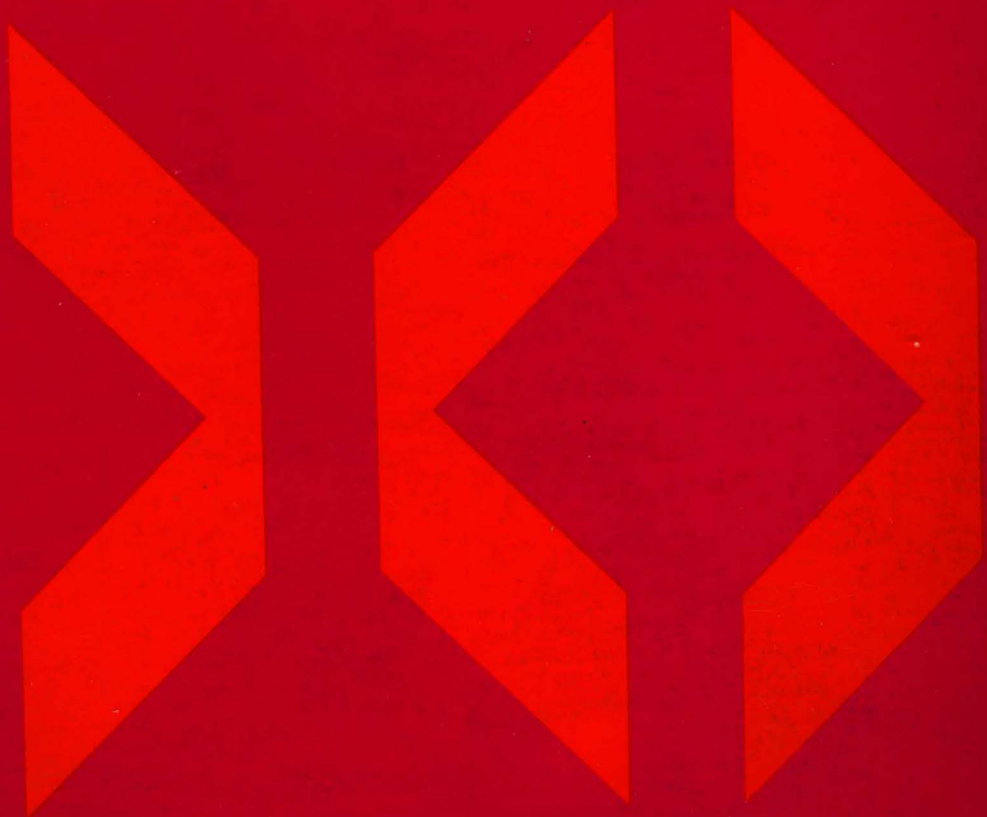
IBM

MVS/Extended Architecture Catalog Administration Guide

Licensed Program



Order Number
GC26-4138-4

Data Facility Product
5665-XA2

Version 2
Release 3.0

MVS/Extended Architecture Catalog Administration Guide

Licensed Program

Order Number
GC26-4138-4

Data Facility Product
5665-XA2

Version 2
Release 3.0

| **Fifth Edition (December 1987)**

| This is a major revision of, and makes obsolete, GC26-4138-3.

This edition applies to Version 2 Release 3.0 of MVS/Extended Architecture Data Facility Product, Licensed Program 5665-XA2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent publication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300, and 9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's program may be used. Any functionally equivalent program may be used instead.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

A Reader's Comment Form is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publishing, P. O. Box 49023, San Jose, California, U.S.A. 95161-9023. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

This publication describes how to use the integrated catalog facility. It is intended for system programmers, data administrators, and other personnel who build, maintain, and support integrated catalog facility catalog structures. Information on VSAM catalogs and OS CVOLs can be found in Appendixes E, F, and G.

Organization

This publication has 16 sections:

- Chapter 1, "Introduction," gives an overview of integrated catalog facility catalogs, and discusses the advantages of using them.
- Chapter 2, "Integrated Catalog Facility Catalog Structure," provides basic information about the integrated catalog facility catalog structure.
- Chapter 3, "Defining, Altering, and Deleting an Integrated Catalog Facility Catalog," describes how the access methods services commands are used to create, alter, move, delete, copy, split, merge, and modify the integrated catalog facility catalogs.
- Chapter 4, "Converting VSAM Catalogs or OS CVOLs to Integrated Catalog Facility Catalogs," describes the various conversion processes and planning requirements needed for each type of catalog conversion and provides examples of their use.
- Chapter 5, "Backing Up and Recovering Integrated Catalog Facility Catalogs," discusses how to develop adequate backup and recovery procedures in an integrated catalog facility environment.
- Chapter 6, "Checking Catalogs for Errors and Synchronization," describes the DIAGNOSE command and explains how to use the DIAGNOSE output to determine what action to take to recover the integrated catalog facility catalog.
- Chapter 7, "Communicating with Catalog Address Space (CAS)," describes the CATALOG options of the system command MODIFY, and provides some examples of their use.
- Appendix A, "The Integrated Catalog Facility Catalog Cell Structure," lists the types of cells in the integrated catalog facility catalog.
- Appendix B, "Conversion from VSAM to an Integrated Catalog Facility Catalog," provides an example of a job stream that manipulates integrated catalog facility catalogs.
- Appendix C, "Alternate Master Catalog Job Stream," provides an example of a job stream used to create an alternate master catalog.
- Appendix D, "Operand Notation for SHOWCAT," describes the SHOWCAT macro and its standard, list, and execute forms.
- Appendix E, "VSAM Catalogs," describes VSAM catalogs for those installations that have VSAM catalogs that are not converted to integrated catalog facility catalogs.
- Appendix F, "CVOL Processor," describes OS CVOLs for those installations that have CVOLs that are not converted to integrated catalog facility catalogs.

- Appendix G, "Using Catalog Management Macro Instructions for OS CVOLs," discusses in detail the various uses of catalog management macro instructions.
- Appendix H, "Region Requirements for Access Method Services Jobs," explains how to calculate the virtual storage requirements for a VSAM job.
- "Glossary of Terms and Abbreviations" lists and defines the terms used in this book.

Prerequisite Knowledge

To use this book efficiently, you should already be familiar with VSAM and the catalog environment.

Required Publications

You should be familiar with the information presented in the following publications:

- *MVS/Extended Architecture VSAM Administration Guide*, GC26-4151, provides information on VSAM optimization options and various VSAM algorithms that affect performance.
- *MVS/Extended Architecture Data Facility Product Version 2: Planning Guide*, GC26-4147, describes planning considerations for integrated catalog facility catalogs and VSAM.

Related Publications

Within the text, references are made to the publications listed in the table below:

Short Title	New Title	Order Number
Access Method Services Reference	*MVS/Extended Architecture Integrated Catalog Administration: Access Method Services Reference*	GC26-4135
	MVS/Extended Architecture VSAM Catalog Administration: Access Method Services Reference	GC26-4136
Catalog Diagnosis Guide	*MVS/Extended Architecture Catalog Diagnosis Guide*	LY26-3955
Catalog Diagnosis Reference	*MVS/Extended Architecture Catalog Diagnosis Reference*	LY26-3956
Checkpoint/Restart User's Guide	*MVS/Extended Architecture Checkpoint/Restart User's Guide*	GC26-4139

Short Title	New Title	Order Number
DADSM and CVAF Diagnosis Guide	*MVS/Extended Architecture DADSM and Common VTOC Access Facility Diagnosis Guide*	LY26-3960
DADSM Diagnosis Reference	*MVS/Extended Architecture DADSM Diagnosis Reference*	LY26-3961
Data Administration Guide	*MVS/Extended Architecture Data Administration Guide*	GC26-4140
Data Facility Product: Planning Guide	*MVS/Extended Architecture Data Facility Product Version 2: Planning Guide*	GC26-4147
Debugging Handbook	*MVS/Extended Architecture Debugging Handbook*, Volumes 1 through 5	LC28-1164 LC28-1165 LC28-1166 LC28-1167 LC28-1168 To order all five, use LBOF-1015.
JCL Reference	*MVS/Extended Architecture JCL Reference*	GC28-1352
JCL User's Guide	*MVS/Extended Architecture JCL User's Guide*	GC28-1351
RACF General Information Manual	*Resource Access Control Facility (RACF): General Information Manual*	GC28-0722
Service Aids	*MVS/Extended Architecture System Programming Library: Service Aids*	GC28-1159
Supervisor Services and Macro Instructions	*MVS/Extended Architecture System Programming Library: Supervisor Services and Macro Instructions*	GC28-1154
System Commands	*MVS/Extended Architecture Operations: System Commands*	GC28-1206
System – Data Administration	*MVS/Extended Architecture System – Data Administration*	GC26-4149
System Generation	*MVS/Extended Architecture Installation: System Generation*	GC26-4148

Short Title	New Title	Order Number
System Macros and Facilities	*MVS/Extended Architecture System Programming Library: System Macros and Facilities*, Volumes 1 and 2	GC28-1150 GC28-1151
System Management Facilities	*MVS/Extended Architecture System Programming Library: System Management Facilities*	GC28-1153
System Messages	*MVS/Extended Architecture Message Library: System Messages*, Volumes 1 and 2	GC28-1376 GC28-1377
System Modifications	*MVS/Extended Architecture System Programming Library: System Modifications*	GC28-1152
Utilities	*MVS/Extended Architecture Data Administration: Utilities*	GC26-4150
VSAM Administration Guide	*MVS/Extended Architecture VSAM Administration Guide*	GC26-4151
VSAM Administration: Macro Instruction Reference	*MVS/Extended Architecture VSAM Administration: Macro Instruction Reference*	GC26-4152
VSAM Logic	*MVS/Extended Architecture VSAM Logic*	LY26-3970

Notational Conventions

A uniform system of notation describes the format of access method services commands. This notation is not part of the language; it merely provides a basis for describing the structure of the commands.

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

- Brackets [ ] indicate optional parameters.
- Braces { } indicate a choice of entry; unless a default is indicated, you must choose one of the entries.
- Items separated by a vertical bar (|) represent alternative items. No more than one of the items may be selected.
- An ellipsis (...) indicates that multiple entries of the type immediately preceding the ellipsis are allowed.
- Other punctuation (parentheses, commas, etc.) must be entered as shown.
- **BOLDFACE** type indicates the exact characters to be entered. Such items must be entered exactly as illustrated (in uppercase, except in TSO).
- *Lowercase italic* type specifies fields to be supplied by the user.

- **BOLDFACE UNDERSCORED** type indicates a default option. If the parameter is omitted, the underscored boldface value is assumed.
- A ' ' in the command format indicates that a blank (an empty space) must be present before the next parameter.

Summary of Changes

Release 3.0 Update, December 1987

Figure 15 in Chapter 2 has been updated to show the new fields for expiration century and creation century needed for data set expiration dates beyond 1999.

"Protecting the Catalog" in Chapter 3 and various sections in Chapter 5, "Backing Up and Recovering Catalogs," have been updated to support enhanced integrated catalog facility recovery.

Sections of Chapter 5, "Backing Up and Recovering Catalogs," have also been updated to reflect changes due to the new capability of automatically exporting aliases and being able to record information about exports of integrated catalog facility catalogs.

A new section, "Recovering Shared Catalogs," has been added to Chapter 5, "Backing Up and Recovering Catalogs." It describes how to use the IMPORT CONNECT ALIAS command to recover shared catalogs.

Chapter 7, "Communicating with Catalog Address Space (CAS)," has been added to describe the CATALOG options of the system command MODIFY.

Information has been added to reflect the new linear data set (LDS) and service changes to the previous edition.

Release 2.0, June 1986

"Catalog Control Interval and Control Area Size" in Chapter 2 has been updated to support the increase in block sizes allowed.

"Resource Access Control Facility (RACF)" and "Deleting Catalogs and Indexes" in Chapter 3 have been updated for ERASE-on-SCRATCH support.

Information has been added to reflect service changes to the previous edition.

Release 1.0 Update, December 1985

Chapter 4, "Converting to Integrated Catalog Facility Catalogs," has been rewritten to describe the various conversion processes and planning requirements needed for each type of catalog conversion and provides examples of their use.

The section on the DIAGNOSE command in Chapter 6, "Checking Catalogs for Errors and Synchronization," has been rewritten to clarify the meaning of DIAGNOSE reason codes and the best use of catalog recovery procedures.

Appendix D, "Operand Notation for SHOWCAT," has been rewritten to include a complete description of the three forms of SHOWCAT: standard, list, and execute.

Information has been added to reflect any service changes.

Release 1.0, April 1985

Enhancements and New Support

The VVR data set information cell and VVR volume information cell formats in Figure 24, "Examples of the VVR Cell Information," have been updated.

The DIAGNOSE "Execution Error Messages" and "Summary Messages" in Chapter 6 have been updated.

"VSAM Volume Record (VVR)" in Chapter 2, "Estimating Space Requirements for the VVDS" in Chapter 3, and "VSAM Volume Data Set (VVDS) Cells" in Appendix A have been updated to include information on how to determine the size of a VVR.

An overview of the catalog address space (CAS) has been added to Chapter 1, "Introduction."

The 3380 Models AD4, BD4, AE4, and BE4 device types have been included in estimating space requirements for the BCS.

The conversion examples of a VSAM catalog and an OS CVOL have been updated for 3380 support.

Appendix C, "Alternate Master Catalog Job Stream," has been updated for 3380 support.

Version 2 Publications

The preface includes the new order numbers for Version 2.

Contents

Chapter 1. Introduction 1
Highlights of Integrated Catalog Facility Catalogs 1
Advantages of the Integrated Catalog Facility Catalog 1
Performance 1
Capability 1
Usability 2
Maintainability 2
Catalog Address Space (CAS) 3

Chapter 2. Integrated Catalog Facility Catalog Structure 5
Basic Catalog Structure (BCS) 5
Records 6
Record Size and Extension Records for Catalogs 14
VSAM Volume Data Set (VVDS) 15
VSAM Volume Control Record (VVCR) 16
VVDS Self-Describing VVR 16
VSAM Volume Record (VVR) 16
Assigning Space to an Integrated Catalog Facility Catalog 18
Catalog Control Interval and Control Area Size 19
Volume Table of Contents (VTOC) Support 19

Chapter 3. Defining, Altering, and Deleting a Catalog 21
Using Access Method Services 21
Using the DEFINE Command 22
Using the Parameters for the DEFINE Command 23
Defining a BCS (DEFINE USERCATALOG) 25
Defining a VVDS (DEFINE VVDS NORECATALOG) 26
Using an Alias to Identify a User Catalog 27
Estimating Space 27
Estimating the Catalog's Space Requirements 27
Estimating Space Requirements for the BCS 27
Estimating Space Requirements for the VVDS 30
Protecting the Catalog 30
Authorized Program Facility (APF) 31
Access Method Services Password Protection 32
Passwords to Authorize Access 32
Password Protection Considerations and Precautions 34
Resource Access Control Facility (RACF) 35
User-Security-Verification Routine (USVR) 37
IGG.CATLOCK Facility 37
Using One Catalog As a Model for Another Catalog 39
Selecting Index Options 40
Index-Set Records in Virtual Storage 40
Size of the Index-Set Control Interval 41
Index and Data on Separate Volumes 41
Replication of Index Records 41
Sequence-Set Records Adjacent to Control Areas 41
Index Option Summary 42
Using JCL and Dynamic Allocation (DEFINE) 42
Identifying the Catalog's Volume 42
Using Share Options 43

Using the Alternate Master Catalog 43
Altering Attributes of Entries in a Catalog 44
Generic Names and ALTER 45
Renaming Generically Named Entries 46
Deleting Catalogs and Indexes 46
DELETE ALTERNATEINDEX 47
DELETE ERASE|NOERASE 47
DELETE FORCE|NOFORCE 47
DELETE PURGE|NOPURGE 47
DELETE RECOVERY|NORECOVERY 48
DELETE SCRATCH|NOSCRATCH 48
DELETE TRUENAME|ALIAS 48
FILE Parameter 49
Examples 49
Example 1: Define a User Catalog 49
Example 2: Define a User Catalog With a Model 50
Example 3: Delete a Catalog and Its Cataloged Objects 52

Chapter 4. Converting to Integrated Catalog Facility Catalogs 55
Converting VSAM Catalogs 55
Choosing a Conversion Technique 55
Planning for Conversion Using CNVTCAT 56
Converting the Catalog 61
Verifying the Conversion 61
Rerunning a Failed CNVTCAT Conversion Process 62
Reorganizing the Catalog after Conversion 62
Converting the Master Catalog 62
Converting OS CVOLS 63
Converting Application Programs 64
Revising Installation Procedures 64
Sample Procedures for Catalog Conversion 66
Converting a Recoverable Catalog 67
Converting a Nonrecoverable Catalog 68
Converting an OS/CVOL 69
Removing an Unavailable Volume from a Nonrecoverable Catalog 70
Removing an Unavailable Volume from a Recoverable Catalog 70
Converting a Master Catalog 72

Chapter 5. Backing Up and Recovering Catalogs 73
A Backup and Recovery Strategy 73
Backing Up a Catalog 74
Backing Up the BCS Using EXPORT 75
Backing Up the BCS Using REPRO 75
Recovering a Catalog 76
Recovering the BCS Using IMPORT 76
Recovering Shared Catalogs 78
Recovering the BCS Using REPRO 79
Moving a Catalog 81
Recording EXPORT Information 81
Problem Prevention 81
Early Indication of Errors 81
Monitoring Usage 82
BCS Reorganization Using EXPORT/IMPORT 82
BCS Reorganization Using REPRO 83
VVDS Reorganization 84

Selecting a Solution to a Problem . . . 85
ALTER REMOVEVOLUMES . . . 86
DEFINE Data Set RECATALOG . . . 88
DEFINE VVDS RECATALOG . . . 89
DEFINE ALTERNATEINDEX RECATALOG . . . 89
DEFINE Cluster RECATALOG . . . 89
DEFINE Path RECATALOG . . . 90
DELETE TRUENAME . . . 90
DELETE VVR . . . 90
DELETE Data Set NOSCRATCH . . . 91
DELETE USERCATALOG FORCE . . . 92
DELETE USERCATALOG RECOVERY . . . 92
DELETE VVDS NOSCRATCH . . . 93
DELETE VVDS RECOVERY . . . 93
EXPORT/IMPORT in Backup and Recovery . . . 94
Interpreting Messages . . . 95
Other Backup and Recovery Facilities . . . 96
Data Facility Data Set Services (DFDSS) . . . 96
Recovery Procedure Scenarios . . . 97
Recover VVDS with Connection Only to BCS on the Same Volume . . . 101
Recover VVDS When BCS Is on Volume but VVDS Is Known to Another BCS . . . 104
Recover VVDS When BCS Is Not on the Volume . . . 110
Examples of Printing and Deleting . . . 112

Chapter 6. Checking Catalogs for Errors and Synchronization . . . 115
Listing Catalog Information . . . 115
Using the DIAGNOSE Command . . . 115
INCLUDE/EXCLUDE Parameters . . . 115
COMPARE Parameter . . . 116
DIAGNOSE Output . . . 117
Storage Estimate for DIAGNOSE . . . 117
Messages from DIAGNOSE . . . 118
Interpreting DIAGNOSE Output . . . 119
Catalog Recovery Procedures . . . 133
Sample DIAGNOSE Output . . . 134

Chapter 7. Communicating with Catalog Address Space (CAS) . . . 139
MODIFY CATALOG . . . 139
MODIFY CATALOG Command Formats and Examples . . . 140
MODIFY CATALOG,ABEND . . . 140
MODIFY CATALOG,CLOSE . . . 140
MODIFY CATALOG,END . . . 141
MODIFY CATALOG,ENTRY . . . 142
MODIFY CATALOG,LIST . . . 142
MODIFY CATALOG,OPEN . . . 142
MODIFY CATALOG,REPORT . . . 143
MODIFY CATALOG,RESTART . . . 143
MODIFY CATALOG,VCLOSE . . . 144
MODIFY CATALOG Messages . . . 144
MODIFY CATALOG Abend Codes . . . 147

Appendix A. The Integrated Catalog Facility Catalog Cell Structure . . . 149
Basic Catalog Structure (BCS) Cells . . . 149
VSAM Volume Data Set (VVDS) Cells . . . 151

Appendix B. Sample Conversion From VSAM to Integrated Catalog Facility Catalog 155

