates the data of a file
AL2(1) indicates the index of a file

Keyword Entries:

Offset Dec	Offset (Hex)		
0	(0)	keyword code[1]	(reserved)

1. Defined by AL2(value):

Keyword	Value	Keyword	Value
For an ACB:		RKP	31
AVSPAC	2	STRNO	32
ACBLEN	3	STMST	35
BUFND	4	STRMAX	128
BUFNI	5		
BUFNO	6	For an EXLST:	
BUFSP	7	EODAD	37
CINV	8	EXCPAD	38
DDNAME	9	JRNAD	39
ERROR	11	LERAD	40
EXLST	12	SYNAD	41
FS	13	EXLLEN	42
MAREA	14		
MLEN	15	For an RPL:	
KEYLEN	16	ACB	43
LRECL	17	AREA	44
NCIS	19	AREALEN	45
NDELR	20	ARG	46
NEXCP	21	KEYLEN	48
NEXT	22	NXTRPL	51
NINSR	23	RECLEN	53
NIXL	24	RPLLEN	55
NLOGR	25	FDBK	56
NRETR	26	RBA	57
NSSS	27	AIXPC	58
NUPDR	28	TRANSID	95
PASSWD	30	FTNCD	99

The TESTCB Parameter List

Header Entry:

Offset Dec	Offset (Hex)			
0	(0)	block or list[1]	X'04'	type of object to be tested[2]
4	(4)	address of the block or list to be tested		
8	(8)	address of the routine to return to from unequal comparisons, or zeros		
12	(C)	(reserved)		

1. X'00' indicates that no block or list is specified to test the standard length of the block(s) or list(s) specified by the operands ACBLEN, EXLLEN, or RPLLEN

   X'A0' indicates ACB
   X'B0' indicates EXLST
   X'C0' indicates RPL

2. AL2(0) indicates the data of a file
   AL2(1) indicates the index of a file

Keyword Entries:

Offset Dec	Offset (Hex)		
0	(0)	keyword code[1]	(reserved)
4	(4)	{value|address|option[2]|name|code[3]} of the keyword	
8	(8)	(required for some keywords[4])	

1. Defined by AL2(value):

Keyword	Value	Keyword	Value
For an ACB:		RKP	31
ACBLEN	3	STRNO	32
ATRB	1	OPENOBJ	33
AVSPAC	2	STMST	35
BUFND	4	CLOSDSP	151
BUFNI	5		
BUFNO	6	For an EXLST:	
BUFSP	7	EODAD	37
CINV	8	EXCPAD	38
DDNAME	9	JRNAD	39
ERROR	11	LERAD	40
EXLST	12	SYNAD	41
FS	13	EXLLEN	42
MAREA	14		
MLEN	15	For an RPL:	
KEYLEN	16	ACB	43
LRECL	17	AREA	44
MACRF	18	AREALEN	45
NCIS	19	ARG	46
NDELR	20	KEYLEN	48
NEXCP	21	NXTRPL	51
NEXT	22	OPTCD	52
NINSR	23	RECLEN	53
NIXL	24	RPLLEN	55
NLOGR	25	FDBK	56
NRETR	26	RBA	57
NSSS	27	AIXPC	58
NUPDR	28	AIXFLAG	59
OFLAGS	29	TRANSID	95
PASSWD	30	FTNCD	99

2. Indicate the options for ATRB, MACRF, OFLAGS, OPTCD, AIXFLAG, OPENOBJ, and CLOSDSP by a 1 in a bit of the fullword:

ATRB Option	Bit
KSDS	0
ESDS	1
WCK	2
SSWD	3
REPL	4
RRDS	5
SPAN	6
UNQ	7

MACRF Option	Bit
KEY	0
ADR	1
CNV	2
SEQ	3
SKP	4
DIR	5
IN	6
OUT	7
NUB	8
UBF	9
NRM	15
AIX	16
NSR	17
LSR	18
NDF	22
DFR	23
RST	28
NRS	29

OFLAGS Option	Bit
OPEN	0

OPTCD Option	Bit
KEY	0
ADR	1
CNV	2
SEQ	3
DIR	4
SKP	5
NUP	8
UPD	9
NSP	10
KEQ	11
KGE	12
FKS	13
GEN	14
MVE	15
LOC	16
FWD	17
BWD	18
ARD	19
LRD	20

AIXFLAG Option	Bit
AIXPKP	0

OPENOBJ Option	Bit
AIX	0
PATH	1
BASE	2

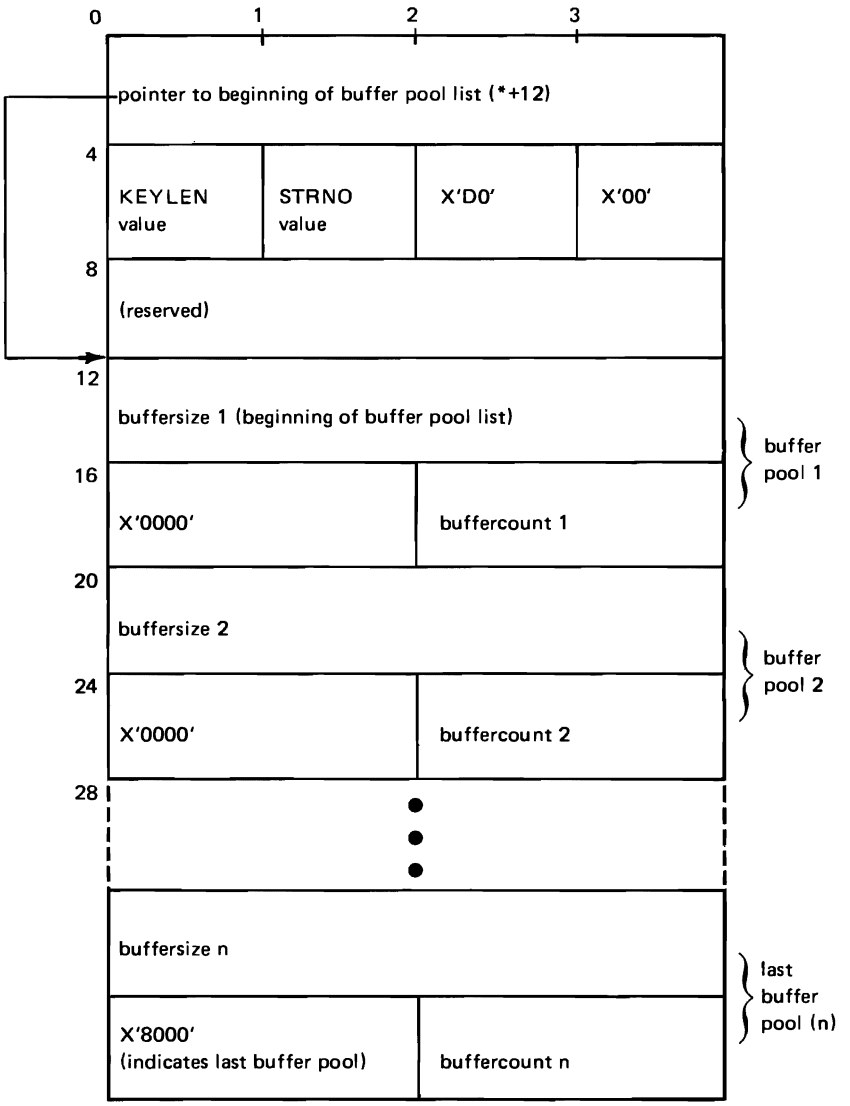
CLOSDSP Option	Bit
KEEP	0
DELETE	1
DATE	2