Appendix C. Alternate Master Catalog Job Stream 157

Appendix D. Operand Notation for SHOWCAT 167
The SHOWCAT Macro 167
Using the SHOWCAT Macro 167
Standard Form of SHOWCAT 168
List Form of SHOWCAT 172
Execute Form of SHOWCAT 172
Expressions That Can Be Used for SHOWCAT 173
Return Codes from SHOWCAT 174

Appendix E. VSAM Catalogs 175
The Data Component 175
The Index Component 175
Size of a Control Area and Location of the Sequence Set 176
A VSAM Catalog's Use in Data and Space Management 176
Information Contained in the Records of a Catalog 177
Information in a Data Set Record 177
Information in a Volume Record 177
Allocation of Catalog Space 178
Utilization of Catalog Space 178
VSAM Catalogs and Volume Ownership 178
Using the DEFINE Command for VSAM Catalogs 180
Creating a VSAM Catalog 181
Defining a VSAM Recoverable Catalog 182
Allocating the Volume 182
VSAM Objects in a Data Space 182
How Space Is Assigned to a VSAM Catalog 183
Estimating the VSAM Catalog's Space Requirements 185
Index Options 188
Index-Set Records in Virtual Storage 189
Size of the Index-Set Control Interval 189
Index and Data on Separate Volumes 189
Replication of Index Records 189
Sequence-Set Index Records Adjacent to Control Areas 190
Index Option Summary 190
Copying a VSAM Catalog 190
Copy Catalog Preparation for a VSAM Catalog 191
Copy Catalog Procedure for a VSAM Catalog 193
Backing Up a VSAM Catalog for Recovery 194
Backing Up the VSAM Master Catalog 194
Dumping a VSAM Catalog and Its Data Sets 195
Unloading a VSAM Catalog 196
Reloading a VSAM Catalog 196
Optimizing the Performance of Unload/Reload 198
Changing the Volume Serial Number 199
Altering Attributes of Entries in a VSAM Catalog 200
Generic Names and ALTER 200
Renaming Generically Named Entries 201
VSAM Catalog Performance 202
Sharing Services with User Catalogs 202
Improving Catalog Performance 202

Performance Measurement 202
VSAM Catalog Backup and Recovery 203
Catalog Unload and Reload 203
Regaining Access to Data 207
VSAM Volume Recovery Function 215
Recovering a VSAM Recoverable Catalog 216
Restoring the Catalog Entry Obtained Using the EXPORTRA Command 221
JCL Requirements 225
Restarting Programs after a Failure 226
Diagnostic Aids 227
Changing a User Catalog to a VSAM Master Catalog 228

Appendix F. CVOL Processor 233
Purpose of the CVOL Processor 233
Functions Performed 233
Functions Not Performed 236
Using the CVOL Processor 236
How CVOLs Are Accessed 236
CVOL Pointers in the Master Catalog 237
CVOL Connector Names in the Master Catalog 237
Resource Access Control Facility (RACF) 237
Restrictions and Limitations 238

Appendix G. Using Catalog Management Macro Instructions for OS CVOLs 245
Catalog Order of Search 245
Return Code Considerations 246
Retrieving Information from a Catalog 246
Retrieving Information by Data Set Name (LOCATE and CAMLST NAME) 247
Retrieving Information by Generation Data Set Name (LOCATE and CAMLST NAME) 250
Retrieving Information by Alias (LOCATE and CAMLST NAME) 251
Reading a Block by Relative Block Address (LOCATE and CAMLST BLOCK) 253
Building and Deleting Indexes 254
Building an Index (INDEX and CAMLST BLDX) 254
Building a Generation Index (INDEX and CAMLST BLDG) 256
Deleting an Index (INDEX and CAMLST DLTX) 258
Assigning an Alias for an Index (INDEX and CAMLST BLDA) 259
Deleting an Alias for an Index (INDEX and CAMLST DLTA) 260
Connecting and Disconnecting OS CVOLs 261
Connecting OS CVOLs (INDEX and CAMLST LNKX) 261
Disconnecting OS CVOLs (INDEX and CAMLST DRPX) 263
Working with Non-VSAM Data Set Catalog Entries 264
Cataloging a Non-VSAM Data Set (CATALOG and CAMLST CAT) . . 264
Uncataloging a Non-VSAM Data Set (CATALOG and CAMLST UNCAT) 267
Recataloging a Non-VSAM Data Set (CATALOG and CAMLST RECAT) 268
OS CVOL Entry Formats 270
OS CVOL Volume Index Control Entry 270
OS CVOL Index Control Entry 271
OS CVOL Index Link Entry and Index Pointer Entry 272
OS CVOL Data Set Pointer Entry 273
OS CVOL Volume Control Block Pointer Entry 274
Volume Control Block 275

OS CVOL Pointer Entry 276
OS CVOL Pointer Entry (OLD) 276
OS CVOL Generation Index Pointer Entry 277
OS CVOL Alias Name 278

Appendix H. Region Requirements for Access Method Services Jobs 279

Glossary of Terms and Abbreviations 283

Index 287

Figures

1. Relationship of the BCS and the VVDS . . . 5
2. Sphere Record for a Key-Sequenced Data Set and an Alternate Index . . . 7
3. Generation Data Group Sphere Record . . . 7
4. Nonsphere Records . . . 8
5. Alias Name Cell Format . . . 9
6. Alternate Index Name Cell Format . . . 9
7. Association Cell Format . . . 9
8. Cluster Name Cell Format . . . 10
9. Data Name Cell or Index Name Cell Format . . . 10
10. Generation Aging Table Cell Format . . . 10
11. Generation Data Group Name Cell . . . 11
12. Generation Data Set Name Cell . . . 11
13. Integrated Catalog Facility Connector Name Cell Format . . . 11
14. Non-VSAM Name Cell . . . 11
15. Ownership Cell . . . 12
16. Path Name Cell . . . 12
17. Relation Cell Format . . . 12
18. Security Cell Format . . . 13
19. True Name Cell Format . . . 13
20. Volume Cell Format . . . 13
21. Extension Record . . . 14
22. VSAM Volume Data Set (VVDS) Structure . . . 15
23. VSAM Volume Record (VVR) . . . 17
24. Examples of VVR Cell Information . . . 18
25. Activities That Downgrade the Integrated Catalog Facility Catalog . . . 80
26. Backup and Recovery Solutions . . . 85
27. Using INCLUDE and EXCLUDE as DIAGNOSE Parameters . . . 116
28. DIAGNOSE Messages . . . 119
29. Example of an Association and Its Logical Connections . . . 126
30. Sample DIAGNOSE Output . . . 135
31. Interrelationship Among Catalog Entries. . . . 168
32. Operand Expressions for the SHOWCAT Macro . . . 173
33. Worksheet for Estimating a VSAM Catalog's Space Requirements . . . 186
34. Catalog Volume Records . . . 208
35. VSAM Object Records . . . 208
36. Catalog Recovery Area Contents . . . 216
37. Sample Job Stream for a Recoverable VSAM Master Catalog . . . 230
38. OS and VSAM Functions Supported . . . 233
39. TSO Command Mapping . . . 235
40. Catalog Functions Not Supported . . . 236
41. RACF Authorization Checking . . . 238
42. Return Codes and Their Meanings . . . 240
43. OS CVOL Volume Index Control Entry . . . 270
44. OS CVOL Index Control Entry . . . 271
45. OS CVOL Index Link and Index Pointer Entries . . . 272
46. OS CVOL Data Set Pointer Entry . . . 273
46. OS CVOL Data Set Pointer Entry . . . 273
47. OS CVOL Volume Control Block Pointer Entry . . . 274
48. OS CVOL Volume Control Block . . . 275
49. OS CVOL Pointer Entry . . . 276
50. OS CVOL Generation Index Pointer Entry . . . 277

51. OS CVOL Alias Name 278

Chapter 1. Introduction

To help you use integrated catalog facility catalogs for controlling data sets, this publication explains how to use the following:

- Access method services commands to define, recover, back up, list, and copy integrated catalog facility catalogs
- Appropriate macros and job control language (JCL)

(For information on VSAM catalogs and OS CVOLs, see Appendixes E, F, and G.)

Highlights of Integrated Catalog Facility Catalogs

The integrated catalog facility catalog functionally replaces OS control volumes (CVOLs) and VSAM catalogs. It has two parts:

- The **basic catalog structure (BCS)** contains volume, security, ownership, and association information for VSAM data sets and the integrated catalog facility catalog. The BCS also contains volume, ownership, and association information for non-VSAM data sets.
- The **VSAM volume data set (VVDS)** contains the data set characteristics and the volume-related information of the VSAM data sets cataloged in the integrated catalog facility catalog. The VVDS physically resides on the same volume as the VSAM data sets. It also contains the data set characteristics of any integrated catalog facility catalog found on the volume.

The BCS and VVDS may reside on the same volume or on separate volumes.

Advantages of the Integrated Catalog Facility Catalog

The integrated catalog facility catalog offers significant advantages over OS CVOLs and VSAM catalogs. Although OS CVOLs and VSAM catalogs are still supported in the MVS/XA environment, integrated catalog facility catalogs give you superior performance, capability, usability, and maintainability.

Performance

Integrated catalog facility catalogs can be updated faster. The catalog information that requires the most frequent updates is physically located in the VVDS on the same volume as the data sets, allowing faster access.

Furthermore, a catalog request is expedited because fewer I/O operations are needed. Related entries, such as a cluster and its alternate index, are processed together.

Capability

Each volume can have entries that are cataloged in as many as 36 integrated catalog facility catalogs.

The VSAM catalog concept of catalog ownership of a volume does not apply to integrated catalog facility catalogs. An integrated catalog facility catalog and its BCS

can have data sets cataloged on any number of volumes. The BCS can have as many as 123 extents on one volume. One volume can have multiple integrated catalog facility catalogs on it. All the necessary control information is recorded in the VVDS residing on that volume.

Usability

When defining an integrated catalog facility catalog, you have more control because you can specify parameters that cannot be specified in a VSAM catalog. With the commands provided, you can reorganize catalogs, move catalogs to different device types, merge two catalogs into one, split one catalog into two or more catalogs, share catalogs, and create portable copies.

All direct access storage device space management functions are performed by direct access device storage management (DADSM). Data sets cataloged in an integrated catalog facility catalog are similar to VSAM UNIQUE data sets; therefore, no VSAM data spaces are necessary.

Significant space savings for generation data groups are achieved in the integrated catalog facility catalog by reusing space when an old generation is deleted and by using an improved method of recording generation data groups.

For multivolume data sets defined in integrated catalog facility catalogs, OPEN requires that all primary volumes be parallel mounted. Subset mounting is eliminated.

Maintainability

Maintainability is improved by simpler backup and recovery procedures, and use of the DIAGNOSE command.

Simpler Backup and Recovery Procedures

The BCS and VSAM data sets can be restored independently.

The dynamic information associated with the VSAM data set (the data set characteristics) resides in the VVDS on the same volume as the VSAM data set itself. The VVDS contains the data set characteristics that must be synchronized with the data set each time it is updated. Therefore, you can copy the volume periodically for backup and recovery without causing the data set and VVDS portion of the integrated catalog facility catalog to become out of synchronization.

Information that can be out of synchronization in the catalog (for example, a frequently used relative byte address on volume and extents) is moved from the catalog to the VVDS. The VVDS resides on the same volume as the data set component. Therefore, the BCS data set(s) and data volume(s) can be backed up independently.

The BCS maintains a record of the associated data sets through the use of a sphere record. For example, the sphere record contains a record of a base cluster and its related alternate indexes. All the objects associated with the sphere record are processed before the sphere record is updated. You can therefore restart most processing at the point of interruption, without losing data or special processing.

Error Diagnosis

You can use the access method services DIAGNOSE command to compare the BCS and/or the VVDS and thus verify catalog integrity. If an error is found, DIAGNOSE reveals exactly what the problem is. Based on the result of the DIAGNOSE output, you can determine how to correct the error.

In addition to the DIAGNOSE command, you can use all the existing access method services and catalog diagnostic aids for the integrated catalog facility catalog, except those related to the catalog recovery area (CRA).

Successful execution of a DIAGNOSE command depends on whether the input and compare data sets can be accessed. If you cannot read records from one of these data sets, the DIAGNOSE command terminates with an appropriate message and return code.

Catalog Address Space (CAS)

To reduce virtual storage requirements and increase the number of catalogs that can be used, system address space for the MVS/Extended Architecture catalog function has been established. This address space is the catalog address space, or CAS. In a CAS environment, most of the catalog modules and control blocks are moved from the CSA and PLPA to the CAS private area. Assuming 30 open catalogs, this frees approximately 1 megabyte of virtual storage below 16 megabytes for additional user programs.

Any program that is executed in a non-CAS address space cannot function correctly if it references control blocks and data structures that have been moved to the CAS address space.

Chapter 2. Integrated Catalog Facility Catalog Structure

The integrated catalog facility catalog is composed of one basic catalog structure (BCS) and at least one VSAM volume data set (VVDS). Figure 1 illustrates a catalog and the relationship of the BCS and the VVDS of that catalog and other volumes. The VVDS contains two types of records—one VSAM volume control record (VVCR), the first record in the VVDS, and one or more VSAM volume records (VVRs). The integrated catalog facility catalog uses the VVDS to contain the information about the VSAM data sets residing on the volume with the VVDS. The extent information is contained in the VVRs within the VVDS. The parts of the catalog environment, including the BCS, VVDS, VVCR, and the VVR, are described in greater detail in the following sections of this chapter.

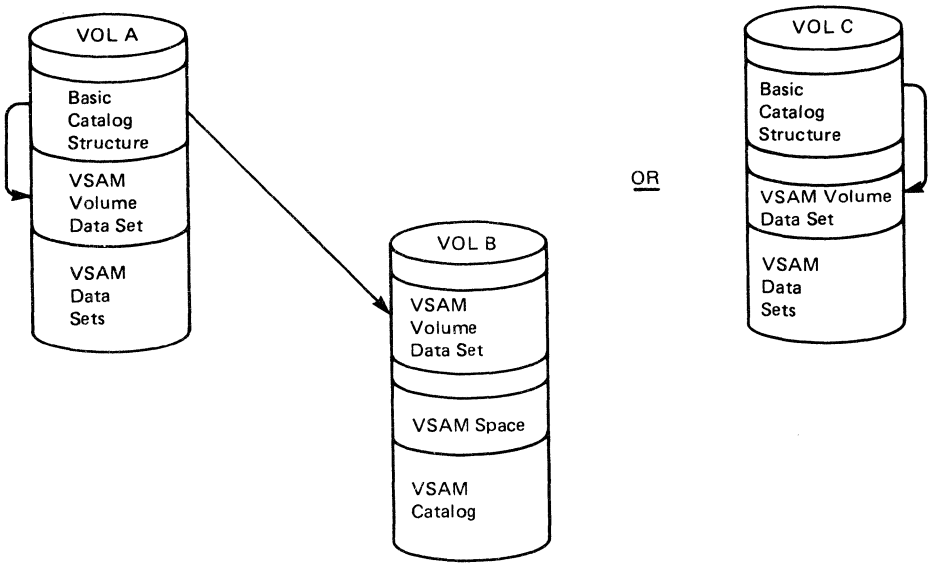


Figure 1. Relationship of the BCS and the VVDS

Basic Catalog Structure (BCS)

The BCS is a key-sequenced data set (KSDS) and contains volume, data set security, ownership, and association information for VSAM and non-VSAM data sets.

Volume ownership restrictions do not apply to the BCS. VSAM data sets residing on one volume may be cataloged in as many as 36 different BCSs.

A BCS can also point to a volume on which only non-VSAM data sets and generation data group data sets reside. The non-VSAM and generation data group set information required by the integrated catalog facility catalog is contained in the BCS itself, and the VVDS does not contain the non-VSAM information.

Attributes that may be defined for the BCS include:

- The control interval size for the data and index component
- The amount of free space that is to be left free after the BCS is loaded or after a control interval or control area split.

Most utility functions and tuning specifications available with VSAM key-sequenced data sets are also available for the BCS.

Related information in the BCS is grouped into logical, variable-length, spanned records related by key. The BCS uses keys that are the data set names (plus 1 character for extensions). A control interval (CI) may contain multiple BCS records. To reduce the number of I/Os necessary for catalog processing, logically related data is consolidated in the BCS.

The BCS cell is the smallest block of information and may contain the name, volume, owner, and association information.

Records

Cells, components, and records are the building blocks of the BCS. A group of logically related cells, physically adjacent in a sphere record, make up a component. An example of a component is a data component, a cluster component, or a generation data group (GDG) component.

There are two types of records: the sphere record and the nonsphere record. A sphere record contains one or more components. Two examples of sphere records are the VSAM sphere record shown in Figure 2 on page 7, and the GDG sphere record shown in Figure 3 on page 7. The nonsphere records are the non-VSAM, alias, connector, path, and truename records. Examples of these records are shown in Figure 4 on page 8. The key length of any record is 45 bytes, consisting of a 44-byte user-supplied name and a 1-byte pad character to indicate an extension record.

When the integrated catalog facility catalog is defined on a volume, a VSAM sphere record is built for the VVDS data set with a name of SYS1.VVDS.Vvolser. This record is built because a VVR flag is turned on, indicating the first occurrence of this catalog in the VVDS.

The integrated catalog facility catalog has a sphere record similar to that of the other VSAM key-sequenced data sets. This is a self-describing sphere record for an integrated catalog facility catalog. The key of this sphere record is binary zeros to ensure it is the first record in the catalog. The data component name is the user-defined catalog name and matches the name on the data component's Format-1 DSCB. A true name record is created for the data and index components. The true name records are related with a key to the user-specified catalog name. A record is created to relate the index component to the catalog name of binary zeros—in the same way as for the data component.

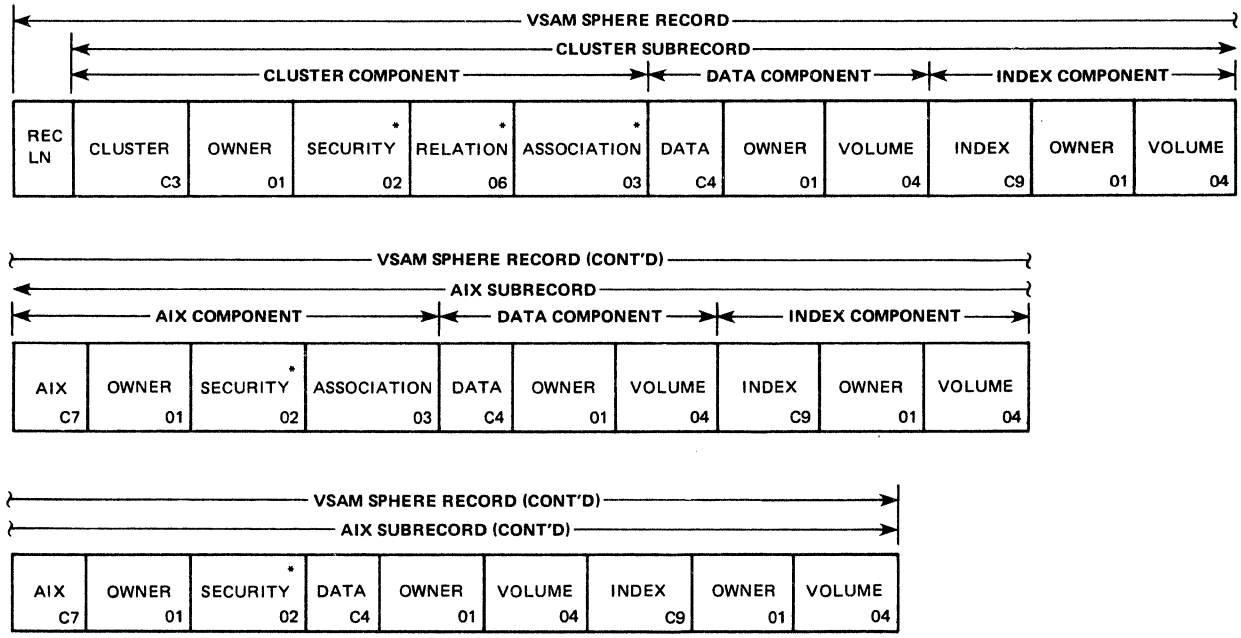


Figure 2. Sphere Record for a Key-Sequenced Data Set and an Alternate Index

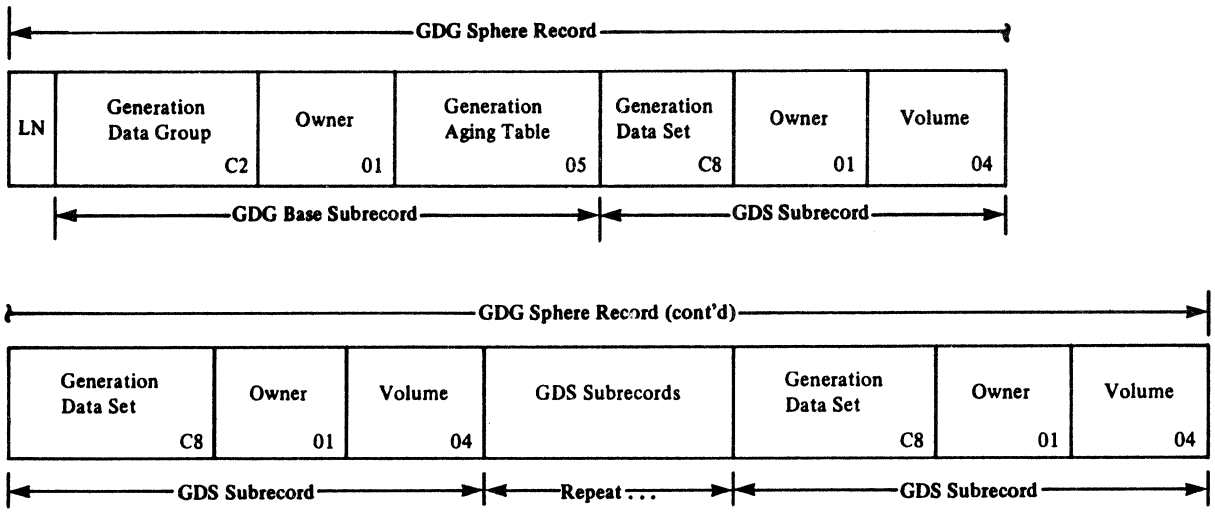


Figure 3. Generation Data Group Sphere Record

Example of a non-VSAM record:

LL	Name Cell	Owner Cell	* Association Cell	Volume Cell	* Volume Cell
	C1	01	03	04	04

Example of an Alias record:

LL	Name Cell	Association Cell
	E7	03

Example of a user Catalog Connector record:

LL	Name Cell	Owner Cell	* Association Cell	Volume Cell
	E4	01	03	04

Example of a Path record:

LL	Name Cell	Owner Cell	* Security Cell	Association Cell
	D9	01	02	03

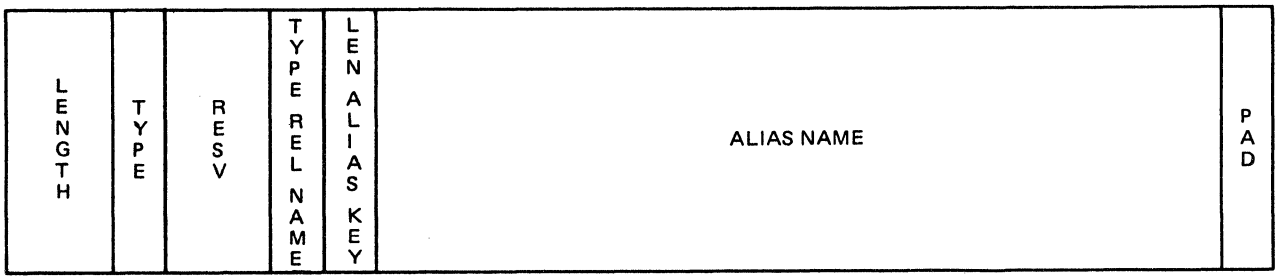
Example of a truename record:

LL	Name Cell	Association Cell
	E3	03

*optional

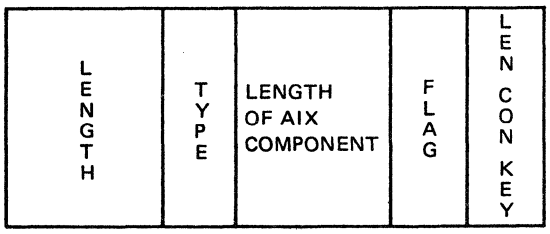
Figure 4. Nonsphere Records

Figures 5 through 20 show the individual formats of the cells that form the sphere and nonsphere records.



RESV = RESERVED
TYPE REL NAME = TYPE OF RELATED NAME

Figure 5. Alias Name Cell Format



LEN CON KEY = LENGTH OF CONDENSED AIX KEY

Figure 6. Alternate Index Name Cell Format

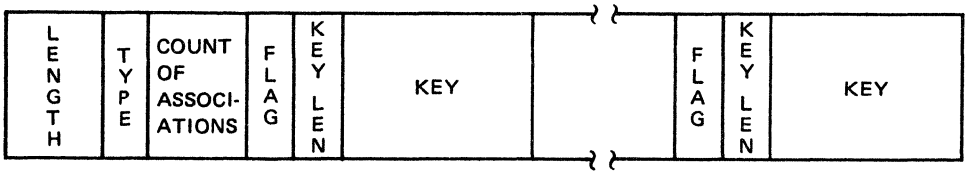
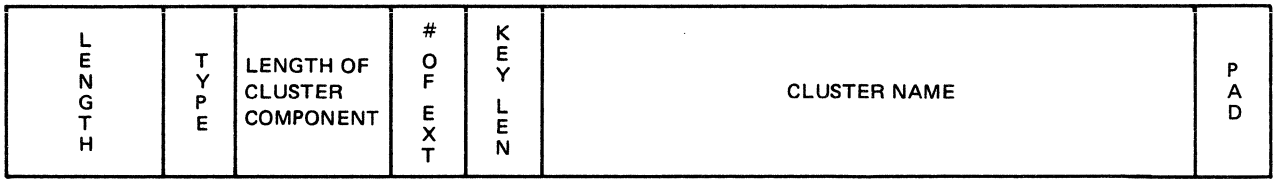
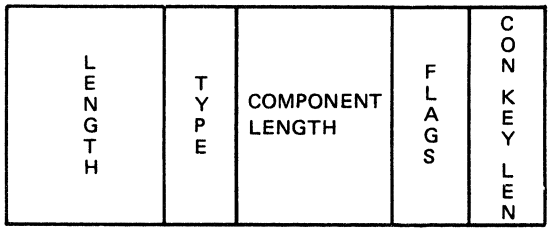


Figure 7. Association Cell Format



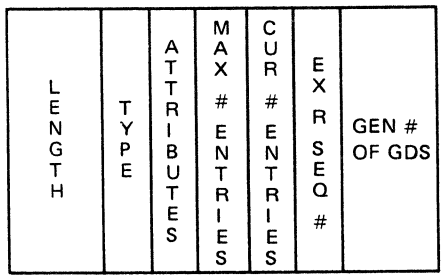
OF EXT = NUMBER OF EXTENSION RECORDS
KEY LEN = LENGTH OF THE CLUSTER KEY

Figure 8. Cluster Name Cell Format



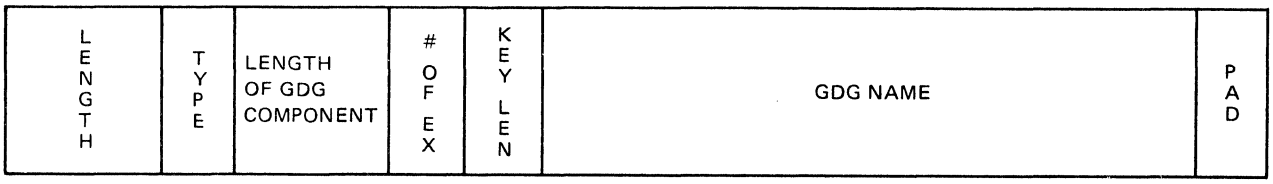
CON KEY LEN = CONDENSED KEY LENGTH

Figure 9. Data Name Cell or Index Name Cell Format



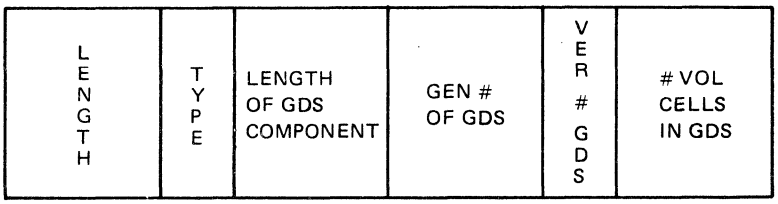
CUR # ENTRIES = CURRENT NUMBER OF ENTRIES IN GAT
MAX # ENTRIES = MAXIMUM GDS ENTRIES IN GDG BASE
EX R SEQ # = EXTENSION RECORD SEQUENCE NUMBER
GEN # OF GDS = GENERATION NUMBER OF GDS

Figure 10. Generation Aging Table Cell Format



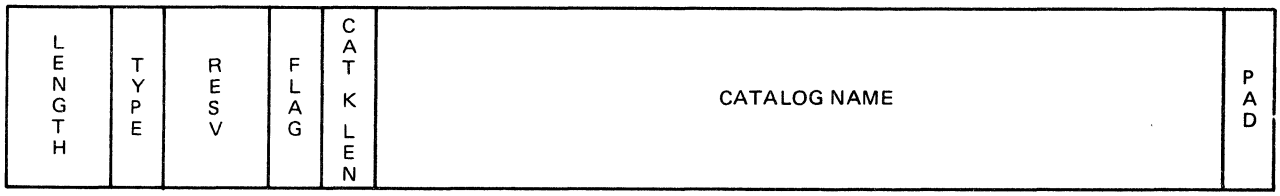
OF EX = NUMBER OF EXTENTION RECORDS
KEY LEN = LENGTH OF GDG KEY

Figure 11. Generation Data Group Name Cell



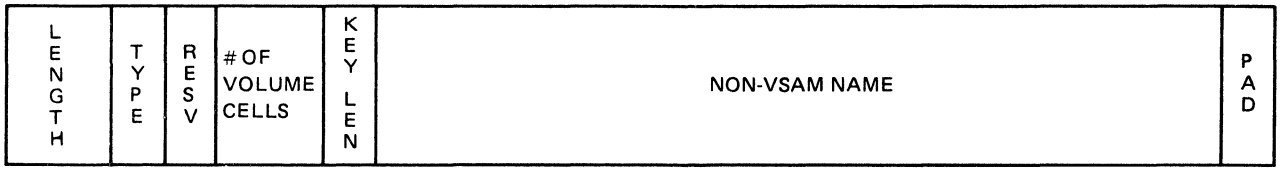
GEN # OF GDS = GENERATION NUMBER OF GDS
VER # GDS = VERSION NUMBER OF GDS

Figure 12. Generation Data Set Name Cell



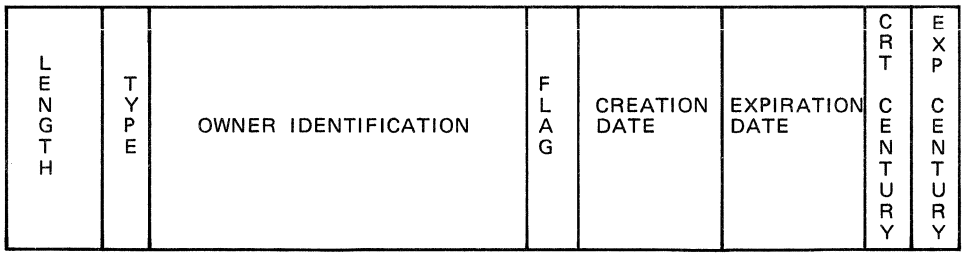
RESV = RESERVED
CAT K LEN = CATALOG KEY LENGTH

Figure 13. Integrated Catalog Facility Connector Name Cell Format



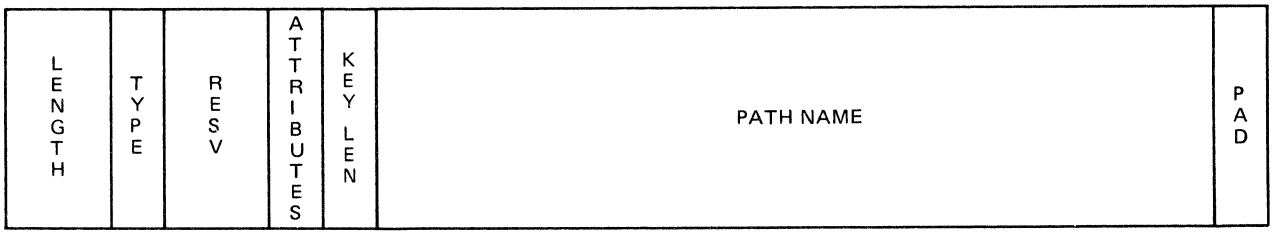
RESV = RESERVED
KEY LEN = LENGTH OF NON-VSAM KEY

Figure 14. Non-VSAM Name Cell



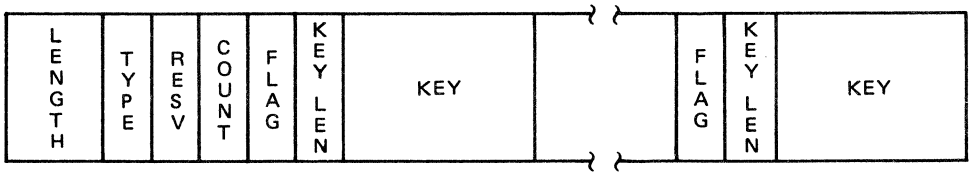
CRT CENTURY = CREATION CENTURY
EXP CENTURY = EXPIRATION CENTURY

Figure 15. Ownership Cell



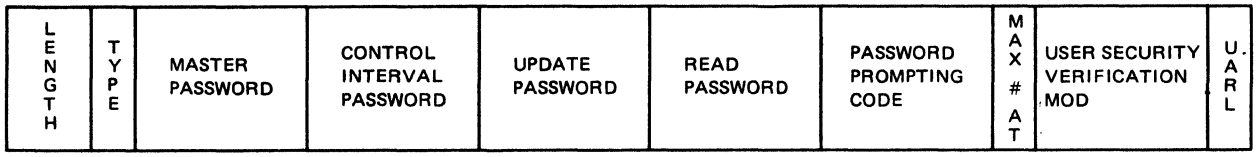
RESV = RESERVED
KEY LEN = LENGTH OF PATH KEY

Figure 16. Path Name Cell



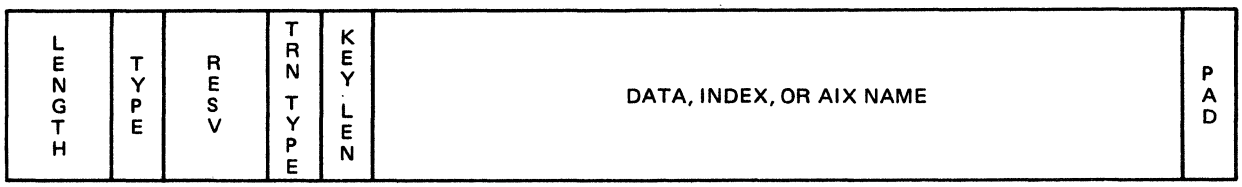
RESV = RESERVED
COUNT = COUNT OF RELKEYS

Figure 17. Relation Cell Format



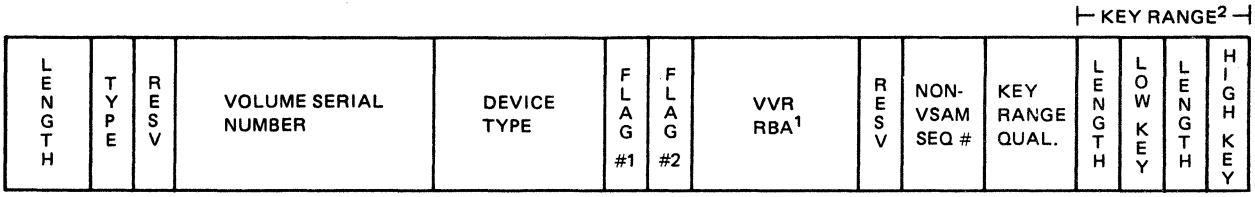
MAX # AT = MAXIMUM NUMBER OF ATTEMPTS
UARL = USER AUTHORIZATION RECORD LENGTH

Figure 18. Security Cell Format



RESV = RESERVED
TRN TYPE = TRUENAME TYPE
KEY LEN = TRUENAME KEY LENGTH

Figure 19. True Name Cell Format



RESV = RESERVED
NON-VSAM SEQ # = NON-VSAM FILE SEQUENCE NUMBER

[1]FOR A NON-VSAM DATA SET, THIS CONTAINS THE FORMAT 1 DSCB TTR.
[2]FOR NON-KEY RANGE DATA SETS, THE LENGTH IS 0.

Figure 20. Volume Cell Format

Record Size and Extension Records for Catalogs

The RECORDSIZE parameter in the DEFINE command allows you to select the average and maximum record size for the logical catalog record. The average record size value must be between 4086 and 32400 bytes and is used to calculate allocated space in records. The default is 4086 bytes. The maximum record size controls the size of the largest spanned record. The maximum value controls, and, in certain catalog records, creates extension records. Figure 21 is an example of an extension record.

Extension records may be created for VSAM sphere records (cluster and its alternate indexes) and GDG sphere records. All other catalog entry types such as non-VSAM, alias, and path do not create extension records. The maximum record size value must be large enough to fit in a spanned record without exceeding the maximum record size. However, an alternate index or GDGs must fit in an extension record.

An extension record is created when:

- An alternate index or GDG is defined and does not fit in the current sphere record.
- A path is defined and the entry required in the association cell does not fit in the sphere record.
- Volumes are added to a cluster or alternate index and the volume cell does not fit in the sphere record.

The key of the extension record is the base cluster or GDG name and the pad character.

The pad character is internally generated, starting with the first extension as X'01', second extension as X'02', to the 240th extension as X'F0', which is the maximum number of extensions allowed.

A component level entity is moved to the new extension record whether it is the component being updated or the last component on the current sphere record. For a VSAM sphere record, this is an alternate index (AIX). For a GDG sphere record, the GDG component is moved. Only one component resides in each extension record.

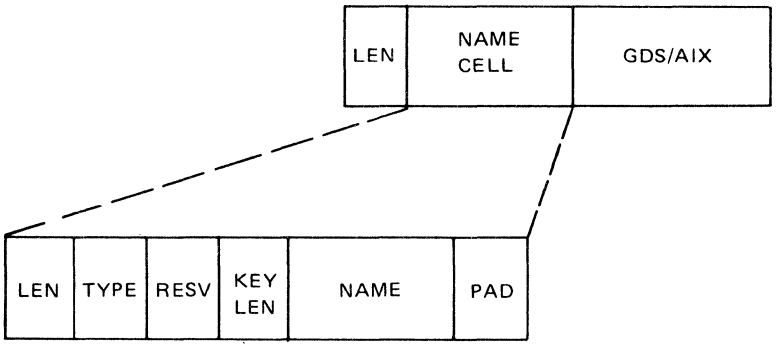


Figure 21. Extension Record

VSAM Volume Data Set (VVDS)

The VVDS is an entry-sequenced data set (ESDS) that has 4K-byte control intervals and contains the information about the VSAM data sets residing on the volume with the VVDS. It also contains names of up to 36 BCSs that have VSAM data set components residing on this volume. Therefore, the VVDS is shared among the BCSs that have the VSAM data sets defined on that volume.

The VVDS is composed of a minimum of two records:

- A VSAM volume control record (VVCR)
- A VVDS self-describing volume record

The first logical record in a VVDS is the VSAM volume control record (VVCR). The second logical record in the VVDS is the VVDS self-describing VVR. The remaining logical records in the VVDS are VSAM volume records (VVRs). See Figure 22 for the structure of a VVDS.

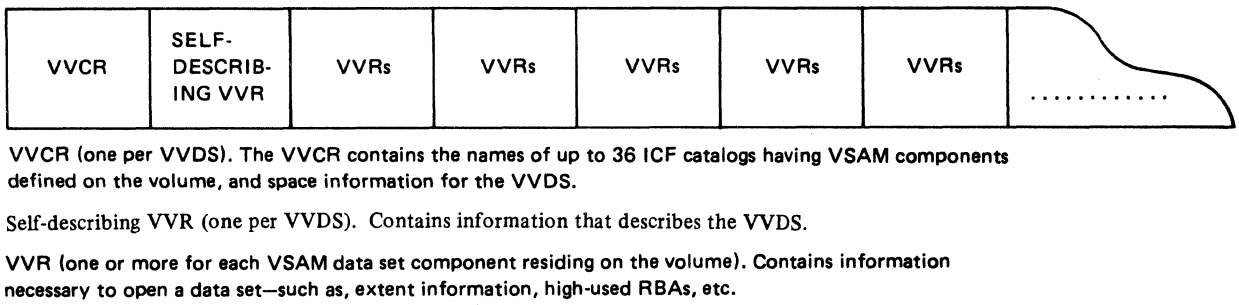


Figure 22. VSAM Volume Data Set (VVDS) Structure

A VVDS may be implicitly or explicitly defined. The VVDS is usually dynamically created (implicitly defined), using default primary and secondary space allocation quantities, when the first VSAM data set or BCS is defined on that volume. Whenever additional space is needed in the VVDS itself, it is extended. You may explicitly define space allocation quantities in the VVDS; use the DEFINE CLUSTER command to override the default from the implicit define.