3. The codes for ERROR and for FDBK are documented with the appropriate macro instructions.

4. The third fullword is required for the ACB operands DDNAME and STMST and for all of the EXLST operands, for which the third fullword indicates [,{A|N}][,L]:

Bit	Meaning When Set to 1
0	Address is active (A)
1	Address is not active (N)
2	Address is of a field containing the name of an exit routine to be loaded (L)
3	Address is specified in the preceding fullword of this entry
4-31	Unused

The BLDVRP Parameter List



Appendix H: Making the VSAM Master Catalog Recoverable

You may want to convert the VSAM master catalog of an existing DOS/VSE system from a nonrecoverable catalog to a recoverable catalog. This is a summary of the steps you should follow to accomplish the conversion:

1. Install the latest release of VSAM on your system.
2. Using Access Method Services, EXPORT all those VSAM files which are to be carried over to the new catalog. Use the EXPORT PERMANENT option to automatically delete the files from the catalog.
3. Delete any other VSAM files and all non-VSAM catalog entries using the Access Method Services DELETE command.
4. Delete all VSAM data space owned by the master catalog by issuing the Access Method Services DELETE SPACE command against the master catalog's volume and all volumes owned by the master catalog. Clean up the master catalog volume by issuing DELETE MASTERCATALOG.
5. Redefine the master catalog using Access Method Services DEFINE MASTERCATALOG specifying the RECOVERABLE attribute.
6. Redefine all necessary VSAM data space with DEFINE SPACE for all applicable volumes. For each volume, the Catalog Recovery Area (CRA) is suballocated from the defined VSAM data space (the initial allocation is one cylinder/max-CA); therefore, you may want to define a larger space than existed under the nonrecoverable catalog.
7. Using Access Method Services, IMPORT the VSAM files that were exported in Step 2.
8. Define any desired non-VSAM files in a nonrecoverable VSAM user catalog using Access Method Services DEFINE NONVSAM.

Appendix I: Password Requirements for Access Method Services Commands

	CATALOG				CLUSTER/AIX			
	MASTER PW	CTLPW	UPDATE PW	READ[6] PW	MASTER PW	CTLPW	UPDATE PW	READ[6] PW
ALTER[1]	X				X			
BLDINDEX								
Input File					X	X	X	X
Output File					X	X	X	
Sort Work Files	X	X	X					
DEFINE								
AIX[2] or PATH[2]	X	X	X		X			
Cluster	X	X	X					
Non VSAM	X	X	X					
Space	X	X	X					
User Catalog	X	X	X					
DELETE								
Catalog or Path	X							
Space (empty)	X	X	X					
(non-empty)	X							
Cluster[1] or AIX[1]	X				X			
EXPORT								
User Catalog	X	X	X					
Cluster or AIX[4]	X				X			
IMPORT[5]	X	X	X		X	X	X	
LISTCAT								
With Passwords	X							
Without p/w		X	X	X				
Entries								
With p/w[1]	X				X			
Without p/w[1]		X	X	X		X	X	X
PRINT								
Catalog[3]	X							
Cluster					X	X	X	X
Component					X	X	X	X
REPRO								
Input File					X	X	X	X
Output File					X	X	X	
Catalog	X							
VERIFY								
File (Cluster, AIX, Component)					X	X		
Catalog	X							

	CATALOG				CLUSTER/AIX			
	MASTER PW	CTLPW	UPDATE PW	READ[6] PW	MASTER PW	CTLPW	UPDATE PW	READ[6] PW
EXPORTRA	X							
IMPORTRA	X							
LISTCRA	X							
RESETCAT	X							

This table summarizes which passwords may be supplied when using the Access Method Services commands. Where more than one password is indicated, any of the indicated passwords may be used. For example; to alter an entry in a password-protected catalog, you must supply the master password of either the catalog or cluster/AIX.

[1] Either catalog or cluster password may be specified.

[2] Both catalog and cluster passwords are checked if both are password-protected.

[3] The master password is required only if passwords are to be printed; otherwise the read password may be used.

[4] If any password-protected paths are defined over the cluster or alternate index being exported, you must supply the master password of the catalog in order to export the protection attribute of all the paths.

[5] Either the catalog or cluster password can be specified. If you are importing into a predefined empty file with passwords that are different from the old passwords of the file being imported, you must specify the file's update or higher-level password.

[6] The READPW is inadequate if the DLBL statement describing the file specifies one of the following:

DISP=NEW
DISP=(,DELETE)
DISP=(,DATE)

You must provide the UPDATEPW (or higher-level password) in these cases.

Passwords Specified by the User				Passwords Assigned by VSAM			
READPW	UPDATEPW	CONTROLPW	MASTERPW	READPW	UPDATEPW	CONTROLPW	MASTERPW
R				R	R	R	R
R		C		R		C	C
R	U			R	U	U	U
R			M	R			M
	U		M		U		M

Appendix J: Release 1 Command Parameters

ALTER Parameters

CATALOG (*catname* [/ *password*] [ъ *dname*])
specifies the name of the catalog that contains the entry to be altered.

catname – specifies the name of the catalog.

password – specifies the master password of the catalog. If the entry to be altered is password-protected and the catalog is also password-protected, you can supply the proper password, either through this parameter or through the *entryname* parameter. If both passwords are specified, the catalog password is used.

dname – specifies the DLBL statement that identifies the catalog, when that catalog is not the catalog obtained by default. VSAM defaults to the job catalog (IJSYSUC) if a job catalog is being used; otherwise it defaults to the master catalog. See "VSAM's Use of Catalogs" in *VSE/VSAM Programmer's Reference* for information about the order in which catalogs are searched.

Abbreviation: CAT

FILE (*dname*)
specifies the *filename* of the DLBL job control statement that, together with an EXTENT statement, identifies volume(s) that must be mounted.

FILE is required when a nonVSAM or unique file is to be renamed (by means of the NEWNAME parameter) or if the entry being altered is in a recoverable catalog. If a nonVSAM or unique file is being renamed, the logical unit number and the volume serial number of the volume that contains the file must be specified in the EXTENT job control statement.

If the entry being altered is in a recoverable catalog, the volume serial number of the recovery volume on which space was allocated for the object defined by the entry must be specified in the EXTENT job control statement. For a definition of recovery volume, see "Recovering Catalog Entries and Data" in *VSE/VSAM Programmer's Reference.* The DEFINE command prints the volume serial number of the recovery volume for the respective object.

BLDINDEX Parameters

INFILE (*dname* [/ *password*])
specifies the *filename* of the DLBL job control statement that identifies a path that points to the base cluster or the base cluster itself. If the base cluster is password protected, you may give any one of its passwords. If a path over the base cluster is specified, the password must be that of the path. **This parameter is required.**

Abbreviation: IFILE

OUTFILE (*dname* [/ *password*] . . . .)
specifies the *filename* of the DLBL job control statement that identifies a path that points to the alternate index or the alternate index itself. If the alternate index is password protected, you must give its update or higher-level password. If a path over the alternate index is specified, the password must be that of the path. More than one alternate index may be built simultaneously by specifying multiple *dname* subparameters. **This parameter is required.**

Abbreviation: OFILE

WORKFILES (*dname1* b *dname2*)
identifies the job control statements identifying work files that will be used if an external sort is performed. Specify this parameter only if work file *dnames* other than IDCUT1 and IDCUT2 are to be used in the case of an external sort. Two sort work files will be dynamically defined by BLDINDEX and opened to perform an external sort. Sort work files will be suballocated from a VSAM data space on the identified volume(s); therefore, sufficient space must already have been defined. See the section "Building an Alternate Index" for a description of how to calculate the amount of space to be suballocated for each sort work file. The sort work files may be multivolume files with up to five volumes each. The files are deleted by BLDINDEX prior to the end of processing the command.

Abbreviations: WORKFILE, WFILE, WFILES

Default: IDCUT1 and IDCUT2

DEFINE ALTERNATEINDEX Parameters

CATALOG (*catname* [/ *password*] [b *dname*])
identifies the catalog that contains the entry of the base cluster named in the RELATE parameter. The alternate index is always cataloged in the same catalog as its base cluster.

catname – specifies the name of the catalog.

password – specifies the update or higher-level password if the catalog is password protected. If no password is specified and the catalog is password protected, VSAM asks the operator for the correct password. If the base cluster's master password is not specified with the RELATE parameter, then the catalog's master password must be specified.

dname – specifies the *filename* of the DLBL job control statement that identifies the catalog. This subparameter must be specified unless the catalog is the job catalog (IJSYSUC), if this is used, or the master catalog (IJSYSCT), if no job catalog is used.

Abbreviation: CAT

FILE (*dname*)
specifies the *filename* of the DLBL job control statement that, together with an EXTENT statement, identifies the logical units and volumes to be used for space allocation. The pertinent volumes must be mounted during the define operation.

You must specify the FILE parameter if you specify UNIQUE or if the alternate index is cataloged in a recoverable catalog. All devices for a unique component of an alternate index must be of the same device type.

A FILE parameter must be specified for each unique component, individually or on the alternate index level. If both components of an alternate index are unique and reside on separate volumes, a single FILE parameter may be specified at the alternate index level. This case requires one DLBL statement with EXTENT statements for each of the two components. You must specify allocation information in the EXTENT statement(s) that corresponds to a unique component. If both components are unique and reside on the same volume, FILE must be specified for both DATA and INDEX. In this case the components cannot share the same DLBL/EXTENT statements.

If the alternate index is being defined in a recoverable catalog, DLBL and EXTENT statements describing the first volume containing the base cluster's index component (if the base cluster is a key-sequenced file) or data component (if the base cluster is an entry-sequenced file) must be provided.

MODEL (*entryname* [/ *password*]
[ƀ *catname* [/ *password*] [ƀ *dname*]])

specifies that the entry of an already-defined alternate index is to be used as a model for the entry being built. See "How to Use One Object as a Model for Another Object" in *VSE/VSAM Programmer's Reference* for more information about MODEL.

entryname – specifies the name of the entry to be used as a model. The entry to be used as a model must be of the same entry type as the entry being built. For example, if you are using this parameter at the data level, the object used as a model must be a data component.

password – specifies a password. If the alternate index or component whose entry is to be used as a model is password protected and is cataloged in a password protected catalog, either the alternate index or component's password or its catalog's password is required. If both the entry password and the catalog password are specified, the catalog password will be used. If the protection attributes are to be copied, give the master password of either the model (following *entryname*) or the catalog (following *catname*). If the model's passwords are not to be copied, any password of either the model or its catalog can be used.

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

dname – specifies the *filename* of the DLBL job control statement that identifies the catalog that contains the entry to be used as a model. This parameter is required if the catalog is neither the master catalog nor a job catalog identified by a job control statement.

Restrictions: Your job may be terminated because of allocation problems if you use the MODEL parameter and (1) specify a device type different from that specified for the model through the VOLUMES parameter, (2) change the length of keys through the KEYS parameter, (3) change the size of records, buffer space, or control intervals through the RECORDSIZE, BUFFERSPACE, or CONTROLINTERVALSIZE parameter, (4) change from UNIQUE to SUBALLOCATION, or vice

versa, or (5) change the unit of allocation through the TRACKS, BLOCKS, CYLINDERS, or RECORDS parameter.

If modeling causes the alternate index or component to be defined as unique, FILE must also be specified.

DEFINE CLUSTER Parameters

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

catname – specifies the name of the catalog.

password – specifies the update or higher-level password if the catalog is password protected. If no password is specified and the catalog is password protected, VSAM asks the operator for the correct password.

dname – specifies the *filename* of the DLBL job control statement that identifies the catalog. See "Appendix I: Using Job Control to Indicate a Catalog." You must specify *dname* unless the catalog is the job catalog (IJSYSUC), if this is used, or the master catalog (IJSYSCT) if no job catalog is used.

Abbreviation: CAT

FILE (*dname*)
specifies the ***filename*** of the DLBL job control statement that, together with an EXTENT statement, identifies the logical units and volumes to be used for space allocation. You must specify the FILE parameter if you specify UNIQUE or if the cluster is cataloged in a recoverable catalog.

All devices for a component of a cluster must be of the same device type.

A FILE parameter must be specified for each unique component individually or on the cluster level. If both components of a key-sequenced file are unique and reside on separate volumes, a single FILE parameter may be specified at the cluster level. This case requires one DLBL statement with EXTENT statements for each of the two components. You must specify allocation information in the EXTENT statement(s) that corresponds to a unique component.

If both components are unique and reside on the same volume, FILE must be specified for both DATA and INDEX. In this case the components cannot share the same DLBL/EXTENT statements.

If the cluster is being defined in a recoverable catalog, DLBL and EXTENT statements describing the first volume containing the base cluster's index component (if the base cluster is a key-sequenced file) or data component (if the base cluster is an entry-sequenced file) must be provided. The pertinent volumes must be mounted during the define operation.

MODEL (*entryname* [/ *password*] [ђ *catname* [/ *password*] [ђ *dname*]])
specifies that the entry of an already-defined cluster is to be used as a model for the entry being built. See "How to Use One Object as a Model For Another Object" in *VSE/VSAM Programmer's Reference* for more information about MODEL.

entryname – specifies the name of the entry to be used as a model. The entry to be used as a model must be of the same entry type as the entry being built. For example, if you are using this parameter at the data level, the object used as a model must be a data component.

password – specifies a password. If the cluster or component whose entry is to be used as a model is password protected and is cataloged in a password protected catalog, either the cluster or component's password or its catalog's password is required. If both the entry password and the catalog password are specified, the catalog password will be used. If the protection attributes are to be copied, give the master password of either the model (following *entryname*) or the catalog (following *catname*). If the model's passwords are not to be copied, any password of either the model or its catalog can be used.