A VVDS is recognized by the VVDS data set name 'SYS1.VVDS.Vvolser', where *volser* is the volume serial number of the volume on which the VVDS resides.

Because of its special use, you cannot password protect, export, or import a VVDS, nor can you alter its attributes by using the ALTER command.

VSAM Volume Control Record (VVCR)

The VVCR is the first logical record in the VVDS. It contains information for management of DASD space and maintains from 1 to 36 BCS back pointers. There is only one VVCR in a VVDS.

VVDS Self-Describing VVR

The VVDS self-describing VVR is the second logical record in the VVDS. This self-describing VVR contains information that describes the VVDS.

VSAM Volume Record (VVR)

The VVRs are the remaining logical records in the VVDS. The VVRs contain information about the VSAM data sets residing on the volume with the VVDS. If more than one VVR is associated with a component, the first (primary) VVR contains information pertaining to the data set as a whole. The other (secondary) VVR(s) do not repeat the same information in their record(s) but contain information for their own component, such as extents, RBAs, and allocation quantities, and most of the information needed to open a VSAM data set.

The number of VVRs for VSAM data sets varies according to the type of data set and the options specified for the data set. The following list contains the number of primary VVRs for each type of data set.

- Entry-sequenced data set:

  1 VVR per volume for the DATA component

- Key-sequenced data set with the NOIMBED option:

  1 VVR per volume for the DATA component
  1 VVR for the INDEX component

- Key-sequenced data set with the IMBED option:

  1 VVR per volume for the DATA component
  1 VVR for the INDEX component
  1 VVR per volume for the sequence set

- Key range key-sequenced data set with the NOIMBED option:

  1 VVR per key range per volume
  1 VVR for the INDEX component

- Key range key-sequenced data set with the IMBED option:

  2 VVRs per key range per volume
  1 VVR per volume for the INDEX component

The size of a VVR depends on its type (primary or secondary) and is measured in terms of its length in bytes. The size of a VVR is determined by the combined lengths of VVR cells. Figure 23 on page 17 shows primary and secondary VVRs and the cells that constitute each VVR. These VVR cells are:

- VVR header cell
- VVR data set information cell
- VVR AMDSB cell
- VVR volume information cell

The VVRLEN field of the VVR header cell contains the length of the entire VVR. The VVRTYPE field of the same cell contains the VVR type code, which is either "Z" (for primary) or "Q" (for secondary).

Figure 24 on page 18 shows the information contained in each type of VVR cell. Further information about VVR cells is in Appendix A, "The Integrated Catalog Facility Catalog Cell Structure" on page 149. For the complete format of each VVR cell, see *Catalog Diagnosis Reference*.

Non-VSAM data sets do not have information in the VVDS.

VVR CELLS FOR PRIMARY VOLUME VVR

VVR LEN	VVR HEADER CELL	VVR DATA SET INFO CELL	VVR AMDSB CELL	VVR VOLUME CELL

VVR CELLS FOR SECONDARY VOLUME VVR

VVR LEN	VVR HEADER CELL	VVR VOLUME CELL

Figure 23. VSAM Volume Record (VVR)

VVR HEADER CELL:

LEN	TYPE	FLAG	KRQ	COMP NAME LEN	COMP NAME	CLS NAME LEN	CLS NAME	CAT NAME LEN	CAT NAME	BASE CLS NAME LEN	BASE CLS NAME

VVR DATA SET INFORMATION CELL:

LEN	TYPE	ATTR1	ATTR2	OPN IND	BUF SZ	PRI SPC	SEC SPC	SPC OPT	DS HU RBA	DS HA RBA	LRECL	RESV	EX-CEP. EXIT	DS HK RBA	CLS ATR	TIME STMP	AL TAB TIME STMP

VVR VOLUME INFORMATION CELL:

LEN	TYPE	VOLFG	#EXT	HKRBA	HURBA	HARBA	BLK SZ	BLK PER TRK	TRK PER AU	TYP EXT	TRK PER CYL	BYT PER TRK	BYT PER AU	LOW KEY LEN	LOW KEY	HI KEY LEN	HI KEY	EXT LEN	EXT

Figure 24. Examples of VVR Cell Information

Assigning Space to an Integrated Catalog Facility Catalog

When defining an integrated catalog facility catalog, use the TRACKS, CYLINDERS, or RECORDS parameter to assign space. If you select RECORDS, you may specify an average record size; the standard default is 4086 for a key-sequenced data set with the spanned attribute. If you specify CYLINDERS, the control area size is 1 cylinder. If space is allocated at the catalog level and not the data or index level, space is subdivided between the data and index according to other defined options.

Because the concept of suballocation and unique space does not exist, the integrated catalog facility catalog resides in its own extents. Like a standard VSAM data set, the catalog may have as many as 123 extents, but it is still limited to a single volume. For further information, see "Estimating the Catalog's Space Requirements" on page 27.

Catalog Control Interval and Control Area Size

Because the BCS is a VSAM key-sequenced data set, the standard control interval and control area calculations are used. See "Estimating Space Requirements for the BCS" on page 27.

The size of the index control interval is the same as the physical block size: from 512 to 8K bytes by multiples of 512 and from 8K to 32K bytes by multiples of 2K. The data component is a multiple of the physical block size that can reside in one control interval. The resulting values for the catalog should be the same as for a key-sequenced data set with the spanned attribute.

Volume Table of Contents (VTOC) Support

The integrated catalog facility catalog supports data sets in both VTOCs and indexed VTOCs. However, the VTOC still maintains the restriction of 16 physical extents for non-VSAM data sets. For VSAM data sets in the integrated catalog facility environment, the extent limit is increased to 123. If space is allocated in cylinders, 1 to 5 extents may be allocated by DADSM per allocation or extent request. Otherwise, DADSM will allocate 1 extent of contiguous tracks.

VTOC Entries for Integrated Catalog Facility Catalogs

There are VTOC DSCBs (data set control blocks) corresponding to the BCS and each VVDS. Two new flags exist in the Format-1 DSCB in the OPTCD field, which were formerly unused by VSAM. OPTCD of X'80' indicates the VSAM data set is contained in the integrated catalog facility catalog and OPTCD of X'40' indicates it is an integrated catalog facility catalog.

The data set name fields in the Format-1 DSCB are described below. In generated names, yyddd is the date and Taaaaaaa and Tbbbbbbb are character strings that form the timestamp values to tell the time of day when the component was defined.

- **BCS**

  The BCS is a key-sequenced data set and has a data and an index component in the VTOC. The data component has the name of the integrated catalog facility catalog (the cluster name is 44 bytes of binary zeros). The index component has a generated name in the form:

  CATINDEX.Tbbbbbbb.VIDyyddd.Taaaaaaa

- **VVDS**

  The VVDS is an entry-sequenced data set and has a data component in the VTOC. The data component had the name of the VVDS, which is:

  SYS1.VVDS.Vvolser

  where volser is the volume serial of the DASD device on which it is defined.

- **Clusters**

  Separate entries appear in the VTOC for the data and index components of a cluster. When a cluster is defined in an integrated catalog facility catalog, the user can specify names for the data and index components. If this is not done, the names are generated with the following format:

  Index: hlq.Tbbbbbbb.VIDyyddd.Taaaaaaa

  where *hlq* is the high-level qualifier of the cluster.

Because the two components are not created simultaneously, each has a different value of Taaaaaaa and Tbbbbbbb in its data set name. The cluster name does not appear in the VTOC.

- **Page Spaces**

  A page space is an entry-sequenced data set. Page spaces do not have explicitly named components. Page spaces always have a generated name of the data type, with the *hlq* equal to the high-level qualifier of the page space name.

  Data: hlq.Tbbbbbbb.VDDyyddd.Taaaaaaa

- **Keyrange Data Sets**

  A keyrange data set only has one index component. However, a data component exists for each key range. If the data component is generated, it will have the following format in the VTOC:

  hlq.Tbbbbbbb.VIDyydd.Taaaaaaa

  Each key range data component has the following format whether the name is generated or explicitly named:

  dddee.rnnn

  where:

  ddd = the first 37 characters of the data set name

  ee = the last 2 characters of the data set name

  r = the character "A." If the name in the data set is already allocated in the VTOC, "A" is changed to "B." If "B" is already allocated, it is changed to "C," with this procedure continuing until an unallocated name is entered.

  nnn = 001 for the first key range, 002 for the second key range, 003 for the third key range, etc.

Chapter 3. Defining, Altering, and Deleting a Catalog

This chapter only deals with defining, altering, and deleting integrated catalog facility catalogs. Chapter 4 contains instructions for converting VSAM catalogs or OS CVOLS to integrated catalog facility catalogs.

Using Access Method Services

Access method services consists of the following functional commands:

- ALTER alters previously defined catalog entries.
- BLDINDEX constructs alternate indexes for existing data sets.
- CHKLIST identifies tape volumes mounted when a checkpoint was taken.
- CNVTCAT converts entries in an OS CVOL or VSAM catalog into integrated catalog facility catalog entries.
- DEFINE creates catalog entries for data sets and catalogs.
- DELETE deletes catalog entries.
- DIAGNOSE scans a BCS or VVDS to validate the data structures to detect structure errors.
- EXAMINE determines whether structural errors exist in the index component and/or the data component of a BCS.
- EXPORT creates a copy of a catalog or a data set for backup or to be portable so that it can be used on another system. EXPORT also disconnects the catalog for use on another system.
- IMPORT reads a backup copy of a VSAM data set or makes a data set that was previously exported from one system to be available for use in another system, restores a catalog, or connects the catalog.
- LISTCAT lists catalog entries.
- PRINT prints data sets or catalogs.
- REPRO:
  - Copies or merges data sets
  - Copies catalogs
  - Splits catalog entries between two catalogs
  - Merges catalog entries into another user catalog or master catalog
- VERIFY causes a catalog to reflect the end of a data set correctly after an error that prevented closing a VSAM data set. The error may have caused the catalog to be incorrect.

Many access method services commands require the volumes that contain the VVDS entries for the object to be mounted, as well as the volume that contains the BCS of the integrated catalog facility catalog to be mounted.

The following commands and parameters require that the VVDS volume be mounted:

- ALTER
  - BUFFERSPACE
  - ERASE|NOERASE
  - EXCEPTIONEXIT
  - INHIBIT|UNINHIBIT
  - NEWNAME
  - NOUPGRADE|UPGRADE
  - RECORDSIZE
  - SHAREOPTIONS
- CNVTCAT
  - OS CVOL to integrated catalog facility catalog
  - VSAM catalog to integrated catalog facility catalog
- DEFINE
  - ALTERNATEINDEX
  - CLUSTER
  - PAGESPACE
  - USERCATALOG
- DELETE
  - ALTERNATEINDEX
  - CLUSTER
  - USERCATALOG
- EXPORT

  Except for EXPORT DISCONNECT
- IMPORT

  Except for IMPORT CONNECT
- LISTCAT
  - ALL
  - ALLOCATE
- REPRO

For further information about access method services commands, see *Access Method Services Reference.*

Using the DEFINE Command

A catalog is the central information point for all VSAM data sets and the direct access volumes on which they are stored. For an integrated catalog facility catalog, use the access method services DEFINE command to define VSAM or non-VSAM objects.

When you issue the DEFINE command to catalog an object, access method services causes a catalog entry to be built that describes the object. The objects you can define are:

- A user catalog, a collection of information about non-VSAM and VSAM objects that reside on that volume. You can create a user catalog at any time. A connector to the user catalog is put in the system's master catalog. You may also define a user catalog into another user catalog.
- A cluster, or VSAM data set, a collection of a user's data records. There are four types of data cluster organizations:
  - Entry sequenced, or sequential, in which data records are read or written sequentially from one end to the other (first to last for writing, either direction for reading).
  - Key sequenced, or indexed, in which a data record is read or written based on its key value. A key is a field, in 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).
  - Linear, a data set that has no record definition field (RDF), no control interval definition field (CIDF), and can be accessed only in control interval mode. It can be defined only in an integrated catalog facility catalog.
- An alternate index, in which a data record from a key-sequenced or entry-sequenced cluster (called the base cluster) is read or written based on an alternate key.
- A path, a data set name for the combination of an alternate index and its base cluster, or an alias for a VSAM cluster.
- A page space, an amount of direct access device space to be used exclusively by the system.

The non-VSAM objects to define are:

- A non-VSAM data set, a collection of data records with sequential, partitioned, direct, or index-sequenced data organization (not VSAM data organization).
- A generation data group, a collection of non-VSAM data sets that are grouped together in a time-dependent manner.
- An alias, an alternate name for a user catalog, non-VSAM data set or OS CVOL.

Every system must have a master catalog before the initial program load (IPL). Before an object can be defined, there must be a catalog in which to define the object.

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 space allocation. After the object is defined, it can be processed with other access method services commands and with the user's program. After a cluster is defined, data records can be loaded by using the REPRO command.

Using the Parameters for the DEFINE Command

When you define a catalog, cluster, or alternate index, you can specify attributes in several different ways. The parameter set for DEFINE USERCATALOG, CLUSTER, and ALTERNATEINDEX is directly related to the way the attributes are stored in the integrated catalog facility catalog. The catalog entries that describe an integrated catalog facility catalog, cluster, or alternate index are:

- The cluster entry describes the attributes of the cluster or catalog as a whole, primarily protection attributes.
- The alternate index entry describes the attributes of the alternate index as a whole, primarily protection attributes.
- The data entry describes the attributes of the data component of a catalog, cluster, or alternate index.
- The index entry describes the attributes of the index component of a catalog, key-sequenced cluster, or alternate index.

When you specify attributes as parameters of USERCATALOG, CLUSTER, or ALTERNATEINDEX, consider the following:

- Attributes specified in the parameters are defined in the cluster or alternate index entry of an integrated catalog facility catalog if they pertain to that entry; for example, protection attributes.
- Attributes specified in the parameters are defined to the data and/or index entries they pertain to. For example, the WRITECHECK parameter of CLUSTER is not defined in the CLUSTER entry, but is entered in the DATA and INDEX entry in the catalog.
- Except for protection attributes, if the same attribute is specified as a subparameter of DATA and/or INDEX, the value of the attribute specified at the DATA/INDEX level overrides the value of the attribute specified at the USERCATALOG, CLUSTER, or ALTERNATEINDEX level.

You may use the LISTCAT command with the ALL option to list catalog entries and to determine where the various attributes are stored in the catalog. Appendix B,"Interpreting LISTCAT Output Listings," in *Access Method Services Reference* describes the attributes for each type of entry.

Because access method services directs the specified attributes to the proper entry, attributes as parameters of DATA or INDEX do not have to be specified. However, attributes as parameters of DATA and INDEX may be specified as:

- A name for the data and index components of a cluster or alternate index.
- Unique protection attributes for the data and index components of a cluster or alternate index.
- Space allocation for the data component only, or for the data components and index components of a catalog, key-sequenced cluster, or alternate index.
- Volumes of a different device type for the data and index components of a key-sequenced cluster or alternate index.
- A different variation of the same attribute for the data and index components of a catalog, key-sequenced cluster, or alternate index. For example, you may want to specify WRITECHECK for the data component and NOWRITECHECK for the index component, or vice versa.

The level at which you specify attributes can also be affected if you use the MODEL parameter of the DEFINE command (see "Using One Catalog As a Model for Another Catalog" on page 39).

Defining a BCS (DEFINE USERCATALOG)

DEFINE USERCATALOG ICFCATALOG creates a BCS and an implicit VVDS if the VVDS does not exist. Space is allocated to the BCS in the same way as for a key-sequenced data set. The command syntax requires that some type of space parameter (RECORDS, TRACKS, or CYLINDERS) *always* be specified at the USERCATALOG level and may also be specified with the DATA component or with the DATA and INDEX components.

- When a space parameter is specified only at the USERCATALOG level, it is assigned to the DATA and INDEX components according to existing procedures.
- When a space parameter is also specified at the DATA component level, but not at the INDEX level, the USERCATALOG space parameter is ignored and space is allocated to the DATA and INDEX components based on the size of the DATA component.
- When a space parameter is specified at the INDEX component level, it must also be specified at the USERCATALOG and DATA levels.

If the ICFCATALOG parameter is not coded as specified, a VSAM catalog is defined as the default (existing job streams will continue to work).

If a VVDS does not already exist and an integrated catalog facility catalog is defined, an attempt is made to create a VVDS implicitly as part of the DEFINE command. If the VVDS cannot be implicitly defined because there is not enough space on the volume, processing is terminated and a message is issued. Lack of space may occur because the volume is full of data sets or a VSAM catalog owns all the space.

The following attributes may not be specified when defining an integrated catalog facility catalog and will result in an error message:

- BIND
- INDEXED
- KEYS
- SPEED
- NOREUSE
- UNIQUE
- RECOVERABLE

You may specify performance-related and buffer-related parameters, and catalog sharing when you define the integrated catalog facility catalog.

The tuning attributes that may be specified when defining an integrated catalog facility catalog are:

- CONTROLINTERVALSIZE—defaults determined as for VSAM clusters
- FREESPACE—defaults to (0 0)
- IMBED|NOIMBED—defaults to IMBED
- RECORDSIZE—defaults to (4086 32400)
- REPLICATE|NOREPLICATE—defaults to NOREPLICATE
- SHAREOPTIONS (3,3 or 3,4)—defaults to (3,4)
- STRNO—minimum value 2; maximum value 255—defaults to 2
- BUFND—minimum value is STRNO plus 1 (If STRNO is specified, the BUFND default is 3. If STRNO is not specified, the BUFND default is 2.)
- BUFNI—minimum value is STRNO plus 1 (If STRNO is specified, the BUFNI default is 3. If STRNO is not specified, the BUFNI default is 2.)

SHAREOPTIONS may only be specified at the data component level but are propagated to the index component. DATA and INDEX will always have the same SHAREOPTIONS value.

STRNO specifies the number of concurrent RPLs VSAM record management can handle. This determines the maximum number of nonupdate catalog requests that can be handled concurrently; additional requests must wait. To specify strings for an integrated catalog facility catalog, use the STRNO parameter. For further information, see *Access Method Services Reference.*

BUFND and BUFNI can be specified for buffer storage to be used for catalog data record buffers (BUFND) and catalog index record buffers (BUFNI). They specify the number of buffers, of the appropriate control interval size, to be used when the integrated catalog facility catalog is opened.

The STRNO, BUFND, and BUFNI parameters can also be specified as AMP parameters in a JOBCAT or STEPCAT DD statement for a user catalog, if the catalog is not already opened. Specification of one or more of these parameters on a JOBCAT or STEPCAT DD statement overrides the define time specification for the duration of the DD statement. The related value in the catalog is not changed.

Defining a VVDS (DEFINE VVDS NORECATALOG)

A VVDS can be defined either:

- Explicitly, via DEFINE CLUSTER, or
- Implicitly, when the first VSAM data set is defined in the integrated catalog facility catalog or a BCS is defined on the volume.

The DEFINE CLUSTER command for a VVDS is recognized by the cluster name SYS1.VVDS.Vvolser and is treated as a special case by access method services. Space is allocated on the requested volume, and the VVDS is built. However, an explicitly defined VVDS is unrelated to any BCS, because the VVCR, which is the first record in the VVDS, contains no pointers to BCSs. The second record in the VVDS contains the VVR. An explicit DEFINE of the VVDS does not update any BCS and, therefore, can be performed before the first BCS in the installation is defined. The VVDS cluster record is added to a related BCS, and the BCS back pointer is put into the VVCR when the first data set is defined on the volume containing the VVDS.

In most cases, it is appropriate to allow the VVDS to be defined implicitly with the default SPACE allocation of TRACKS(3 2). See "Estimating Space Requirements for the VVDS" on page 30.

One advantage of an explicit definition of the VVDS is the ability to change the SPACE allocation from its default value. However, it may be desirable to control the location of the VVDS to make it adjacent to the BCS data component.