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

dname – specifies the *filename* of the DLBL job control statement that identifies the catalog that contains the entry to be used as a model. This parameter is required if the catalog is neither the master catalog nor a job catalog identified by a job control statement.

Restrictions: Your job may be terminated because of allocation problems if you use the MODEL parameter and (1) specify a device type different from that specified for the model through the VOLUMES parameter, (2) change the length of keys through the KEYS parameter, (3) change the size of records, buffer space, or control intervals through the RECORDSIZE, BUFFERSPACE, or CONTROLINTERVALSIZE parameter, (4) change from UNIQUE to SUBALLOCATION, or vice versa, or (5) change the unit of allocation through the TRACKS, BLOCKS, CYLINDERS, or RECORDS parameter.

If modeling causes the cluster or component to be defined as UNIQUE, FILE must also be specified.

DEFINE MASTERCATALOG Parameter

FILE(*dname*)
specifies the *filename* of the DLBL job control statement that, together with an EXTENT statement, identifies the logical unit, volume to be used for the master catalog (always IJSYSCT), and allocation information. **This parameter is required.**

DEFINE NONVSAM Parameter

CATALOG(*catname* [/ *password*] [ъ *dname*])
identifies the catalog in which the nonVSAM file is to be defined. The catalog must not be recoverable.

catname – specifies the name of the catalog.

password – specifies the update or higher-level password of the catalog if it is password protected.

dname – specifies the *filename* of the DLBL job control statement that identifies the catalog. This subparameter must be specified unless the catalog is either the master catalog (IJSYSCT) or the job catalog (IJSYSUC).

Abbreviation: CAT

DEFINE PATH Parameters

CATALOG(*catname* [/ *password*] [b *dname*])
identifies the catalog that defines the alternate index or the base cluster named in the PATHENTRY parameter.

catname – specifies the name of the catalog.

password – specifies the update or higher-level password if the catalog is password protected. If no password is specified and the catalog is password protected, VSAM asks the operator for the correct password.

dname – specifies the *filename* of the DLBL job control statement that identifies the catalog. This subparameter must be specified unless the catalog is the job catalog (IJSYSUC), if this is used, or the master catalog (IJSYSCT), if no job catalog is used.

Abbreviation: CAT

FILE(*dname*)
specifies the *filename* of the DLBL statement that, together with an EXTENT statement, identifies the recovery volume of the alternate index or base cluster named in the PATHENTRY parameter, if the path is being defined in a recoverable catalog. The recovery volume is the first volume of the index component for the base cluster (if the cluster is a key-sequenced file), or the first volume of the data component for the base cluster (if the cluster is an entry-sequenced file).

MODEL(*entryname* [/ *password*]
[b *catname* [/ *password*] [b *dname*]])
specifies that the entry of an already-defined path is to be used as a model for the entry being built. See "How to Use One Object as a Model for Another Object"in *VSE/VSAM Programmer's Reference* for more information about MODEL.

entryname – specifies the name of the path entry to be used as a model.

password – specifies a password. If the path whose entry is to be used as a model is password protected and is cataloged in a password protected catalog, either the path's password or its catalog's password is required. If both the entry password and the catalog password are specified, the catalog password is used. If the protection attributes are to be copied, substitute the master password of either the path (following *entryname*) or its catalog (following *catname*). If the model's passwords are not to be copied, any password of either the path or its catalog can be used.

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

dname – specifies the *filename* of the DLBL statement that identifies the catalog that contains the entry to be used as a model. This parameter is required if the catalog is neither the master catalog nor a job catalog identified by a job control statement.

DEFINE SPACE Parameters

CATALOG (*catname* [/ *password*] [ƀ *dname*])
identifies the catalog in which the data space(s) are to be defined or in which candidate volume(s) are to be indicated.

catname – specifies the name of the catalog.

password – specifies the update or higher-level password if the catalog is password protected.

dname – specifies the *filename* of the DLBL statement that identifies the catalog. You must specify *dname* unless the catalog is the job catalog (IJSYSUC), if this is used, or the master catalog (IJSYSCT), if no job catalog is used.

Abbreviation: CAT

FILE (*dname*)
specifies the *filename* of the DLBL job control statement that, together with an EXTENT statement(s), identifies the logical unit(s) and volumes(s) to be used for space allocation. The corresponding EXTENT statement(s) must contain allocation information unless CANDIDATE is specified.

DEFINE USERCATALOG Parameters

FILE (*dname*)
specifies the *filename* of the DLBL job control statement that, together with the EXTENT statement, identifies the logical unit, volume to be used for the user catalog, and allocation information.

MODEL (*entryname* [/ *password*]
[ƀ *catname* [/ *password*] [ƀ *dname*]])
specifies that an existing user or master catalog is to be used as a model for the user catalog being defined. 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 you specify the MODEL parameter you must specify the following four parameters in the DEFINE command even if no attributes of the model are to be changed or added:

- Space allocation parameter (BLOCKS, CYLINDERS, RECORDS, or TRACKS) at the user catalog level.
- FILE(dname)
- NAME(entryname)
- VOLUME(volser)

In regards to the space allocation parameter at the user catalog level, the recommended approach is to specify the same value as you specified in the job control EXTENT statement. This avoids the possibility of an improper specification because the total allocated data space is controlled by the

EXTENT statement, and the area suballocated to the catalog itself is controlled by:

- The space allocation attributes previously established by the model catalog, or, by
- The explicit specification of space parameters (in the current DEFINE USERCATALOG command) at the data or index component level (an explicit specification overrides the model attributes).

For more information about MODEL, see "How to Use One Object as a Model For Another Object" in *VSE/VSAM Programmer's Reference.*

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

password – specifies a password of the catalog to be used as a model if it is password protected. If both the entry password and the catalog password are specified, the catalog password is used. If the protection attributes are to be copied, you must specify the master password. If the protection attributes are not to be copied, you may specify any password.

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

dname – specifies the *filename* of the DLBL statement that identifies the catalog that contains the entry to be used as a model. This parameter is required if the catalog is neither the master catalog nor a job catalog.

The VOLUME parameter cannot be modeled and must be specified to uniquely identify the volume.

Your job may be terminated because of allocation problems if you use the MODEL parameter and (1) specify a device type different from that specified for the model through the VOLUME parameter, or (2) change the buffer space through the BUFFERSPACE parameter.

DELETE Parameters

CATALOG (*catname* [/ *password*] [b *dname*])
specifies the name of the catalog that contains the entries to be deleted. CATALOG cannot be specified when a user or master catalog that is password-protected is to be deleted; instead, specify the name of the user or master catalog in the *entryname* parameter, together with its master password. Catalog must be specified when the FORCE parameter is used to scratch nonempty data spaces from the VTOC if the catalog owning the volume to be scratched is password-protected.

catname – identifies the catalog that contains the entry to be deleted.

password – specifies the master password of the catalog. If objects 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.

dname – specifies the *filename* of the DLBL job control statement that identifies the catalog that defines the objects to be deleted. The *dname* is required if the desired catalog is neither the master nor job catalog.