Using an Alias to Identify a User Catalog

When you define an alias for a user catalog connector, you should structure the alias so the catalog's cataloged data sets can be located when the alias is used. If the system is searching for a user-specified entry name and does not find its entry in the master catalog or a user catalog identified with the JOBCAT or STEPCAT DD statements, and if the entry name's catalog is not specified, the catalog assumes that:

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

In order to use an alias to identify the user catalog to be searched, the entry name or the generation data group base name must be a qualified name.

For a detailed description of the correct search sequence of the various commands, see "Order of Catalog Use" in *Access Method Services Reference.*

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

Estimating Space

Estimating the Catalog's Space Requirements

The integrated catalog facility catalog does not use fixed-length records to store variable-length information. An integrated catalog facility catalog results in a considerable reduction in DASD space requirements; however, it also makes it impossible to precisely estimate DASD space usage requirements with any straightforward algorithm. The following information serves only as an *approximation* for your catalog space requirements.

Estimating Space Requirements for the BCS

To estimate the size of a BCS, it is useful to know the record sizes for different types of catalog entries. Actual record sizes depend on the:

- Length of the data set component names
- Number of volumes per data set
- Number of relationships between components
- Number of alternate indexes
- Number of paths
- Presence or absence of security information

The table below lists the approximate *minimum* lengths (in bytes) for various records, and the variables that affect the length of each record.

Record	Approx. Minimum Length	Variables Affecting Length
GDG	260	Length of GAT cell; number and length of association and volume cell(s).
GDS	90	Number and length of volume cell(s); presence of association cell.
ALIAS	60	Length of association cell.
Non-VSAM	90	Number and length of volume cell(s); presence of association cell.
User Catalog Connector	90	Number and length of volume cell(s); presence of association cell.
ESDS	230	Number and length of association and volume cell(s); presence of security cell(s).
KSDS	400	Number and length of alternative index subrecords and volume cells; presence of association, relation, and security cell(s).
LDS	230	Same as for ESDS
RRDS	230	Same as for ESDS
Alternate Index	170	Length of association, data name, index name and volume cell(s).
Truename	70	Length of association cell.
Path	80	Length of association cell; presence of security cell.

The space parameters that are specified to provide for the estimated size of the BCS should take into account the:

- Data control interval size
- Device type
- Data control area size
- Embedded index (by default)

The relationships between these factors and the space requirements are the same as those that apply to standard key-sequenced data sets.

Because most of the processing is random and no benefit is gained from large control interval sizes, the control interval size for the data component should be specified as 4096 bytes. Most data records are a few hundred bytes long, and a 4096-byte control interval size provides a useful compromise between minimizing data transfer time and reducing the incidence of a record spanning a control interval.

The control area size for the data component is effectively controlled by the secondary allocation quantity. If the secondary allocation quantity is:

- One track or less (allocation is records), the control area size is 1 track.
- Greater than 1 track but less than 1 cylinder, the control area size is the same as the secondary allocation quantity.
- One cylinder or greater, the control area size is 1 cylinder.

The control area size should be large enough to contain a maximum length record; the default maximum record length for the BCS (a spanned record data set) is 32400 bytes. The following list shows the smallest data control area sizes that can contain a default maximum length spanned record for a 4096-byte data control interval size:

Device Type	Data Control Area (tracks)
IBM 3380[1]	1
IBM 3375	1
IBM 3350	3
IBM 3340	6
IBM 3330	3

If a smaller control area size is selected, the maximum record size must be reduced to fit in a control area by specifying the RECORDSIZE parameter with the DEFINE USERCATALOG command.

To optimize the performance of the BCS, keep the number of levels in the index to a minimum. If no more than two index levels are to be used, a selected index control interval size will limit the capacity of the data set.

Too large a control area size extends the index beyond two levels. The following list shows the maximum control area sizes to maintain a two-level index for two different index control interval sizes. Because a second-level index record can address up to 121 data control areas, the total data capacity from these allocations is also given:

Device Type	Data Control Area (tracks)	Index Control Interval Size (bytes)
IBM 3380[1]	3	1024
	1	512
IBM 3350	8	1024
	3	512
IBM 3330	11	1024
	5	512

If the BCS is defined with the IMBED attribute, an additional track for each control area is required for the sequence set records.

The process for estimating the space allocation for the BCS is to:

1. Estimate the total record size requirements in number of 4096-byte data control intervals.
2. Increase this quantity by the amount of free space required (for example, 20%).
3. Select an index control interval size and data control interval size based on the total data requirement and the device type.
4. Calculate the number of data control areas required.
5. Calculate total number of tracks required based on selected control area size plus 1 additional track for each control area for the sequence set records.

---

[1] 3380 models AO4, AA4, BO4, AD4, BD4, AE4, and BE4

6. Specify the space allocation for the data component as TRACKS (P S), where P was calculated above and S is the selected data control area size.
7. Specify index space allocation as TRACKS (1 1).

The above procedure assumes a reorganized BCS without unused space caused by control interval and control area splits. An empty catalog just defined contains records for the catalog cluster, with its data and index components and also the VVDS. Subsequent additions to the catalog are insertions and may cause a significant number of control interval and control area splits, resulting in an underused BCS. The amount of space required during the initial period when CNVTCAT or DEFINE is used to build entries in the BCS may exceed the estimated size, possibly by a factor of 4. Reorganizing the BCS by using EXPORT and IMPORT reduces the space requirements to a value nearer the one resulting from the above estimate.

The definition of the BCS can include the specification of other parameters to select performance characteristics. The number of concurrent catalog accesses can be specified directly by using the STRNO parameter. Initially, use the default value of STRNO(2). This can be increased if frequent enqueue contention occurs for the resource SYSZRPLW.catname. The major name is SYSZRPLW and the minor name is catname.

Estimating Space Requirements for the VVDS

The VVDS contains VSAM volume records (VVRs) that contain information about VSAM data sets residing on that physical volume.

Unless it is anticipated that a large number of VSAM data sets will reside on a physical volume, you may allow the system to automatically define the VVDS with standard default allocation quantities of 3 tracks primary and 2 tracks secondary. "VSAM Volume Record (VVR)" on page 16 lists the number of VVRs for each type of data set. It also explains how to determine the size of a VVR. You may want to refer to that section before you compute the space you need.

Use the following formula to compute the approximate space requirements for a VVDS:

Primary = (G x A) + B

where:

G = the number of VVRs in the VVDS (exclusive of self-describing VVR)
A = the average size of a VVR
B = CIs for VVCR and self-describing VVR (8K bytes)

Protecting the Catalog

The protection of data includes:

- Data security—the safety of data from theft or intentional destruction
- Data integrity—the safety of data from accidental loss or destruction

The following sections describe the data protection available for an integrated catalog facility catalog including:

- Authorized program facility (APF)
- Access method services password protection

- Resource Access Control Facility (RACF)
- User-security-verification routine (USVR)
- IGG.CATLOCK facility

Authorized Program Facility (APF)

The authorized program facility (APF) limits the use of sensitive system services and resources to authorized system and user programs.

For information about program authorization, see "Authorized Program Facility (APF)" in *Supervisor Services and Macro Instructions.*

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

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

The following situations could cause the invalidation of APF authorization for access method services:

- An access method services module is loaded from an unauthorized library.
- A user-security-verification routine (USVR) is loaded from an unauthorized library during access method services processing.
- An exception exit routine is loaded from an unauthorized library during access method services processing.
- A user-supplied special graphics table is loaded from an unauthorized library during access method services processing.

Because APF authorization is established at the job step task level, access method services is not authorized if invoked by an unauthorized problem program or by an unauthorized terminal monitor program (TMP).

Under TSO, if the system does not have the TSO Command Package Program Product, you can authorize your TMP by relink-editing it with the SETCODE AC(1) attribute. You must enter the names of those access method services commands requiring APF authorization to execute under TSO in the authorized command list.

The restricted functions performed by access method services that cannot be requested in an unauthorized state are:

- CNVTCAT—when converting to an integrated catalog facility catalog
- DEFINE—when the RECATALOG parameter is specified
- DELETE—when the RECOVERY parameter is specified
- EXPORT—when the object to be exported is a BCS
- IMPORT—when the object to be imported is a BCS
- PRINT—when the object to be printed is an integrated catalog facility catalog
- REPRO—when the BCS is copied or merged
- VERIFY—when a BCS is to be verified

If the above functions are required and access method services is invoked from a problem program or a TSO terminal monitor program, the invoking program must be authorized.

Access Method Services Password Protection

Access method services provides options to protect data sets against unauthorized use and loss of data. To effectively use the protection features, you must understand the difference between operations on a catalog and operations on data sets represented by a catalog entry:

- Referring to a catalog entry when new entries are defined (DEFINE), or existing entries are altered (ALTER), deleted (DELETE), or listed (LISTCAT)
- Using the data set represented by a catalog entry when it is connected to a user's program (OPEN), or disconnected (CLOSE)

Different passwords may be needed for each type of operation.

Operations on a catalog may be authorized by the catalog's password or, in some cases, by the password of the data set defined in the catalog. *Access Method Services Reference* describes which level of password is required for each operation.

The following are examples of passwords required for defining, listing, and deleting catalog entries.

- Defining a data set in a password-protected catalog requires the catalog's update (or higher) password.
- Listing or deleting a data set's catalog definition requires the appropriate password of either the catalog or the data set. However, if the catalog, but not the data set, is protected, no password is needed to list the data set's catalog definition, or to alter or delete the data set's catalog entry.

OPEN and CLOSE operations on a data set may be authorized by the password pointed to by the PASSWD parameter of the ACB macro. The "ACB Macro" section in *VSAM Administration: Macro Instruction Reference* describes which level of password is required for each type of operation.

Passwords to Authorize Access

You may, optionally, define passwords for access to clusters, cluster components (data and index), page spaces, alternate indexes, alternate index components (data and index), paths, master and user catalogs. Different passwords have different degrees of security, with higher levels providing greater protection than lower levels. The levels are:

- Full access. This is the master password, which allows you to perform all operations (retrieving, updating, inserting, and deleting) on an entire VSAM data set and any index and catalog record associated with it. The master password allows all operations and bypasses any additional verification checking by the user-security-verification routine.
- Control access. This password authorizes you to use control interval access. For further information on control interval access, see *VSAM Administration Guide*.

- Update access. This password authorizes you to retrieve, update, insert, or delete records in a data set. The update password does not allow you to alter passwords or other security information.
- Read access. The read-only password allows you to examine data records and catalog records, but not to add, alter, or delete them, nor to see password information in a catalog record.

If you define passwords for any data sets in a catalog, you must also define passwords for the catalog in order for the data set passwords to have effect. If you do not define passwords for the catalog, no password checking will take place during operations on the data set's catalog entries.

Each higher-level password allows all operations permitted by lower levels. The existence of a higher-level password does not, however, prevent functions allowed at a lower level of authority. Catalog passwords prevent classes of functions from occurring; they do not protect the catalog itself. For example, a catalog with a master password does not require master authority for its use. The master password permits those functions requiring master authority to be done.

Any level may be null (not specified), but, if a low-level password is specified, the DEFINE and ALTER commands give the higher passwords the value of the *highest* password specified. For example, if only a read-level password is specified, the read-level becomes the update-, control-, and master-level password as well. If you specify a read password and a control password, the control password value will become the master-level password as well. However, in this case, *the update-level password will be null* because the value of the read-level password will not be given to higher passwords.

Catalogs are themselves VSAM data sets, and may have passwords. For some operations (for example, listing all the catalog's entries with their passwords or deleting catalog entries), the catalog's passwords may be used instead of the entry's passwords. If the master catalog is protected, the update- or higher-level password is required when defining a user catalog, because all user catalogs have an entry in the master catalog. When deleting a protected user catalog, the user catalog's master password must be specified.

Some access method services operations may involve more than one password authorization. For example, importing a data set involves defining the data set and loading records into it. If the catalog into which the data set is being imported is password protected, its update-level (or higher-level) password is required for the definition; if the data set is password protected, its update-level (or higher-level) password is required for the load. The IMPORT command allows you to specify the password of the catalog; the password, if any, of the data set being imported is obtained by the commands from the exported data.

Every VSAM data set is represented in a catalog by two or more components: a cluster component and a data component, or, if the data set is a key-sequenced data set, a cluster component, a data component, and an index component. Of the two or three components, the cluster component is the controlling component. Each of the two or three components can have its own set of four passwords; the passwords you assign have no relationship to each other. For example, password-protecting a cluster but not the cluster's data component, allows someone to issue LISTCAT to determine the name of your cluster's data component, open the data component, and access records in it, even though the cluster itself is password protected.

One reason for password-protecting the components of a cluster is to prevent access to the index of a key-sequenced data set, because one way to gain access to an index is to open it independently of the cluster. (For a description of access to an index, see *VSAM Administration Guide*.)

Password Protection Considerations and Precautions

For a Catalog

Observe the following precautions when using protection commands for the catalog:

- To create a catalog entry (with the DEFINE command), the update- or higher-level password of the catalog is required.
- To modify a catalog entry (with the ALTER command), the master password of the entry or the master password of the catalog that contains the entry is required. However, if the entry to be modified is a non-VSAM or GDG entry, the update-level password of the catalog is sufficient.
- To gain access to passwords in a catalog (for example, to list or change passwords), specify the master-level password of either the entry or the catalog. A master-level password must be specified with the DEFINE command to model an entry's passwords.
- Deleting a protected data set entry from a catalog requires the master-level password of the entry or the master-level password of the catalog containing the entry. However, if the entry in a VSAM catalog describes a VSAM data space, the update-level password of the catalog is sufficient.
- To delete a non-VSAM, GDG, or alias entry, the update level password of the catalog is sufficient.
- To list catalog entries with the read-level passwords, specify the read password of the entry or the catalog's read-level password. However, entries without passwords may be listed without specifying the catalog's read-level password. To list the passwords associated with a catalog entry, specify the master password of the entry or the catalog's master password.

  To avoid unnecessary prompts, specify the catalog's password, which allows access to all entries that the operation affects. A catalog's master-level password allows you to refer to all catalog entries. However, a protected cluster cannot be processed with the catalog's master password.
- Specification of a password where none is required is always ignored.

For a Data Set

Observe the following precautions when using protection commands for data sets:

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

stricted access to the VSAM data set as a whole but protects against unauthorized modification of the data or index as separate components.

Relation of Data Set and Catalog Protection

If you define passwords for any data sets in a catalog, you must also protect the catalog by defining passwords for the catalog or by defining the catalog to RACF. If you do not protect the catalog, no password checking takes place during operations on the data set's catalog entries or during open processing of data sets cataloged in that catalog.

Password Prompting

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

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

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

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

Note: Using the TSO logon password counts as one attempt.

When you define a non-VSAM data set in an integrated catalog facility catalog, the data set is not protected with passwords in its catalog entry. To protect a non-VSAM data set with a password when the data set is created, specify LABEL=(PASSWORD|NOPWREAD) in the DD statement that describes the data set (for more details, see *JCL Reference* and *JCL User's Guide*). Use the PROTECT macro instruction to assign a password to the non-VSAM data set (for more details, see *Data Administration Guide* and *System—Data Administration*).

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

Resource Access Control Facility (RACF)

Resource Access Control Facility (RACF) provides an optional software access control measure you may use in addition to or instead of passwords. Password protection and RACF protection can coexist for the same data set. When RACF protection is applied to a data set that is already password protected, password protection is bypassed and access is controlled solely through the RACF authorization mechanism. If a user-security-verification routine (USVR) exists, it is not

invoked for RACF-defined data sets. See "User-Security-Verification Routine (USVR)" on page 37 for more information.

To have password protection take effect for a data set, the catalog containing it must be RACF protected or password protected and the data set itself not defined to RACF.

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

Generic Profile-Checking Facility

RACF provides a *generic profile-checking facility*. With the always-call capability of integrated catalog facility catalogs, you can consolidate the access authorization requirements of several similarly named and similarly used data sets under a single generic profile definition. A generic profile is used to protect a single cluster or a group of clusters that require similar access authority. For example, you could build a generic profile with a high-level qualifier of *userid.**.

VSAM data sets that are generically protected are not RACF indicated in the catalog. Therefore, RACF is always called for any access to data sets cataloged in integrated catalog facility catalogs whether or not the data set is RACF defined or password protected. If the data set is not protected by either a discrete profile or a generic profile, password protection is in effect. The integrated catalog facility catalog does not have to be RACF protected in order for its data sets to be RACF protected.

For clusters cataloged in an integrated catalog facility catalog, a generic profile will be used to verify access to the entire cluster or to any of its components. Discrete profiles for the individual components may exist, but **only** the cluster's profile (generic or discrete) will be used to protect the components in the cluster. (**Note**: Profiles defined by Automated Data Set Protection (ADSP) processing during a data set define operation will be cluster profiles only.)

Data sets protected with discrete profiles are flagged as "RACF indicated." If a data set protected by a discrete profile is moved to a system where RACF is not installed, no user will be given authority to access the data set. However, if the data set is protected with a generic profile, it is not flagged as "RACF indicated"; therefore, access authority is determined by normal VSAM password protection.

RACF Authorization Checking

RACF authorization checking is generally compatible with password authorization checking. The compatibility includes the time the authorization check is made and the sources of authorization. The RACF authorization levels of alter, control, update, and read correspond to the password levels of master, control, update, and read in VSAM.

Deleting any type of RACF-protected entry from a RACF-protected catalog requires alter-level authorization to the catalog or the entry being deleted.

Altering the passwords in a RACF-protected catalog entry requires RACF alter authority to the entry being altered, or the operations attribute. Alter authority to the catalog itself is not sufficient for this operation.

It is your responsibility to ensure RACF profiles are correct after using REPRO MERGECAT or CNVTCAT on a catalog that uses RACF profiles. If the target and source catalogs are on the same volume, the RACF profiles remain unchanged. If the target and source catalogs are on different volumes, the old RACF profile is deleted and a new profile is created for the user naming the job. For further information, see *RACF General Information Manual.*

RACF-Controlled ERASE Options

DELETE processing removes catalog and VTOC information; it also makes the associated DASD space available for a new allocation. By default, this process does not erase the data from the disk. Sensitive data should be erased (overwritten) before its space is made available.

You can use RACF commands to specify an ERASE or NOERASE attribute in generic or discrete profiles. When so specified, these attributes become default attributes for:

- DELETE processing of an alternate index or cluster cataloged in an integrated catalog facility catalog (see "DELETE ERASE|NOERASE" on page 47 for more information).
- Scratch and partial release processing of non-VSAM data sets (see *System—Data Administration* for more information).

Note: For VSAM objects cataloged in a VSAM catalog (where erasure is controlled, as before, by using DEFINE and DELETE commands), the RACF ERASE|NOERASE attributes are ignored.

For information about specifying and using the RACF ERASE|NOERASE attributes, see *RACF General Information Manual* and associated RACF publications.

User-Security-Verification Routine (USVR)

In addition to password protection, VSAM allows you to protect data by specifying a program to verify a user's authorization. Specific requirements of the user-security-verification routine are described under "User-Written Exit Routines" in *VSAM Administration Guide.* To use this routine, specify the name of the authorization routine you have written in the AUTHORIZATION parameter of the DEFINE or ALTER command.

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

IGG.CATLOCK Facility

Access to a catalog must be restricted during catalog recovery. If access is not restricted (by locking, by terminating user sessions, or another method), users may be able to update the catalog during recovery and create a data integrity exposure.

An integrated catalog facility catalog can be locked to restrict access. Attempts to access a locked catalog by unauthorized users will fail with a return code indicating that the catalog is temporarily unavailable. After recovery has been completed, recovery personnel must unlock the catalog so that normal operations can resume. This locking/unlocking procedure ensures that a catalog will not be accessed during

the recovery, and eliminates the need to terminate user sessions or subsystems oriented to an integrated catalog facility catalog in order to deny access to that catalog.

The IGG.CATLOCK facility, in conjunction with normal security checking, controls who may invoke the lock attribute and who may access the locked catalog. Only users who have authority to the IGG.CATLOCK facility can lock an existing unlocked catalog or a newly defined catalog, unlock an existing locked catalog, or access and repair a locked catalog. If an installation does not use RACF or a RACF-equivalent product, an MVS router exit must be supplied to authorize the user.