Abbreviation: CAT

FILE (*dname*)
specifies the *filename* of the DLBL job control statement that identifies the object to be deleted. This parameter is required when:

- The object to be deleted is also to be erased. (If the file was previously defined in a user catalog, you must explicitly name the user catalog by specifying the CAT parameter in the file's DLBL job control statement. See "DELETE ERASE With Explicit User Catalog" in Appendix I.)
- The object to be deleted is a data space or a unique file. (EXTENT statements must identify all volumes associated with the data space or unique file.)
- The object to be deleted is a nonVSAM file and the file is to be scratched, that is, removed from the VTOC.
- The object to be deleted is in a recoverable catalog.

If the object being deleted is defined in a recoverable catalog, an EXTENT statement associated with the DLBL job control statement must also specify the recovery volume (that is, the volume whose CRA contains a copy of the object being deleted.) FILE cannot be specified when MASTERCATALOG or USERCATALOG is specified.

EXPORT Parameter

INFILE(*dname*)
specifies the name of the DLBL statement for the file (cluster or alternate index) to be exported. The DLBLs associated EXTENT statement(s) must identify all volumes on which the file resides. **This is a required parameter for exporting files.**

The file that is to be exported must have been defined in the job catalog (IJSYSUC), or, if no job catalog is being used, in the master catalog (IJSYSCT).

Abbreviation: IFILE

EXPORTRA Parameters

CRA((*dname1* {ƀALL[ƀINFILE(*dname2*)] |ƀNONE
|ƀENTRIES((*entryname* [ƀ *dname3*]) . . .) }) . . .)
specifies the job control statements that identify the CRA and the files to be recovered, and the statements that identify the volumes that contain multivolume VSAM files. The CRA parameter sublist is specified for each CRA to be accessed, whether or not any files are to be recovered from it.

The maximum repetition for CRA subparameters is 20.

dname1 – specifies the *filename* of the DLBL/EXTENT statement that identifies the volume that contains the CRA from which files are to be recovered. (The DLBL statement must only specify *filename* and the 'VSAM' code. Do not specify the *file-ID*.)

All volumes identified by the *dname1* parameter must be mounted before the command is executed.

ALL
indicates that all of the catalog entries recorded in the CRA on the volume indicted by *dname1* are to be recovered. This is the default.

INFILE(*dname2*)
identifies multiple volumes (including any candidate volumes) needed to recover the files recorded in the CRA identified via *dname1*. The CRA on each of these volumes must also be identified as *dname1* in

another CRA parameter sublist. Omit INFILE if the CRA identified by *dname1* contains no files on any other volumes.

See "Using EXPORTRA for Entries on Multiple Volumes" for further information and an example of the INFILE parameter.

dname2 – specifies the *filename* of the DLBL/EXTENT statement(s) that identify all the volumes (including the CRA volume) needed to recover the files from the CRA. (The DLBL statement must only specify *filename* and the 'VSAM' code. Do not specify the *file-ID*.)

Abbreviation: IFILE

NONE
indicates that none of the files recorded in this volume's CRA are to be recovered. The CRAs for all volumes in all multivolume files that are to be recovered must be specified as a *dname1* parameter. See "Using EXPORTRA For Entries on Multiple Volumes" for further information and an example of the NONE parameter.

Be aware that the CRA for an alternate index is always on the same volume as its base cluster even though the alternate index can be on a different volume.

Restriction: *dname1* NONE must be specified for each volume identified via a *dname2* or *dname3* DLBL/EXTENT statement if you do not wish the entries in its CRA to be recovered.

ENTRIES((*entryname* [*dname3*]) . . .)
specifies which entries are to be recovered from this CRA. The maximum repetition for ENTRIES subparameters is 100.

entryname – specifies the name of the alternate index or cluster for which catalog records and data are to be recovered, or the name of the user catalog whose catalog records are to be recovered. See "Using EXPORTRA For Selected Entries" for further information and an example of the ENTRIES subparameter.

dname3 – specifies the *filename* in the DLBL/EXTENT statement that identifies the volumes (including any candidate volumes) containing this file. The CRA on each of these volumes must also be identified as *dname1* in another CRA parameter sublist. The DLBL and the corresponding EXTENT statements must specify the volume serial number, the 'VSAM' code, and logical unit number (since the catalog containing the file is presumably not available). *File-ID* should not be specified. This subparameter is not required for a user catalog or for a file that resides entirely on one volume.

Abbreviation: ENT

IMPORT Parameters

FILE(*dname*)
specifies the *filename* of the DLBL statement that, together with associated EXTENT statement(s), identifies the volume(s) and extents allocated to the data and index components of an alternate index or key-sequenced cluster that is to be imported.

This parameter is required only when the data and index components (of an alternate index or key-sequenced file) reside on the same volume and

one or both components are defined with the UNIQUE attribute. (If only one component is unique you code one FILE parameter; if both components are unique you code two FILE parameters)

The *dname* (and DLBL/EXTENT statements) must be different from the *dname* specified in the OUTFILE parameter (the data and index component *dnames* must also be different from each other).

Do not specify the FILE parameter when importing into a predefined empty file.

If you rename the data or index component with the NEWNAME parameter, the file-ID in the DLBL statement must be the same as the new name. Otherwise file-ID must be the existing component name.

`OUTFILE(` *dname* [*/ password*])

dname – specifies the *filename* of the DLBL statement that, together with associated EXTENT statement(s), identifies the volumes and extents allocated to the alternate index or cluster to be imported.

password – is required only if importing into a predefined empty file with passwords which are different from the old passwords of the file being imported. The password must be the file's update or higher level password.

If you rename the cluster or alternate index with the NEWNAME parameter, the *file-ID* in the DLBL statement must be the same as the new name. Otherwise, *file-ID* must be the existing cluster or alternate index name.

All devices for a component of a cluster or alternate index must be of the same device type.

The EXTENT statement(s) must include the extent allocation information for the file when importing an entry-sequenced or relative record file that is defined as unique (unless importing into a predefined empty file).

The EXTENT statement(s) must include the extent allocation information for the unique component(s) (unless importing into a predefined empty file) when importing a key-sequenced file or alternate index with one or more components defined unique and the components do not have any volumes in common (for unique components with common volumes, see the FILE parameter).

Note: If you are importing onto new volume(s) in a recoverable catalog that contains a copy of the cluster or alternate index (from EXPORT TEMPORARY), you must supply an EXTENT statement identifying the CRA volume containing the old entry so that it can be deleted.

Abbreviation: OFILE

IMPORTRA Parameters

`FILE(` *dname*)

specifies the *filename* of the DLBL statement that, together with associated EXTENT statement(s), identifies the volume(s) and extents allocated to the data and index components of an alternate index or key-sequenced cluster that is to be imported.

This parameter is required only when the data and index components (of an alternate index or key-sequenced file) reside on the same volume and one or both components are defined with the UNIQUE attribute. (If only

one component is unique you code one FILE parameter; if both components are unique you code two FILE parameters.)

Do not specify the FILE parameter for the importation of suballocated (non-unique) components.

The *entryname* parameter preceding the FILE parameter cannot be a cluster or alternate index name; it must be a data or index component name.

The *file-ID* in the DLBL statement pointed to by *dname* must be the same as the component name (that is, the *entryname* parameter preceding the FILE parameter).

OUTFILE(*dname*)
specifies the *filename* in the DLBL statement used to identify the volumes onto which IMPORTRA will import VSAM files. The *file-ID* on the DLBL statement must not appear in the catalog or among any of the files on the portable volume. (The files on the portable volume are those which were exported by EXPORTRA.) The *file-ID* specified is used by IMPORTRA during its processing only and will not exist after the job terminates. The DLBL statement must have associated EXTENT statements for all of the volumes to be used by IMPORTRA.

Note: If you are importing onto a new volume(s) via the VOLUMES parameter, you must supply EXTENT statements for the CRA volume(s) containing the old entries of the files so that VSAM can delete them.

Abbreviation: OFILE

LISTCAT Parameter

CATALOG(*catname* [/ *password*] [b *dname*])
specifies the catalog that contains the entries that are to be listed. If CATALOG is not specified, the job catalog specified by a job control statement or the master catalog itself is listed.

catname – is the name of the catalog.

password – specifies the read or higher-level password of the catalog if it is password protected. If the entries to be listed contain password protection information and this information is to be listed, then the master password must be supplied either through this parameter or through the ENTRIES parameter.

dname – specifies the name of the DLBL statement that identifies the catalog if the desired catalog is not the master or job catalog.

Abbreviation: CAT

LISTCRA Parameters

CATALOG(*catname* [/ *password*] b *dname*)
specifies the name of the catalog that owns the volume containing the CRA. If COMPARE is specified, entries in this catalog will be compared with entries in the CRA.

catname – is the name of the catalog.

password – specifies the master password of the catalog identified by *catname* (must be specified if COMPARE is specified and the catalog is password protected.)

dname – specifies the *filename* of the DLBL job control statement that identifies the catalog as a standard VSAM file to enable LISTCRA to read records from it.

Abbreviation: CAT

Restriction: If COMPARE is specified, this parameter must also be specified. Cannot be specified with SEQUENTIALDUMP. If the CATALOG parameters are specified incorrectly, the NOCOMPARE default is taken and all of the CRA records are listed.

`INFILE(` *dname* `[b...])`

specifies the *filename* of the DLBL/EXTENT job control statements that identify the volume of the CRA to be listed.More than one CRA may be listed if multiple *dname* subparameters are supplied.

All volumes identified by the *dname* parameter must be mounted before the command is executed.

Abbreviation: IFILE

PRINT Parameter

`INFILE(` *dname* `[/` *password* `] [b ENVIRONMENT (` *subparameters* `) ])`

describes the input file. If the file was previously defined in a user catalog, you must explicitly name the user catalog by specifying the CAT parameter in the file's DLBL job control statement. *dname* – specifies the *filename* of the DLBL or TLBL statement that identifies the input file to be listed. **This is a required parameter**. You can list a base cluster in alternate key sequence by specifying a path name as the *file-ID*.

password – specifies the password of a password protected VSAM file or component. For printing a data or index component, the password to be supplied is the read or higher-level password of the component or the master password of the cluster or alternate index to which the component belongs; for a path, it is the password of the path.

Abbreviation: IFILE

REPRO Parameters

`INFILE(` *dname* `[/` *password* `] [b ENVIRONMENT (` *subparameters* `) ])`

describes the input file. If the file was previously defined in a user catalog, you must explicitly name the user catalog by specifying the CAT parameter in the file's DLBL job control statement.

dname – specifies the *filename* of the DLBL or TLBL job control statement that identifies the file to be copied. You can process a base cluster in alternate key sequence by specifying a path name as *file-ID*. **This is a required parameter.**

password – the read or higher-level password of the password-protected file to be copied. If a catalog is to be copied, the catalog's master password is required. For a path, it is the password of the path.

Abbreviation: IFILE

OUTFILE (*dname* [/ *password*] [ƀENVIRONMENT (*subparameters*)])
specifies the *filename* of the DLBL or TLBL job control statement that identifies the output file. **This is a required parameter.** For VSAM files, the *file-ID* may be that of a path. A master catalog must be copied to a nonVSAM file.

If the file was previously defined in a user catalog, you must explicitly name the user catalog by specifying the CAT parameter in the file's DLBL job control statement.

password – the update or higher-level password of the VSAM file being used for output if it is password protected. If a catalog is to be unloaded, the catalog's master password is required. For a path, it is the password of the path.

Abbreviation: OFILE

RESETCAT Parameters

CATALOG (*catname* [/ *password*] ƀ *dname*)
identifies the catalog to be reset; it is a required parameter. The catalog specified by *catname* must have been defined with the RECOVERABLE attribute. Exclusive control of the catalog is required throughout the duration of the command; also, no VSAM files cataloged in the catalog being recovered may be open. The master catalog may be reset while it is in use as a master catalog, or it may be connected to the system as a user catalog.

password – specifies the master password of the catalog if it is password protected.

dname – specifies the *filename* of the DLBL statement for the catalog being reset. The same catalog (the file-ID of the catalog being reset) must also be specified in an IJSYSUC DLBL statement

Abbreviation: CAT

CRAFILES ((*dname* { ƀALL | NONE }) [(*dname* { ƀALL | NONE }) . . .])
specifies a list of DLBL/EXTENT statements that identify (via the *dname* subparameter) all the volumes to be used to reset the catalog. You must specify the *dname* subparameter for each CRA to be accessed, whether or not any catalog records are to be reset in the catalog (see the NONE subparameter for more information).

See *VSE/VSAM Programmer's Reference* for further information about the contents of CRAs and how catalog entries are placed in CRAs.

dname – specifies the *filenames* of the DLBL/EXTENT statements that identify the volumes containing CRAs to be accessed.

All volumes identified by the *dname* parameter must be mounted before the command is executed.

ALL – specifies that the catalog records in the CRA on the volume specified by *dname* are to be reset in the catalog. This is the default.

NONE – specifies that the catalog records in the CRA of the volume specified by *dname* are not to be reset in the catalog. If a volume being recovered contains a multivolume file, other volumes of that multivolume file may be needed during the reset operation; they must be specified in job control statements and NONE specified after the CRAFILES *dname*

parameter for the volume if you do not wish the entries in the CRA's of the additional volumes to be reset.

WORKCAT (*catname* [/ *password*] [*dname*)

specifies the name of the catalog where the WORKFILE is to be defined. **This is a required parameter.** This catalog must not be the catalog being reset.

password – specifies the update or higher level password if the catalog is password protected.

dname – specifies the job control statement for the work catalog. The work catalog must be specified in the CAT parameter of the DLBL statement for the workfile. This catalog must not be the catalog being reset.