For RACF systems, establishing authority to the IGG.CATLOCK facility requires issuing a RACF RDEFINE command for the profile IGG.CATLOCK, which is of class type FACILITY. PERMIT commands must then be issued to give READ access to IGG.CATLOCK for each user to be authorized. The FACILITY resource class is new with RACF Version 1 Release 7, and installations that have earlier RACF releases must add the FACILITY class to the class descriptor table (CDT). Non-RACF systems must be capable of interpreting and processing the profile IGG.CATLOCK of class type FACILITY.

To ensure the integrity of integrated catalog facility catalogs, installations should review the number of users who have authority to the IGG.CATLOCK facility and restrict this access to as few individuals as possible. Because normal processing of catalogs does not require authority to the IGG.CATLOCK facility, most users can be limited to update access on their catalogs if they only need to define and delete objects in the catalog.

Authorized users will still be unable to do lock/unlock actions or repair a locked catalog unless they also meet all other security checking criteria established for the catalog. For RACF systems, locking an unlocked (existing) catalog or unlocking a locked catalog requires RACF ALTER authority. On non-RACF systems, the user will be required to meet any protection measures provided by the system beyond the IGG.CATLOCK authorization, such as password validation.

The actual locking and unlocking of a catalog is done with two parameters, LOCK and UNLOCK, and the access method services commands ALTER, DEFINE, and IMPORT. LOCK and UNLOCK apply only when the target catalog is an integrated catalog facility catalog.

ALTER LOCK locks an existing catalog. ALTER UNLOCK unlocks a catalog, and should be used as the last step of catalog recovery. DEFINE USERCATALOG|MASTERCATALOG LOCK and AND DEFINE USERCATALOG|MASTERCATALOG UNLOCK control the setting of the lock attribute for newly defined catalogs.

An example of how LOCK and UNLOCK are used during recovery of an integrated catalog facility catalog is provided in Chapter 5, "Recovering the BCS Using IMPORT" on page 76.

Using One Catalog As a Model for Another Catalog

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

You can override some of the model's attributes by explicitly specifying them with the DEFINE command. If you do not want to change or add any attributes, you need only supply the entry name of the object being defined and the MODEL parameter. When you define a catalog, you must also specify the catalog's volume and space information. When you define an alternate index or a path, you must also specify the object to which it is related.

The MODEL parameter is designed so identical data sets are easy to define. By explicitly specifying certain parameters (for example, protection attributes), you can override any attributes of the model. When you use the MODEL parameter, you should ensure that the job is not terminated because of allocation problems when you explicitly do any of the following:

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

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

- If MODEL is specified as a parameter of USERCATALOG, PAGESPACE, or PATH, the following steps occur:
  1. The attributes of the model are copied for the object being defined.
  2. Any attributes explicitly specified as parameters in the DEFINE command override those of the model.
- If MODEL is specified as a parameter of CLUSTER or ALTERNATEINDEX (at the cluster level) but is not specified as a subparameter of the DATA or INDEX parameter, the following steps occur:
  1. The attributes of the model are copied for the cluster or alternate index entry of the object being defined.
  2. Any attributes explicitly specified as parameters of CLUSTER or ALTERNATEINDEX override those of the model for the cluster or alternate index entry.
  3. The attributes of the model's data and index components are copied for the object's data and index components.

4. Attributes explicitly specified as parameters of CLUSTER or ALTERNATEINDEX are reproduced to the data and index components, overriding those of the model.
5. Attributes explicitly specified with subparameters of the DATA or INDEX parameters override the previous two steps.

- If MODEL is specified both as a subparameter of DATA or INDEX and as a parameter of CLUSTER or ALTERNATEINDEX, the following steps occur:
  1. The attributes of the CLUSTER or ALTERNATEINDEX model are copied for the cluster or alternate index entry of the defined object.
  2. Any attributes explicitly specified as parameters of CLUSTER or ALTERNATEINDEX override those of the model for the cluster or alternate index entry.
  3. Attributes explicitly specified as parameters of CLUSTER or ALTERNATEINDEX are reproduced to the object's data and index components.
  4. Attributes of the model specified with the MODEL subparameter of the DATA or INDEX parameters are copied, overriding the previous step.
  5. Attributes explicitly specified with the subparameters of the DATA or INDEX parameters are copied, overriding the previous two steps.

Selecting Index Options

Five options influence performance through the use of the index of a key-sequenced data set. Each option improves performance, but some require that you provide additional virtual storage or auxiliary storage space. The options are:

- Index-set records in virtual storage
- Size of index control interval
- Index and data set on separate volumes
- Replication of index records (REPL option)
- Sequence-set records adjacent to control areas (IMBED option)

Index-Set Records in Virtual Storage

To retrieve a record from a key-sequenced data set or to store a record in it using keyed access, VSAM needs to examine the index of that data set. Before your processing program begins to process the data set, it must specify the amount of virtual storage it is providing for VSAM to buffer index records. Enough space for one I/O buffer for index records is the minimum, but a serious performance problem would occur if an index record were continually deleted from virtual storage to make room for another and then retrieved again later when it is required. Ample space to buffer index records can improve performance by preventing this situation.

You ensure that index-set records will be in virtual storage by specifying enough virtual storage for index I/O buffers when you begin to process a key-sequenced data set. VSAM keeps as many index-set records in virtual storage as the space will hold. Whenever an index record must be retrieved to locate a data record, VSAM makes room for it by deleting from the space the index record that VSAM judges to be least useful under the circumstances then prevailing. This is generally the index record that belongs to the lowest index level or that has been used the least.

Size of the Index-Set Control Interval

The second option you might consider is ensuring that the index-set control interval is large enough to cover a full control area. Thus, the index-set control intervals might be larger than actually required to contain the pointers to the sequence-set level. However, this option also keeps to a minimum the number of index levels required, thereby reducing search time and improving performance. This option increases rotational delay and transfer time.

Index and Data on Separate Volumes

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

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

Replication of Index Records

You can specify that each index record be replicated (written on a track of a direct access volume as many times as it will fit). Replication reduces the time lost waiting for the index record to come around to be read (rotational delay). Average rotational delay is half the time it takes for the volume to complete one revolution. Replication of a record reduces this time; for example, if 10 copies of an index record fit on a track, average rotational delay is only 1/20th of the time it takes for the volume to complete one revolution.

On an IBM 3340, the time usually is reduced by 50%. On an IBM 3380, the time is reduced to 1/n, where n is the number of times the index is replicated on the track.

Because there are usually few control intervals in the index set, the cost in terms of direct access storage space is small. If the entire index set is not being held in storage and there is significant random processing, then replication is a good choice. If not, replication does very little. Because its cost is small and it is an attribute that cannot be altered, it may be desirable to choose this option.

Sequence-Set Records Adjacent to Control Areas

When the data set is defined, you can specify that the sequence-set index record for each control area is to be embedded on the first track of the control area. This reduces disk arm movement because it is not necessary to do separate seeks to locate both the sequence-set index record and the data record. One arm movement enables VSAM to retrieve or store both the index record and the contents of the control interval in which the data record is stored.

When the IMBED option is chosen, sequence-set records are replicated, regardless of whether you also chose the REPL option. This means that one track of each control area is used for sequence set records. In some situations, this may be too much space for index in relation to the data. For example, the space required for the sequence set is 1/12th of the data space on an IBM 3340, and 1/15th of the data space on an IBM 3380. IMBED must be specified explicitly to get the performance benefits of a replicated, embedded sequence set.

Index Option Summary

On an IBM 3340, place the index and data on separate volumes and do not replicate or embed them. Provide index buffer space to hold the entire index set plus one sequence set when doing random processing. If direct access storage space is not a problem, if index and data cannot be put on different volumes, or if index buffer space is not available, specify REPL IMBED for random processing.

On an IBM 3380, specify REPL IMBED and make certain that the allocation unit is in cylinders.

Using JCL and Dynamic Allocation (DEFINE)

Before access method services can process a data set or volume, the system allocates the data set or volume to the job. Usually, the data set or volume is allocated when the scheduler processes the job step job control language (JCL) DD statements. However, when a data set name or volume serial number is specified with the access method services command, but the data set name or volume has not been described with a DD statement, an attempt will be made to allocate the data set or volume dynamically. See "Dynamic Allocation" in *Access Method Services Reference*.

Dynamic allocation has special usability requirements. To ensure successful dynamic allocation, access method services recommends that a data set's volume(s) be mounted on the system, with the resident or reserved attribute, prior to job step initiation.

Any volume must be mounted whenever its VTOC or VVDS has to be consulted or modified, or a catalog is being defined. Usually, the volume is allocated to the system component when the scheduler processes your job control language (JCL) DD statements.

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

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

- UNIT= can specify a device type and unit count for any direct access storage device that is supported by your system.
- VOLUME=SER= indicates the serial number(s) of the volume(s) on which data sets are to be allocated. When more than one volume is indicated, all volumes must be of the same device type. If the device types are different, you must concatenate the DD statements.
- DISP=OLD is required because VSAM must allocate space to VSAM objects (that is, catalogs, clusters, alternate indexes, and page spaces).
- DEFER indicates that the system is not to mount the volume until the data set is opened.

Identifying the Catalog's Volume

When you define a catalog, you can use a JCL DD statement to identify and mount the volume on which the catalog is to reside. If a JCL DD statement is not used and the volume is mounted as permanently resident or reserved, an attempt is made to dynamically allocate the catalog's volume. If JCL is used, include a DD statement of the form:

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

The volume on which the catalog is to be allocated must be mounted.

Using Share Options

With an integrated catalog facility catalog you may specify SHAREOPTIONS (3 3) or (3 4) for a BCS. SHAREOPTIONS (3 4) is the default and allows the BCS to be shared.

If SHAREOPTIONS (3 3) is specified, the catalog cannot be shared, even if the BCS is on a shared DASD device. This performance option should only be specified if you are sure the BCS *will not be shared* across multiple systems. Specifying SHAREOPTIONS (3 3) and then sharing a BCS may result in unpredictable damage to the BCS.

It is possible to specify the number of concurrent requests into the BCS with a minimum of 2 and a maximum of 255. The number relates only to read requests, not to update requests.

The number of buffers specified applies to each processor. If a request is for UPDATE, an ENQ with exclusive control is issued on the BCS itself. However, looking at the type of data usually kept in the VVDS, it could be expected that the majority of accesses to the BCS will be read operations.

The SHAREOPTIONS for the VVDS is (3 4) and the number of concurrent requests will be variable because they are provided for dynamically.

If a GET for update with no length change is requested, the ENQ on the VVDS is shared, and on the control interval it is with exclusive control. However, if a length change is required, the ENQ on the VVDS is with exclusive control. A VSAM volume record (VVR) may not span control intervals. For further information about SHAREOPTIONS, see *VSAM Administration Guide*.

Using the Alternate Master Catalog

The alternate master catalog provides an easy way to back up the master catalog in case of damage to the master catalog. It also provides a simple method to convert a VSAM master catalog to an integrated catalog facility master catalog without the need for an additional system.

Several steps are necessary to generate an alternate master catalog including:

1. The new volume must be initialized with IPL text.
2. If volume IPL is desired, some of the system data sets must be copied to the new volume.
3. If copied, SYS1.PROCLIB and SYS1.VTAMLIB must be allocated under a different name, because they are in use.
4. After allocation, they may be renamed to the original names.
5. A new master catalog is defined.

6. New storage index and page data sets must be allocated and defined in the new catalog.
7. Then define all system data sets in the new catalog.
8. Initialize LOGREC, place the new catalog pointer in the nucleus, and update PARMLIB pointers. (It would be useful to list the VTOC and catalog to provide a system record.)

For a sample job stream used to create an alternate master catalog, see Appendix C, "Alternate Master Catalog Job Stream" on page 157.

The system operator may specify which SYSCATnn member of the SYS1.NUCLEUS data set is to be used to find the master catalog for the duration of the current IPL. The selected master catalog must already exist and have the necessary system data sets cataloged in it. (See *System Generation.*)

The master catalog should be an integrated catalog facility catalog. A bit in the SYSCATnn entry must be set on to indicate an integrated catalog facility catalog. Failure to set this bit will cause the IPL to fail. If the named catalog entry is not found, the IPL will fail.

The SYSCATnn member is specified during NIP (nucleus initialization program) time. The operator will receive a message on the system console asking for a 2-character reply:

```
IEA347A SPECIFY MASTER CATALOG PARAMETER
```

This 2-character reply will be appended to the name "SYSCAT" to form the member of SYS1.NUCLEUS that contains the master catalog information. If the reply is a null line, then the default member "SYSCATLG" will be used.

The old member "SYSCATLG" will be replaced with a new member of the same format, except that the name will be "SYSCATnn," where n is any EBCDIC character. Many unique members may be in SYS1.NUCLEUS but only one may be selected at IPL time. A re-IPL must be done to change master catalogs.

The catalog name parameter in SYSCATnn is optional when the pointer is to a VSAM catalog. Because the integrated catalog facility catalog is found under its own name and not in the 'Z9999994' entry in the VTOC, the pointer must contain the name of the catalog.

Altering Attributes of Entries in a Catalog

The ALTER command is used to alter attributes in catalog entries and to rename VSAM data sets or members of non-VSAM partitioned data sets. To alter an entry, you must supply its name and the attributes to be altered.

Altering an entry does not normally require that the entry's volume be mounted, because the entry's use of space and the availability of space in the volume's data spaces can be determined by examining the catalog. The entry's volume must be mounted whenever the volume's VTOC must be consulted or modified, such as when a VSAM component or non-VSAM data set is renamed. A JCL DD statement can be used to cause the data set or volume to be allocated. If a ddname is not specified, access method services will dynamically allocate one. The volume must be mounted as permanently resident or reserved.

Most attributes of the VSAM data set are associated with its data or index components. Certain attributes such as retention period, owner ID, and cluster protection are associated with the cluster entry. If you use the cluster name in the ALTER command, only the cluster name attributes are changed. If you use the cluster name in the ALTER command to alter any attribute not associated with the cluster entry, access method services terminates the ALTER request and issues an error message. The reverse is also true; if you specify a data or index component name and attempt to alter an attribute associated with a cluster entry, access method services terminates the ALTER command and issues an error message. You must specify the explicit data or index component name to alter any of the remaining attributes such as share options, buffer space, write check, data, or index protection. You should specify the data and index component names at DEFINE time to facilitate the use of the ALTER command. Otherwise, you have to use the long system-generated names. Refer to "ALTER" in *Access Method Services Reference* to determine which values or attributes you may alter for a particular entry type.

You cannot ALTER the ATTRIBUTE parameters of an integrated catalog facility catalog. All attributes are modified by specifying data and index names. If an attribute only exists at the cluster level, the catalog name is specified. If you alter the expiration date to an invalid date, such as '99999', access method services returns a condition code of '0', even though the expiration date remains unchanged.

Generic Names and ALTER

To alter entries with qualified names (for example, PAYROLL.74.MAY) you can identify the entry with its full name (all qualifiers) or with all qualifiers but one. The unspecified qualifier is indicated with an asterisk (for example, PAYROLL.*.MAY). You must always specify the first qualifier of a qualified entry name. This kind of shortened name, PAYROLL.*.MAY, is called a generic name. If you specify full entry name, only that entry is modified. If you specify all qualifiers of the entry name but one, the entries whose entry names match the supplied qualifiers are altered.

For example, when you specify PAYROLL.*, the entries that might be altered are the entry names that contain two qualifiers; the first qualifier is PAYROLL. When you specify PAYROLL.*.MAY, the entries that might be altered are the entry names that contain three qualifiers; the first is PAYROLL and the third and last are MAY.

When you identify a catalog (by specifying the ALTER command's CATALOG parameter), access method services searches only the specified catalog for generically named entries. If no catalog is specified, the catalog is selected as indicated in "Order of Catalog Use: ALTER" in *Access Method Services Reference* and is searched for generic name entries.

You must identify all qualifiers in the qualified name. For example, PAYROLL.* cannot be used to identify a qualified name that contains three or more qualifiers, even though its first qualifier is PAYROLL. PAYROLL.81.MAY.* cannot be used to identify a qualified name that has more or fewer than four qualifiers.

Renaming Generically Named Entries

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

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

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

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

Deleting Catalogs and Indexes

The delete command should be used carefully. DELETE data set, DELETE CLUSTER, and DELETE USERCATALOG may affect data sets, the BCS or VVDS, or catalogs in different ways, depending on what other parameters are specified. This section on the DELETE command, as well as the DELETE command in *Access Method Services Reference*, should be reviewed carefully before using the DELETE command.

The following types of entries or objects may be deleted:

- Alias
- Alternate index
- Cluster
- Generation data group
- Non-VSAM data set
- Page space
- Path
- True name
- User catalog
- VVDS
- VVR

DELETE ALTERNATEINDEX

If a DELETE ALTERNATEINDEX is interrupted, the "delete-in-progress" bit in the cluster component is left on. However, DELETE itself ignores the bit and can be executed again after the reason for the failure has been corrected. LISTCAT may be used to check if the bit is on.

DELETE ERASE|NOERASE

DELETE processing removes catalog and VTOC information; it also makes the associated DASD space available for a new allocation. Sensitive data should be erased (overwritten) before its space is made available.

The erasure of data can be controlled by ERASE|NOERASE specifications in DEFINE and DELETE commands and in RACF profiles. See "Resource Access Control Facility (RACF)" on page 35 for further information.

DELETE ERASE can be used with an alternate index or cluster to request that the data component be erased before it is deleted. The volume must be mounted and the NOSCRATCH parameter cannot be used. DELETE ERASE is the default action when DEFINE ERASE has been specified or (for VSAM objects cataloged in an integrated catalog facility catalog) when RACF ERASE applies.

DELETE NOERASE can be used to override an ERASE attribute specified at DEFINE time; however, it does not override an associated RACF ERASE attribute. If RACF ERASE applies and NOERASE is desired, a RACF command must be used to change the attribute before the DELETE command is issued.

DELETE FORCE|NOFORCE

DELETE FORCE is used with the catalog or a GDG. A DELETE FORCE for a catalog deletes from the master catalog the catalog's self-describing VVRs and DSCBs, all its aliases, and the catalog's connector. The VVRs and DSCBs of any VSAM data sets cataloged in that catalog are deleted. Non-VSAM data sets are uncataloged when the catalog is deleted. The master password of the catalog must be supplied. An active master catalog cannot be deleted. If the FILE parameter is specified, it should point to all volumes containing VSAM data sets cataloged in the BCS.

Using DELETE FORCE for GDGs deletes the aliases of all the GDSs and the BCS entry for the GDG. The GDSs become uncataloged when the GDG is deleted.

DELETE NOFORCE is used with the catalog or GDGs and is the default. It deletes an empty BCS that only contains entries for itself and the VVR for this BCS. The master password of the catalog must also be supplied for DELETE NOFORCE. For empty GDGs, the BCS entry for the GDG is deleted.

DELETE PURGE|NOPURGE

DELETE PURGE is used for data sets when an entry is to be deleted even if the retention period has not expired. This parameter only relates to the retention period and does not interact with any other parameter.

DELETE NOPURGE is the default. An entry is not to be deleted if the retention period has not expired.

If DELETE PURGE/NOPURGE is used for objects that do not have an expiration date associated with them (for example, VVRs, aliases, and non-VSAM data sets), the PURGE/NOPURGE parameter is ignored, and the object is deleted.

DELETE RECOVERY|NORECOVERY

DELETE RECOVERY is used with a VVDS or a catalog. For catalogs, it deletes the BCS, its self-describing VVRs, and VTOC entries. The VVRs and DSCBs for the objects defined in the catalog are not deleted. The master password of the master catalog must be supplied. If the VVDS is damaged, it is unusable and the VSAM data sets must be deleted, redefined, and loaded. This command is used to remove a damaged VVDS before rebuilding it. The master password of the master catalog must be given. There should not be any VVRs for data sets cataloged in the master catalog in the VVDS that will be deleted.

DELETE NORECOVERY is the default. It processes the entry as described by the other parameters that are specified.

DELETE SCRATCH|NOSCRATCH

DELETE SCRATCH is used for VSAM and non-VSAM data sets. For any VSAM data set that is not a VVDS, the DSCBs, VVR, and BCS entries are deleted. For a VVDS, the DSCB and the entry in the specified BCS are deleted if the VVDS is empty. The entries for the VVDS in any other BCSs remain intact. The VVDS must be empty or the command will fail. You must identify the catalog of the entry to be deleted with a JOBCAT or STEPCAT DD statement, the CATALOG parameter, or the first qualifier in the entry's name. For non-VSAM data sets, the DSCBs and BCS entries are deleted. SCRATCH is the default for DELETE.