Abbreviation: WCAT

WORKFILE (*dname* [/ *password*])

specifies the *filename* of the DLBL/EXTENT statement that identifies the workfile to be used by RESETCAT. The DLBL statement contains the VSAM *file*-ID to be used for the workfile and the EXTENT statement(s) contain volume serial numbers and units to be used by the workfile. If WORKCAT is not the default catalog, the DLBL statement must contain the CAT=filename parameter. The VSAM file will be defined by RESETCAT and used as temporary storage while processing the command; it will be deleted at the end of the command. This file must not already be defined. If WORKFILE is not specified, IDCUT1 is used as a default *dname* of a specified DLBL statement. See "WORKFILE Space Requirements" in "Using RESETCAT: Reseting Catalog Entries" for WORKFILE space requirements.

password – If desired, *password* may be specified to protect the workfile. Since the workfile will contain a copy of the catalog records, it may contain security sensitive information. The *password* specified becomes the workfile's master password. The *password* may be specified only if *dname* is explicitly specified.

Abbreviation: WFILE

VERIFY Parameter

FILE (*dname* [/ *password*])

dname – specifies the *filename* of the DLBL job control statement that identifies the object to be verified. The DLBL statement must also contain the name of the object (*file-ID*).

If the file was previously defined in a user catalog, you must explicitly name the user catalog by specifying the CAT parameter in the file's DLBL job control statement.

password – is the control or master password of a password protected cluster, alternate index, or component, or the master password of a password protected catalog.

Appendix K: Acronyms and Abbreviations

AIX	ALTERNATEINDEX
ALC	ALLOCATION
ATT	ATTEMPTS
AUTH	AUTHORIZATION
AVOL	ADDVOLUMES
BIX	BLDINDEX
BLK	BLOCKS
BLKSZ	BLOCKSIZE
BLOCK	BLOCKS
BUFSP	BUFFERSPACE
BUFSPC	BUFFERSPACE
CAN	CANDIDATE
CAT	CATALOG
CHAR	CHARACTER
CIM	CIMODE
CISZ	CONTROLINTERVALSIZE
CL	CLUSTER
CMPR	COMPARE
CNVSZ	CONTROLINTERVALSIZE
CON	CONNECT
CRA	catalog recovery area
CRAFILE	CRAFILES
CRAVOL	CRAVOLUMES
CTLPW	CONTROLPW
CYL	CYLINDERS
DCON	DISCONNECT
DED	DEDICATE
DEF	DEFINE
DEL	DELETE
DEVT	DEVICETYPES
DFVOL	DEFAULTVOLUME
DLBL	DASD label
DS	DATASET
DTF	define the file
DTFIS	indexed sequential DTF
DTFMT	magnetic tape DTF
DTFSD	sequential disk DTF
EEXT	EXCEPTIONEXIT
ENT	ENTRY, ENTRIES
ENV	ENVIRONMENT
ERAS	ERASE
ESORT	EXTERNALSORT
EXP	EXPORT
F	fixed unblocked
FADDR	FROMADDRESS
FB	fixed blocked
FIXBLK	fixed blocked
FIXUNB	fixed unblocked
FKEY	FROMKEY
FNUM	FROMNUMBER
FRC	FORCE
FSEQN	FILESEQUENCENUMBER
FSPC	FREESPACE
GRPH	GRAPHICS
HDEV	HINDEXDEVICE
ID	identification
IDS	INDATASET
IFILE	INFILE
IGN	IGNORE
IMBD	IMBED
IMP	IMPORT
INH	INHIBIT
INHS	INHIBITSOURCE
INHT	INHIBITTARGET
ISORT	INTERNALSORT
IVOL	INVOLUMES
IX	INDEX
IXD	INDEXED
KRNG	KEYRANGES
LISTC	LISTCAT
LISTR	LISTCRA
MAR	MARGINS
MCAT	MASTERCATALOG
MPRA	IMPORTRA
MDLE	MODULE
MRCAT	MASTERCATALOG
MRPW	MASTERPW
NAL	NOALLOCATION
NCMPR	NOCOMPARE
NERAS	NOERASE
NEWNM	NEWNAME
NFRCE	NOFORCE
NIGN	NOIGNORE
NIMBD	NOIMBED
NINHS	NOINHIBITSOURCE
NINHT	NOINHIBITTARGET
NIXD	NONINDEXED
NLBL	NOLABEL
NPRG	NOPURGE
NREP	NOREPLACE
NREPL	NOREPLICATE
NREW	NOREWIND
NRUS	NOREUSE
NRVBL	NOTRECOVERABLE
NSCR	NOSCRATCH
NSPND	NONSPANNED
NULL	NULLIFY
NUMD	NUMBERED
NUNQK	NONUNIQUEKEY
NUPD	NOUPDATE
NUPG	NOUPGRADE
NUS	NOTUSABLE
NVSAM	NONVSAM
NWCK	NOWRITECHECK
OBJ	OBJECTS
ODS	OUTDATASET
OFILE	OUTFILE
OPW	OUTPW
ORD	ORDERED
ORG	ORIGIN
PDEV	PRIMEDATADEVICE
PENT	PATHENTRY
PERM	PERMANENT
PRG	PURGE
PW	password
RBA	relative byte address
RCAT	RESETCAT
RCVY	RECOVERY
RDPW	READPW
REC	RECORDS
RECFM	RECORDFORMAT
RECM	RECORDMODE
RECSZ	RECORDSIZE
REL	RELATE
REP	REPLACE
REPL	REPLICATE
RETN	RETENTION
REW	REWIND
RUS	REUSE
RVBL	RECOVERABLE
RVOL	REMOVEVOLUMES
S	variable spanned unblocked
SB	variable spanned blocked
SCR	SCRATCH
SDUMP	SEQUENTIALDUMP
SHR	SHAREOPTIONS
SLBL	STDLABEL
SPC	SPACE
SPNBLK	variable spanned blocked
SPND	SPANNED
SPNUNB	variable spanned unblocked
STRG	STRING

SUBAL	SUBALLOCATION
TADDR	TOADDRESS
TEMP	TEMPORARY
TLBL	tape label
TNUM	TONUMBER
TRK	TRACKS
U	undefined
UA	unassigned
UCAT	USERCATALOG
UNDEF	undefined
UNINH	UNINHIBIT
UNLD	UNLOAD
UNORD	UNORDERED
UNQ	UNIQUE
UNQK	UNIQUEKEY
UPD	UPDATE
UPDPW	UPDATEPW
UPG	UPGRADE
USVR	user security verification routine
V	variable unblocked
VARBLK	variable blocked
VARUNB	variable unblocked
VB	variable blocked
VFY	VERIFY
VOL	VOLUME, VOLUMES
volser	volume serial
VTOC	volume table of contents
WCAT	WORKCAT
WCK	WRITECHECK
WFILE	WORKFILE, WORKFILES
WFILES	WORKFILE, WORKFILES
WVOL	WORKVOLUMES
XPRA	EXPORTRA

Appendix L: Glossary

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

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

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

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

alternate index. A collection of index entries organized by the alternate keys of its associated base data records. Its function is to provide alternate means of locating records in a data component on which the alternate index is based.

alternate key. One or more consecutive characters taken from a data record and used to build an alternate index or to locate one or more base data records via an alternate index. (*See also* key, key field, and prime key.)

backup file. A copy that can be used to reconstruct a damaged file.

base cluster. A key-sequenced or entry-sequenced file over which one or more alternate indexes may be built.

catalog. (*See* master catalog, recoverable catalog, and user catalog.)

catalog recovery area. (*See* CRA.)

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

cluster. A named structure consisting of a group of related components (e.g., a data component with its index component). A cluster may consist of a single component. (*See also* base cluster.)

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 the index portion (for a key-sequenced file only), referred to in this manual as data component and index component, of a cluster, an alternate index, or a catalog.

compression. (*See* key compression.)

control area. A group of control intervals used as a unit for formatting a file before adding records to it. Also, in a key-sequenced file, the set of control intervals pointed to by a sequence-set index record; used by VSAM for distributing free space and for placing a sequence-set index record adjacent to its data.

control area split. The movement of the contents of some of the control intervals in a control area to a newly created control area, to facilitate the insertion or lengthening of a data record when there are no remaining free control intervals in the original control area.

control interval. A fixed-length area of auxiliary-storage space in which VSAM stores records and distributes free space. It is the unit of information transmitted to or from auxiliary storage by VSAM, independent of physical record size.

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

control interval split. The movement of some of the stored records in a control interval to a free control interval, to facilitate the insertion or lengthening of a record that won't fit in the original control interval.

count-key-data. (*See* CKD.)

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

data integrity. Preservation of data or programs for their intended purpose. As used in this publication, the safety of data from inadvertent destruction or alteration.

data record. A collection of items of information from the standpoint of its use in an application and not from the standpoint of the manner in which it is stored (*see also* stored record).

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

data space. A storage area defined in the volume table of contents of a direct-access volume for the exclusive use of VSAM to store files, indexes, and catalogs.

direct access. The retrieval or storage of data by a reference to its location in a file 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 file for inserting new records into the file in key sequence; also, whole control intervals reserved in a control area for the same purpose.

entry name. A unique name for each component or object as it is identified in a catalog. This name is the same as the *file-ID* in the associated DLBL job control statement.

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

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

exception. An abnormal condition such as an I/O error encountered in processing a file.

extent. A contiguous space allocated on a direct-access storage volume, reserved for a particular data space or file. (*See also* primary space allocation and secondary space allocation.)

external sort. Sorting of data records into a new sequence using work files obtained when an alternate index is built. (*See also* internal sort.)

FBA. (*See* fixed block architecture.)

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

file. The major unit of data storage and retrieval in the operating system, consisting of data in a prescribed arrangement and described by control information to which the system has access. (*See also* key-sequenced file, entry-sequenced file, and relative-record file.)

fixed block architecture. A direct access storage device that is linear in structure; it contains fixed-length (512 byte) blocks of data. The blocks are numbered from 0 to 'n-1', where 'n' depends on the capacity of the auxiliary storage medium. Each block is addressed by its Relative Block Number. In Fixed Block Architecture there are no keys as in Count-Key-Data DASD. The only way to identify a block is by specifying its Relative Block Number. With FBA you are not aware of the physical structure of the auxiliary storage medium. You don't need to know the capacity of tracks, how many tracks per cylinder, nor how many cylinders per DASD there are. The only thing you have to know is the maximum number of blocks per spindle.

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

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

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

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

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

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

index replication. The use of an entire track of direct-access storage to contain as many copies of a single index record as possible; reduces rotational delay.

index set. The set of index levels above the sequence set. The index set and the sequence set together comprise the index.

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

integrity. (*See* data integrity.)

internal sort. Sorting of data records into a new sequence using virtual storage obtained by a GETVIS. (*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 file with an index.

job catalog. A catalog made available for a job by means of the *filename* IJSYSUC in a DLBL job control statement.

key. As used in this publication, one or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records. (*See also* key field and alternate 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 file, whose content is a key of a record.

key sequence. The collating sequence of data records, determined by the value of the key field in each of the data records. May be the same as, or different from, the entry sequence of the records.

key-sequenced file. A file whose records are loaded in key sequence and controlled by an index.

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

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

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

master catalog. A key-sequenced file with an index containing extensive file and volume information that VSAM requires to locate files, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a file, and to accumulate usage statistics for files.

Max-CA. (*See* Min-CA.)

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

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

objects. Logical entities created by VSAM such as clusters and their respective components, alternate indexes and their respective components, catalogs and their respective components, paths, and volumes (i.e., VSAM data spaces).

password. A unique string of characters stored in a catalog that a program or a computer operator must supply to meet security requirements before a program gains access to a file.

path. A named, logical entity composed of one or more clusters (an alternate index and its base cluster, for example).

physical record. A physical unit of recording on a medium, for example, the physical unit between address markers on a disk.

pointer. An address or other indication of location. For example, an RBA is a pointer that gives the relative location or a data record or a control interval in the file to which it belongs. (*See also* horizontal pointer and vertical pointer.)

portability. The ability to use VSAM files with different operating systems. Volumes whose files 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 files can be transported between operating systems using Access Method Services.

primary space allocation. A contiguous space on a direct access storage device, occupied by or reserved for a particular file. (*See also* secondary space allocation.)

prime index. The index component of a key-sequenced file that has one or more alternate indexes. (*See also* index and alternate index.)

prime key. (*See* key.)

random access. (*See* direct access.)

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

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

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 the prime index, if the base cluster is a key-sequenced file; otherwise, the first volume of the base data.

relative byte address. (*See* RBA.)

relative-record file. A file whose records are loaded into fixed-length slots.

relative-record number. A number that identifies not only the slot, or data space, in a relative-record file but also the record occupying the slot.

replication. (*See* index replication.)

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

secondary space allocation. A contiguous space on a direct access storage device, occupied by or reserved for a particular file, 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 file; it gives the locations of the control intervals in the file 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.)

skip sequential access. Keyed sequential retrieval or storage of records here and there throughout a file, skipping automatically to the desired record or collating position for insertion: VSAM scans the sequence set to find a record or a collating position.

sort. (*See* external sort and internal sort. *See also* collating sequence.)

spanned record. A logical record whose length exceeds control interval length, and as a result, crosses, or spans, one or more control interval boundaries within a single control area.

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

upgrade set. All the alternate indexes that VSAM has been instructed to update whenever there is a change to the data component of the base cluster.

user catalog. 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.

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 file controlled by the index.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Using VSE/VSAM Commands and Macros (File No. S370-30) Printed in U.S.A. SC24-5144-1

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

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

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

Using VSE/VSAM
Commands and Macros
SC24-5144-1

READER'S COMMENT FORM

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

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

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

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

• What is your occupation? _____

• How do you use this publication:

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

Your comments:

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

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

Reader's Comment Form

Cut or Fold Along Line

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

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

Fold | Fold

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

Your Name _____

Company Name _____ *Department* _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM Branch Office serving you _____

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

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

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

Using VSE/VSAM Commands and Macros (File No. S370-30) Printed in U.S.A. SC24-5144-1

Using VSE/VSAM
Commands and Macros
SC24-5144-1

READER'S COMMENT FORM

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

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

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

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

• What is your occupation? _____

• How do you use this publication:

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

Your comments:

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

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

Reader's Comment Form

Cut or Fold Along Line

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

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

Fold | Fold

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

Your Name _____

Company Name _____ *Department* _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM Branch Office serving you _____

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

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

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

Using VSE/VSAM Commands and Macros (File No. S370-30) Printed in U.S.A. SC24-5144-1