DELETE NOSCRATCH is used for VSAM and non-VSAM data sets. For non-VSAM data sets, the DSCBs are not deleted when the BCS entries are deleted. For any VSAM data set that is not a VVDS, the DSCBs and VVRs are not deleted when the BCS entry is deleted. For a VVDS, the BCS entry and the back pointer to the BCS in the VVDS's VVCR are deleted. The VVDS must be on a mounted volume. The delete will fail if the VVDS contains any VVRs for data sets cataloged in the specified BCS. NOSCRATCH cannot be used with ERASE or USERCATALOG.

DELETE TRUENAME|ALIAS

DELETE TRUENAME allows you to remove a named catalog entry so that the name may be used in a subsequent DEFINE. The true name entry is deleted only if its associated base record is missing or inaccessible. This can be determined by using the DIAGNOSE command, or by getting duplicate name errors for a DEFINE when the sphere record does not exist in the BCS.

DELETE ALIAS deletes an alias catalog record for a user catalog or for a non-VSAM data set.

The catalog and data set volumes must be mounted when:

- A data space, unique component, or user catalog is to be deleted.
- A cluster or alternate index is to be erased or scratched.
- A non-VSAM data set is to be scratched.
- A VVDS is to be deleted.
- A VVR is to be deleted.
- A catalog is to be deleted with the FORCE parameter specified.

FILE Parameter

If the FILE parameter is not specified and access is required to a volume or volumes during the deleting process, an attempt is made to dynamically allocate the volumes. A STEPCAT is required for dynamic allocation. See "Dynamic Allocation" in *Access Method Services Reference*.

DELETE deallocates only the last DDNAME that was dynamically allocated.

Examples

This set of examples illustrate how to define and delete integrated catalog facility user catalogs. It is assumed that the integrated catalog facility system catalog that exists is security protected at the update-password, control-password, and master-password levels.

Example 1: Define a User Catalog

In this example, an integrated catalog facility user catalog is defined. The master catalog is password protected. The user catalog in this example is also password protected. The volume VSER02 can be referenced by one VSAM catalog and up to 36 integrated catalog facility catalogs.

```
//DEFCAT   JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *

  DEFINE USERCATALOG -
         (NAME(USERCAT1) -
          ICFCATALOG -
          MASTERPW(UCATMRPW) -
          UPDATEPW(UCATUPPW) -
          FOR(365) -
          CYLINDERS(15 5) -
          VOLUMES(VSER02)) -
       DATA( -
          CYLINDERS(3 1) ) -
       INDEX( -
          CYLINDERS(1 ) ) -
       CATALOG(ICFMAST1/MASTMPW1)

  DEFINE ALIAS -
         (NAME(D40) -
          RELATE(USERCAT1)) -
          CATALOG(ICFMAST1/MASTMPW1)

  DEFINE ALIAS -
         (NAME(D50) -
          RELATE(USERCAT2)) -
          CATALOG(ICFMAST1/MASTMPW1)
/*
```

Explanation of Commands

- The USERCATALOG parameter is required and NAME specifies the user catalog being defined.
- ICFCATALOG specifies that the type of catalog to be defined is integrated catalog facility.
- The MASTERPW parameter specifies the master password for this catalog.
- The UPDATEPW parameter specifies the update password of this catalog.
- The FOR parameter specifies the retention period for this file, in this case, one year.
- The CYLINDERS parameter specifies the amount of space to be allocated to the catalog's data space. A space parameter is required.
- The VOLUMES parameter is required and specifies the volume containing this catalog. Access method services will dynamically allocate the catalog's volume; the volume should be mounted permanently RESIDENT or RESERVED to ensure successful dynamic allocation.
- The DATA parameter specifies the amount of space to be allocated to the catalog's data component.
- 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 DATA and INDEX and determines the appropriate proportions for each component.
- The CYLINDERS parameter specifies the amount of space to be allocated to the catalog's INDEX component.
- Because the master catalog is password protected, the CATALOG parameter is required. 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 User Catalog With a Model

This example defines an integrated catalog facility user catalog by using the previously defined user catalog, USERCAT1, as a model.

This example depends on the successful completion of example 1 for the user catalog, USERCAT1.

```
//DEFCAT2  JOB  ...
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *

  DEFINE USERCATALOG -
       (NAME(USERCAT2) -
       ICFCATALOG -
       MODEL(USERCAT1/UCATMRPW USERCAT1) -
       VOLUME(VSER03) -
       CYLINDER(15 5)) -
       CATALOG(ICFMAST1/MASTMPW1)

/*
```

Explanation of Commands

The DEFINE command defines a user catalog on volume VSER03. The catalog's attributes are modeled from a previously-defined user catalog, USERCAT1. The VOLUME and CYLINDERS parameters are required parameters, even though a model catalog is used.

- The USERCATALOG parameter is required and NAME specifies the name of the user catalog, USERCAT2.
- ICFCATALOG specifies that the type of catalog to be defined is integrated catalog facility.
- The MODEL parameter names the object to be used as a model and its master password, USERCAT1/UCATMRPW. The next field names the catalog that contains the model object's catalog entries. In this case, USERCAT1's self-describing entries describe the object model, USERCAT1.
- The VOLUME parameter is required and names the volume that is to contain the user catalog. Access method services will dynamically allocate the catalog's volume; the volume should be mounted permanently RESIDENT or RESERVED to ensure successful dynamic allocation.
- The CYLINDERS parameter specifies the primary and secondary allocation amounts (of cylinders) to be allocated to the catalog. The space is allocated to data and index components of the catalog as if it was not modeled, because the space specified overrides that of the model.
- The CATALOG parameter is required, because the master catalog is password protected. It names the master catalog, ICFMAST1, and specifies its master password. When a user catalog is defined, a user catalog connector entry is built and written into the integrated catalog facility master catalog. The user catalog connector entry enables the user catalog's volume to be located through the master catalog.

The defines of the alias names permit references to the respective user catalogs by the alias names. Alias names for integrated catalog facility catalogs also provide a way by which data set names can be used to direct the search for catalog information about data sets. Some of the following examples show how data set naming conventions are used to simplify JCL specification of JOBCAT and STEPCAT DD statements.

- The ALIAS parameter is required and NAME specifies the alias name of the user catalog.
- The RELATE parameter specifies the catalog for which the alias is defined.
- The CATALOG parameter supplies the update or master password of the master catalog, ICFMAST1.

Example 3: Delete a Catalog and Its Cataloged Objects

This example deletes all entries defined in the previous examples and deletes the user catalogs defined. Integrated catalog facility catalogs are assumed.

```
//DELETE   JOB   ...
//JOBCAT   DD    DSN=USERCAT1,DISP=OLD
//         DD    DSN=USERCAT2,DISP=OLD
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *

  /* DELETE THE NONVSAM DATA SET EXAMPLE.NONVSAM1 */

  DELETE  EXAMPLE.NONVSAM1 -
          NOSCRATCH -
          NONVSAM -
          CATALOG(USERCAT1/UCATUPPW)

  /* DELETE VSAM DATA SETS CATALOGED IN */
  /* USER CATALOGS USERCAT1 AND USERCAT2 */

  DELETE  (D50.EXAMPLE.ESDS1/ESD1PWMR -
          D50.EXAMNEW.KSDS1/KSD1PSWD -
          D40.MYDATA) -
          PURGE -
          CLUSTER

  /* DELETE THE ENTRY-SEQUENCED VSAM */
  /*   DATA SET EXAMPLE.ESDS2        */

  DELETE  EXAMPLE.ESDS2/ESD2MRPW -
          PURGE -
          CATALOG(USERCAT2)

  /* DELETE THE KEY-SEQUENCED VSAM */
  /*   CLUSTER EXAMPLE.KSDS2       */

  DELETE  EXAMPLE.KSDS2 -
          CATALOG(ICFMAST1/MASTMPW1)

  /* DELETE THE RELATIVE-RECORD VSAM */
  /*   CLUSTER EXAMPLE.RRDS1 */

  DELETE  EXAMPLE.RRDS1 -
          CLUSTER -
          CATALOG(USERCAT2)

  /* DELETE THE USER CATALOG USERCAT1 */

  DELETE  USERCAT1/UCATMRPW -
          USERCATALOG -
          PURGE

  /* DELETE THE USER CATALOG USERCAT2 */

  DELETE  USERCAT2/UCATMRPW -
          USERCATALOG -
          PURGE
/*
```

Explanation of Commands

The first DELETE command deletes the non-VSAM data set named EXAMPLE.NONVSAM1.

- The name of the non-VSAM data set is required.
- The NOSCRATCH parameter specifies that the VTOC entry of the object being deleted is not to be removed.
- The NONVSAM parameter ensures that the entry being deleted is a non-VSAM data set.
- The CATALOG parameter is required in this example to identify the catalog that describes the data set to be deleted. The catalog's update password is required to delete a non-VSAM data set.

The second DELETE command deletes VSAM data sets cataloged in two integrated catalog facility user catalogs, USERCAT1 and USERCAT2.

- The names of the VSAM data sets are specified; in addition, the master passwords of two protected data sets are required. Because the JOBCAT DD statement was provided, the CATALOG parameter is not required to direct the catalog search.
- The PURGE parameter causes the entries to be deleted without regard for the retention period.
- The CLUSTER parameter ensures that the catalog object being deleted is a VSAM data set.

The third DELETE command deletes the entry-sequenced VSAM data set EXAMPLE.ESDS2 from the user catalog, USERCAT2.

- The name of the entry-sequenced data set, EXAMPLE.ESD2, and its master password, ESD2MRPW, are required to identify the object to be deleted.
- The PURGE parameter causes the entry to be deleted without regard to its retention period.
- The CATALOG parameter is provided in this example to identify the catalog that describes the data set to be deleted.

The fourth DELETE command deletes the key-sequenced VSAM data set EXAMPLE.KSDS2 from the master catalog. Because this data set is a base cluster, its related alternate index, EXAMPLE.AIX, and its path, EXAMPLE.PATH, are also automatically deleted from the master catalog.

- The name of the key-sequenced VSAM data set, EXAMPLE.KSDS2, is required to identify the object to be deleted.

- The CATALOG parameter is required, because the delete process will result in more than one protected object being deleted. It names the catalog that contains the catalog entry of each object to be deleted and supplies the catalog's master password, which allows each object in the catalog to be deleted.

The fifth DELETE command deletes the relative record data set EXAMPLE.RRDS1 from the master catalog.

- The name of the relative record data set, EXAMPLE.RRDS1, is required to identify the object to be deleted. Because the data set was created without passwords, no password is required to delete it.
- The CLUSTER parameter specifies that a cluster is being deleted.

The sixth DELETE command deletes the integrated catalog facility user catalog USERCAT1.

- The name of the catalog and its master password are required to identify the object to be deleted.
- The USERCATALOG parameter is required to specify that a user catalog is being deleted.
- The PURGE parameter causes the entry to be deleted without regard to its retention period.

The seventh DELETE command deletes the integrated catalog facility user catalog USERCAT2.

- The name of the user catalog and its master password are required.
- The USERCATALOG parameter is required to specify that the object being deleted is a user catalog.
- The PURGE parameter causes the entry to be deleted without regard for the retention period.

Chapter 4. Converting to Integrated Catalog Facility Catalogs

Installations may find it desirable to convert a VSAM user catalog, a VSAM master catalog, or an OS/VS system catalog (OS CVOL) to an integrated catalog facility catalog. This chapter describes the various conversion processes and planning requirements needed for each type of catalog conversion and provides examples of their use.

Converting VSAM Catalogs

Converting VSAM catalogs to integrated catalog facility catalogs requires choosing a suitable conversion strategy, planning for such a conversion, and then converting a catalog, verifying the conversion, and reorganizing an integrated catalog facility catalog after conversion.

Choosing a Conversion Technique

You may consider three conversion techniques when planning the conversion of a VSAM catalog to an integrated catalog facility catalog:

- Converting one catalog at a time
- Converting one volume at a time
- Converting on an application, user, or data set basis

Converting One Catalog at a Time

The simplest method of converting existing catalogs is to use the access method services CNVTCAT command to convert one catalog at a time; it should be used whenever possible. To use the CNVTCAT command, certain prerequisites must be met. (See "Planning for Conversion Using CNVTCAT" on page 56.) In some cases, it is necessary to do some preparatory work to meet these prerequisites. The entire VSAM catalog and all its owned volumes are converted by executing the CNVTCAT command. In a CNVTCAT conversion, entries are built in the target integrated catalog facility catalog to reflect the existing VSAM and non-VSAM data set entries. Existing data sets are not moved or renamed as part of the CNVTCAT conversion process.

Converting One Volume at a Time

If each VSAM catalog controls only the volume it resides on, use the procedure described above for converting one catalog at a time.

If the VSAM catalog is recoverable, use EXPORTRA ALL followed by IMPORTRA to import the volume's data sets into the target integrated catalog facility catalog.

If a nonrecoverable VSAM catalog controls more than one volume, conversion on a volume-by-volume basis can be achieved only by explicitly exporting each data set from the VSAM catalog and then importing them into the target integrated catalog facility catalog.

Converting on an Application, User, or Data Set Basis

This approach uses an explicit EXPORT and IMPORT command sequence for each data set used by that application or user.

Planning for Conversion Using CNVTCAT

The CNVTCAT command provides a way to convert VSAM catalog entries to integrated catalog facility catalog entries with a basic catalog structure (BCS) and one or more VSAM volume data sets (VVDSs). The source VSAM catalog is deleted in the process, and new entries are built in the predefined target catalog. Data sets are not moved during the conversion.

Results to Expect From CNVTCAT

A CNVTCAT conversion run results in the following:

- VSAM catalog entries are converted to their equivalent types in the integrated catalog facility target catalog. Entries are processed in ascending collating sequence on a logical sphere basis, matching corresponding entries from the source catalog; for example, a cluster, an alternate index, and paths. As each base object is found in the source catalog and defined in the target catalog, its associated objects are also defined in the target catalog.
- All VSAM data spaces defined in the source catalog are deleted from the VTOC of the appropriate volume.
- The data space containing a VSAM catalog is freed. Thus, at the end of the conversion process, the VSAM catalog will no longer exist.
- The VSAM-ownership bit in the Format-4 DSCB of the VTOC of all owned volumes is turned off.
- All suballocated VSAM data sets defined in the source catalog have Format-1 DSCBs built. The name in each DSCB is the name of the data or index component as defined in the source catalog.
- The integrated catalog facility-owned catalog indicator is set on in the OPTCD field of the Format-1 DSCB for each data set in the integrated catalog facility catalog.
- VVR entries are added to the VVDS for each component converted.
- If no VVDS exists on a volume owned by the source catalog, one is defined; it is allocated before any space is deleted on the volume.
- An entry for each VVDS is added to the target catalog.
- The VVCR in each VVDS is updated to show a connection to the target catalog.

Because the master catalog is unchanged after CNVTCAT processing, alias entries associated with the source catalog name are left in place. To start using the integrated catalog facility catalog, delete the aliases that formerly directed the catalog search to the VSAM catalog and set up new aliases to point to the integrated catalog facility catalog.

DASD Space Requirements

Because the source VSAM catalog and the target integrated catalog facility catalog coexist during the CNVTCAT process, additional DASD space is required during the conversion.

In planning for DASD space allocation for the BCS and the VVDS, see the section "Estimating Space" in Chapter 3, "Defining, Altering, and Deleting a Catalog" on page 21. You should consider the following advice in planning for a target integrated catalog facility catalog:

- There must be sufficient contiguous free space in a single extent for a VVDS on each of the volumes owned by the converted catalog. The VVDS must not be allowed to extend during the conversion process.
- There must be sufficient space for the BCS, which may be placed on any volume. The BCS must be defined before running CNVTCAT.
- The space required for the BCS is usually about 25% of the space capacity required for a similar VSAM catalog; remember, both catalogs must exist simultaneously if CNVTCAT is to be used.
- There must be sufficient free space in the VTOC of each volume owned by the catalog being converted to allow for new DSCBs that will be required.

  Each suballocated VSAM data set requires additional Format-1 (and possibly Format-3) DSCBs in the VTOC for each of its components following conversion. The BCS and each VVDS will also require DSCBs. The total requirement is reduced by the number of existing DSCBs used for the VSAM data spaces that are released during conversion.

Anticipating Problems

The CNVTCAT process will fail for any of the following reasons:

- There is not enough space in the VTOC of a volume to contain the DSCBs created by CNVTCAT.
- There is not enough space on the volume to contain the VVDS.
- Volumes owned by the source catalog are not available to CNVTCAT.
- The source catalog is damaged.
- The data sets whose entries are in the catalog to be converted are in use during the conversion.

Some preparation may be required before using CNVTCAT. When planning for conversion, you should use access method services to list the source catalog, and IEHLIST to list the VTOCs of all volumes owned by the source catalog. Use these listings to determine whether any of the listed conditions exist that will cause a conversion to fail. The following considerations describe some conditions that will cause CNVTCAT to fail. Included in these considerations are procedures you may use to remove these conditions.

Full Volumes: When a catalog owns a VSAM volume with no free space for a VVDS and you cannot move non-VSAM or VSAM UNIQUE data sets to another volume, all VSAM data sets in a VSAM data space can be exported; then the VSAM data space can be deleted, providing space for the VVDS. Because the format of the portable data set is unchanged in the integrated catalog facility environment, a data set exported from a VSAM catalog can be imported into an integrated catalog facility catalog.

The entire contents of a volume owned by a VSAM recoverable catalog can be exported by using EXPORTRA ALL and later can be imported into an integrated catalog facility catalog. All VSAM objects on that volume, including the catalog recovery area (CRA), should be deleted before the import operation. Because the integrated catalog facility catalog does not have a catalog recovery area, the space previously occupied by the CRA should provide enough space for the VVDS.

Full Volumes with Some UNIQUE Data Sets: When a VSAM volume includes one or more data sets that are defined with the UNIQUE parameter, export enough data sets with the PERMANENT option to release space for a VVDS. The remainder of the volume can then be converted using the CNVTCAT command, and the exported data set can be imported into the target catalog. In some cases, the amount of space required for the VVDS may prevent the exported data set from being reimported to its original volume. In such cases, there is no alternative but to import the data set to another volume.

There must be enough contiguous free space for the VVDS to be contained in a single extent during conversion; the VVDS must not be allowed to extend during the conversion.

Full Volumes with Some Non-VSAM Data Sets: When a VSAM volume includes one or more non-VSAM data sets, move enough data sets to another volume to release space for a VVDS. The remainder of the volume can then be converted using the CNVTCAT command, and the data sets can be moved back to the original volume. In some cases, the amount of space required for the VVDS may prevent the data sets from being moved back to the original volume. In such cases, there is no alternative but to leave the data sets on another volume. A combination of non-VSAM and VSAM UNIQUE data sets may be moved/exported to satisfy VVDS space requirements.

There must be enough contiguous free space for the VVDS to be contained in a single extent during conversion; the VVDS must not be allowed to extend during the conversion.

Full VTOC: If the VTOC on a volume has insufficient free space to accommodate the extra DSCBs, it is necessary to rebuild the volume with a larger VTOC. VSAM data sets on such a volume should be exported with the PERMANENT option. Non-VSAM data sets should be dumped using functions provided by Data Facility Data Set Services (DFDSS) or a similar function. If there is a VSAM catalog on the volume that owns other volumes, it should be unloaded using REPRO. A larger VTOC on the volume can then be built using functions provided by Device Support Facilities and the non-VSAM data sets restored. If a VSAM catalog was unloaded, it should be reloaded using REPRO and converted with CNVTCAT; otherwise, an integrated catalog facility catalog should be defined and the exported data sets then imported into the integrated catalog facility catalog to complete the conversion.

A listing of the VSAM catalog (via the LISTCAT command) can be used to determine the number of extents occupied on each volume by the data sets.

Use the following guidelines to estimate the number of additional DSCBs required when converting suballocated VSAM data sets:

- Each component of each suballocated cluster requires a Format-1 DSCB; the Format-1 DSCB defines up to 3 extents for the component.
- A Format-3 DSCB is required for each 13 extents beyond the first 3 extents defined by the Format-1 DSCB.
- If the catalog being converted is a recoverable catalog, the DSCB describing the CRA will be freed.
- A Format-1 DSCB is required to define the VVDS.
- The DSCB describing the data space will be freed during the conversion.

Unavailable Unused Volumes, Nonrecoverable Catalog: If a nonrecoverable catalog owns a volume that no longer exists, an attempt at conversion will fail when CNVTCAT processing requests that the volume be mounted. Therefore, references to that volume and all data sets on that volume must be removed before running CNVTCAT. VSAM objects may be removed from the catalog by using:

```
DELETE entryname CLUSTER NOSCRATCH
```

The volume entry may then be removed with:

```
DELETE volser SPACE FORCE NOSCRATCH
```

Unavailable Unused Volumes, Recoverable Catalog: If a volume owned by a recoverable catalog is not available, the objects cataloged on that volume and the volume entry for the volume must be removed before converting the catalog. The procedure for removing references to the objects and to the volume from a recoverable catalog differs from the procedure used on a nonrecoverable catalog. The following procedure may be used to remove unavailable objects from the catalog before conversion:

1. Export the catalog (with the DISCONNECT option) that references the available volume(s).
2. Define a user catalog on a different volume with the same attributes (RECOVERABLE) and name as the disconnected catalog.
3. Execute a RESETCATALOG command on the newly created catalog. The DD statement specified by the CRAFILES parameter of the

RESETCATALOG command should include all volumes, except the missing volumes, owned by the original catalog.

Special Considerations for Multiple Device Types: If the source catalog owns volumes of different device types and you want the volumes premounted, the FILE parameter of the CNVTCAT command is required. The operand of the FILE parameter specifies the ddname for the volumes owned by the source catalog as a concatenated DD statement. For example:

```
//  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//DD1 DD VOL=SER=INDEX1,UNIT=3350,DISP=OLD
//    DD VOL=SER=DATA01,UNIT=3380,DISP=OLD
//SYSIN    DD *
  CNVTCAT INDATASET(VSAM.CAT)      -
          OUTDATASET(ICF.CAT)      -
          FILE(DD1)
```

In the example, the VSAM catalog, VSAM.CAT, is converted to an integrated catalog facility catalog. The source catalog owns a 3350 volume, INDEX1, and a 3380 volume, DATA01. The DD statement, DD1, specified as the operand of the CNVTCAT FILE parameter, references these volumes as a concatenated DD statement.

Special Considerations for Removable Volumes: If a VSAM catalog owns many removable volumes, and there are not enough spindles on which to mount all of them at the same time for CNVTCAT processing, then CNVTCAT cannot be used to convert the catalog. When this condition exists, the catalog can be converted by exporting all objects owned by the catalog and importing them into the target integrated catalog facility catalog.

Damaged Source Catalog: If catalog damage prevents conversion of a VSAM catalog by using CNVTCAT, evaluate the damage and repair the catalog before proceeding with conversion. If the source catalog cannot be repaired, you should export accessible objects cataloged in the damaged catalog and then import them into the integrated catalog facility catalog.

Backup of Volumes Owned by VSAM Catalog before Conversion

During conversion processing, the CNVTCAT command deletes all VSAM data spaces and makes the source catalog unusable. As the conversion proceeds, integrated catalog facility catalog entries are built and DSCBs are created for the data sets that were suballocated. The conversion passes through a vulnerable period when data sets are not yet represented in the VTOC but these data spaces have been scratched. If CNVTCAT fails during this vulnerable phase, the data sets are lost and conversion cannot be restarted. Similar considerations apply to the volume containing the VSAM catalog to be converted. After conversion has started for the catalog volume, the VSAM catalog is no longer accessible, because its VTOC DSCB is removed.

All these considerations make it necessary to back up all volumes owned by the catalog being converted.

Converting the Catalog

Because the CNVTCAT process deletes DSCBs for VSAM data spaces before defining DSCBs for the converted objects, the conversion should be done in a single initiator.

Following CNVTCAT processing, the user catalog connector for the source catalog is unchanged, and aliases for the source catalog are left in place. To complete the conversion, remove the user catalog connector from the source catalog by using EXPORT DISCONNECT; this also removes the aliases. Then redefine the aliases for the target integrated catalog facility catalog.

With many alias entries, as in some TSO installations in which each userid is an alias of a catalog, the generation of DEFINE ALIAS commands to rebuild these alias entries may be automated. Use an online editor to process a LISTCAT output listing of the source catalog before conversion.

Verifying the Conversion

The conversion of the catalog can be checked in the following ways:

- Inspecting the printed output from CNVTCAT

  CNVTCAT issues messages to indicate the start and successful completion of conversion for each entry in the source catalog. It also issues a message as each volume's VSAM data spaces are about to be deleted.

- Inspecting the printed output from EXPORT/IMPORT

  If IMPORT is used for all or part of the conversion process, it gives the date and time the exported copy was made. Verify this against the EXPORT listing to ensure that an earlier copy of the data was not imported.

- Listing the VTOC of the converted volume

  The LISTVTOC FORMAT printed output from IEHLIST shows that successful conversion is completed in accordance with the following:

  - The source catalog's data space entry is scratched.
  - The data and index components of data sets that were formerly suballocated are in the VTOC.
  - Converted data set entries have OPTCD=X'80'. The integrated catalog facility catalog entries have OPTCD=X'C0'.

  The VSAM-ownership bit in the Format-4 DSCB is turned off. This can be checked by running a LISTVTOC DUMP or an AMASPZAP VERIFY on the DSCB.

- Listing the target catalog

  The target catalog shows all the data set entries that were in the source catalog. Owned volumes have a VVDS entry instead of a volume record. The catalog cluster name always shows as 44 bytes of zeros, reflecting the catalog's self-describing record key of binary zero. The catalog cluster record always appears first in the catalog listing.

- Running DIAGNOSE on the BCS and its connected VVDSs

  DIAGNOSE allows the BCS and each of its VVDSs to be checked against each other for consistency. No entries should be flagged as an error.

If the LIST option is selected, the name and type of each entry in the BCS and VVDS are listed.

Rerunning a Failed CNVTCAT Conversion Process

If CNVTCAT fails, execute the following steps before rerunning the conversion:

- Delete the target integrated catalog facility catalog with the FORCE option.
- Restore the volume containing the source catalog and all volumes owned by that catalog.
- Correct the condition that caused CNVTCAT to fail.

Reorganizing the Catalog after Conversion

Before conversion, the target BCS contains:

- Its own sphere record
- The BCS data and index records
- The VVDS sphere records for the volume containing the BCS and for other volumes containing preallocated VVDSs.

As each object in the source catalog is converted, records are inserted into the target BCS. Apart from true name records with generated names (that is, names prefixed with VSAMDSET), these insertions take place in approximately ascending collating sequence; as a result, the BCS may have several control interval and control area splits. Because the splitting process leaves both the control intervals and control areas half full, it is possible to produce a BCS that is only one-quarter full.

Listing the cluster entry for the BCS following a conversion will show whether control interval or control area splits have occurred. To improve BCS space utilization and remove the fragmentation caused by control interval and control area splits, the BCS is usually reorganized following conversion. This is done by exporting the BCS and then importing it. The processing done by IMPORT results in the requested FREESPACE parameter (specified in the DEFINE command) amounts being preserved.

Converting the Master Catalog

The considerations for converting a VSAM catalog to an integrated catalog facility catalog also apply to converting a VSAM master catalog. There is, however, an additional restriction that an active master catalog cannot be converted if it is currently in use by the system.

One approach to converting a VSAM master catalog is to use a temporary, or alternate, master catalog while the VSAM master catalog is being converted. A pointer to this alternate master catalog is put in a SYS1.NUCLEUS member, SYSCATnn. At IPL time, the operator specifies SYSCATnn, and the alternate catalog is used as the master catalog. Any existing SYSCATnn member may be used, including the default member SYSCATLG, or a new SYSCATnn member can be created. For further information on SYS1.NUCLEUS, see *System Generation*; for further information on SYSCATnn, see "Using the Alternate Master Catalog" on page 43.

The temporary (alternate) master catalog must be built by the installation, and needs to contain only system data set entries and connectors to the source and target

catalogs. The page and swap data sets defined in the temporary catalog can be of a minimum size necessary to run the conversion, and should be independent of the catalog being converted.

Note that a procedure can be used to generate a fully operational alternate VSAM or integrated catalog facility master catalog for the system. New page space data sets for the alternate master catalog must be allocated because the previously used page space data sets cannot be utilized. When the VSAM master and alternate catalogs are functioning correctly, the VSAM master catalog may be converted. For the alternate master catalog generation procedure, see Appendix C, "Alternate Master Catalog Job Stream" on page 157.

An alternative approach is to use a system that does not use the master catalog being converted, as for example:

- An alternate system that can access the volume that contains the catalog.
- An operating system that can be restored to the processor where the conversion is being performed. The operating system should have its own VSAM or integrated catalog facility master catalog.

Converting OS CVOLS

The following considerations pertain to OS CVOLs:

Converting an OS CVOL to an integrated catalog facility catalog includes constructing an integrated catalog facility catalog that is equivalent to an OS CVOL, and updating the alias entries to point to the new catalog. Use the access method services CNVTCAT command to build an integrated catalog facility catalog to contain all the entries of the OS CVOL. The source catalog is unchanged by the conversion, so there is no special requirement to back up the OS CVOL. However, if the CVOL contains GDGs, the generation data sets need to be backed up. The target catalog does not need to be empty and OS CVOLs can be merged during conversion; otherwise, each OS CVOL can be converted to an integrated catalog facility catalog, and the integrated catalog facility catalog is then tested for function and performance before performing any reorganization. The access method services REPRO command may then be used to split or merge parts of different integrated catalog facility catalogs.

WARNING: CNVTCAT should be run only *once* against any particular OS CVOL so that generation data sets are not inadvertently scratched. If it becomes necessary to rerun the CNVTCAT, then all GDG entries in the integrated catalog facility catalog should be removed by using the command DELETE NOSCRATCH.

When many OS CVOLs are converted, or when there is a constraint on CSA space, it may be advisable to minimize the number of integrated catalog facility catalogs to avoid too large a CSA requirement. For details on CSA space requirements, see *Data Facility Product: Planning Guide.*

No automatic method is provided for altering the alias entries in the master catalog to point to the integrated catalog facility catalog. All aliases must be deleted (either individually or by deleting the SYSCTLG.Vvolser non-VSAM entry, which includes all its aliases), and new aliases must be defined for the integrated catalog facility catalog.

If the number of alias entries is large, as in some TSO installations where each userid is an alias of a catalog, the generation of DEFINE ALIAS commands to rebuild these alias entries may be automated. Use an online editor to process a LISTCAT output listing of the catalog before conversion, or use a simple program that performs a similar function.

Where an OS CVOL catalog contains entries for a GDG data set, a model DSCB for new GDG members normally resides on the same volume as the catalog. If the catalog volume is changed during conversion, care must be taken to identify and move these model DSCBs so that they reside on the volume containing the integrated catalog facility catalog.

Following the conversion, revise your procedures to allow for the integrated catalog facility catalog. Using the corresponding access method services functions, revise job streams that used IEHLIST, IEHPROGM, or IEHMOVE to manage the OS CVOL.

After the integrated catalog facility catalog is working correctly, the SYSCTLG data set and its master catalog entry can be deleted.

To reduce the space requirements of the newly formed BCS, refer to "Reorganizing the Catalog after Conversion" on page 62.

Converting Application Programs

No application employing standard VSAM macro calls should require modification following the conversion to integrated catalog facility catalogs.

Test the following types of programs in the integrated catalog facility catalog environment to determine whether they operate correctly:

- Programs that issue SVC 26
- Programs that use CATLG and CAMLST

Programs that read catalogs as data sets need careful attention. If the functions performed by these programs are still required in the integrated catalog facility catalog environment, the programs must be changed to allow for new catalog record formats. Because catalog information resides in both the BCS and the VVDS, such programs may require substantial rewriting. The access method services PRINT command can be used to display the contents of BCS and VVDS records to show typical layouts.

Revising Installation Procedures

A particular user installation's existing backup and recovery procedures may or may not handle the integrated catalog facility environment. Users of VSAM recoverable catalogs may find that their existing procedures may no longer work. A new backup and recovery procedure should be designed and tested before using the integrated catalog facility catalog.

Production job streams may contain catalog names in STEPCAT or JOBCAT DD statements. If any catalogs had their names changed, these job streams must be revised.

Production job streams or TSO CLISTs may contain access method services commands. Most of these commands function normally without alteration. However, the DEFINE SPACE, EXPORTRA, LISTCRA, and RESETCAT commands perform differently or do not apply to the integrated catalog facility environment. (For more information about these access method services commands, see *Access Method Services Reference*.) If catalog names are included in commands contained in production job streams, these may need to be revised.

VSAM data set suballocation of data space does not exist in an integrated catalog facility catalog. The UNIQUE and SUBALLOCATION parameters of the DEFINE commands will be ignored. All objects will be defined UNIQUE, and space requests for unique allocations (for all integrated catalog facility catalog allocations) are not rounded to a cylinder as they are with VSAM catalogs. Thus, applications that have used more than the requested space in VSAM catalogs may fail for lack of space when data sets are deleted and redefined for integrated catalog facility catalogs. If allocation is given in tracks, they will be in contiguous tracks. In addition, secondary extents of such data sets on the first volume are not released when they are opened for reuse. Secondary extents on any other volume than the first volume will be released.

The volume-oriented commands EXPORTRA, IMPORTRA, and LISTCRA are not available with integrated catalog facility catalogs. If users have relied on these commands to back up their data sets, then the backup procedures must be changed.

If an installation has used VSAM volume ownership to restrict VSAM allocations by volume, these procedures must be changed.

The procedures for creating catalogs must be changed because the default definition is for VSAM catalogs; furthermore, default parameters of the DEFINE command may not be appropriate for performance objectives. Catalogs that are used as models for other catalogs must also be redefined.

Sample Procedures for Catalog Conversion

In the examples that follow, the catalog structure environment described below is assumed.

- A VSAM master catalog named AMASTCAT with master password MASTMPW1 exists on volume PAGEVS.
- A VSAM recoverable catalog named VSAM.CAT1 connected to the master catalog is on a 3380 volume labeled 338001. This catalog owns, in addition to its own volume, a 3380 volume labeled 338002. LTH00 is defined as an alias for this catalog.

  Volume 338001 contains the following cataloged data sets:

  - LTH00.KSDS is a key-sequenced suballocated data set.
  - LTH00.ESDS is an entry-sequenced suballocated data set.
  - LTH00.UNIQ is a key-sequenced unique data set.
  - LTH00.OSDS is a non-VSAM data set.
  - LTH00.BIGKSDS forms part of a multivolume suballocated key-sequenced data set. The second volume is 338002.

  Volume 338002 contains the following cataloged data sets:

  - LTH00.KS02 is a key-sequenced suballocated data set.
  - LTH00.ES02 is an entry-sequenced suballocated data set.
  - LTH00.UN02 is a key-sequenced unique data set.
  - LTH00.OS02 is a non-VSAM data set.
  - LTH00.BIGKSDS forms part of the multivolume suballocated key-sequenced data set.

  A nonrecoverable VSAM catalog named VSAM.CAT2 connected to the master catalog is on a 3380 volume labeled 338004. This catalog owns, in addition to its own volume, a 3350 volume labeled 335001. SLC00 is defined as an alias for this catalog.

  Volume 338004 contains the following cataloged data sets:

  - SLC00.MVKSDS.D is a data component of a suballocated key-sequenced data set. The index component is on volume 335001.
  - SLC00.KSDS is a key-sequenced suballocated data set.
  - SLC00.ESDS is an entry-sequenced suballocated data set.

  Volume 335001 contains the following cataloged data sets:

  - SLC00.MVKSDS.I is the index component of the suballocated key-sequenced data set whose data component is on volume 338004.
  - SLC00.KSDS50 is a suballocated key-sequenced data set.
  - SLC00.OSDS is a non-VSAM data set.

  A 3380 volume labeled 338005 is an OS/CVOL. WLB00 and BAT00 are aliases for this catalog. This volume contains two cataloged data sets:

  - BAT00.DATA is a sequential data set.

– WLB00.TEST is a sequential data set.

A 3380 volume labeled 338003 is an empty volume. This volume is not owned by either VSAM catalog.

Converting a Recoverable Catalog

In the following example, a VSAM catalog, VSAM.CAT1, is converted to an integrated catalog facility catalog, ICF.CAT1.

```
//CNVTCAT JOB ,
//CNVTCAT1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *

  DEFINE  USERCATALOG                -
          (NAME(ICF.CAT1)            -
          CYLINDERS(2,1)             -
          VOLUME(338001)             -
          ICFCATALOG)

  CNVTCAT INDATASET(VSAM.CAT1)       -
          CATALOG(ICF.CAT1)

  EXPORT  VSAM.CAT1 DISCONNECT

  DEFINE  ALIAS                      -
          (NAME(LTH00)               -
          RELATE(ICF.CAT1))          -
          CATALOG(AMASTCAT)
```

In the above example:

- EXEC PGM = IDCAMS invokes access method services.
- SYSPRINT defines message output for access method services.
- DEFINE USERCATALOG defines the new integrated catalog facility catalog with the name of ICF.CAT1 into the master catalog. The integrated catalog facility user catalog is on volume 338001.
- CNVTCAT command converts VSAM.CAT1 into the above-defined integrated catalog facility catalog ICF.CAT1.
- EXPORT DISCONNECT disconnects the VSAM catalog VSAM.CAT1 from the master catalog. The aliases for VSAM.CAT1 are deleted.
- DEFINE ALIAS defines the aliases formerly associated with the VSAM.CAT1 with the integrated catalog facility catalog ICF.CAT1.

Converting a Nonrecoverable Catalog

In the following example, a nonrecoverable catalog, VSAM.CAT2, is converted to an integrated catalog facility catalog.

```
//CNVT  JOB ,
//CNVTCAT2 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//DD1 DD VOL=SER=338004,UNIT=3380,DISP=OLD
//    DD VOL=SER=335001,UNIT=3350,DISP=OLD
//SYSIN DD *

  DEFINE  USERCATALOG                -
          (NAME(ICF.CAT2)            -
          CYLINDERS(2,1)             -
          VOLUME(338004)             -
          ICFCATALOG)

  CNVTCAT INDATASET(VSAM.CAT2)       -
          FILE(DD1)                  -
          CATALOG(ICF.CAT2)

  EXPORT  VSAM.CAT2 DISCONNECT

  DEFINE  ALIAS                      -
          (NAME(SLC00)               -
          RELATE(ICF.CAT2))          -
          CATALOG(AMASTCAT)
```

In the above example:

- EXEC PGM = IDCAMS invokes access method services.
- SYSPRINT defines message output for access method services.
- DD1 defines the devices and volumes owned by the source catalog, VSAM.CAT2. The access method services command CNVTCAT will reference this DD statement with the FILE parameter.
- DEFINE USERCATALOG defines the new integrated catalog facility catalog, ICF.CAT2, in the master catalog. The catalog is on volume 338004.
- CNVTCAT converts the source catalog, VSAM.CAT2, into the target integrated catalog facility catalog, ICF.CAT2.
- EXPORT DISCONNECT removes the catalog connector and aliases for VSAM.CAT2 from the master catalog.
- DEFINE ALIAS defines the alias SLC00 for the converted catalog ICF.CAT2.

Converting an OS/CVOL

In the following example, an OS/CVOL on volume 338005 is converted to an integrated catalog facility catalog, ICF.CVOL.

```
//CNVTCVOL EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//DD1 DD DSN=SYSCTLG,VOL=SER=338005,UNIT=3380,DISP=OLD
//SYSIN DD *

  DEFINE  USERCATALOG              -
          (NAME(ICF.CVOL)          -
          CYLINDERS(1,1)           -
          VOLUME(338005)           -
          ICFCATALOG)

  CNVTCAT INFILE(DD1)              -
          CATALOG(ICF.CVOL)

  DELETE  (SYSCTLG.V338005)        -
          NONVSAM                  -
          NOSCRATCH

  DEFINE  ALIAS                    -
          (NAME(WLB00)             -
          RELATE(ICF.CVOL))

  DEFINE  ALIAS                    -
          (NAME(BAT00)             -
          RELATE(ICF.CVOL))
```

In the above example:

- EXEC PGM = IDCAMS invokes access method services.
- SYSPRINT defines message output for access method services.
- DD1 defines the data set SYSCTLG on volume 338005. The FILE parameter of the access method services CNVTCAT command references the DD statement.
- DEFINE USERCATALOG defines an integrated catalog facility catalog, ICF.CVOL, in the master catalog. The new catalog is on volume 338005.
- CNVTCAT converts the OS CVOL into the integrated catalog facility catalog, ICF.CVOL.
- DELETE NONVSAM uncatalogs the OS CVOL from the master catalog and deletes the associated aliases; the OS CVOL itself is not deleted.
- DEFINE ALIAS (two statements) defines the aliases formerly associated with the OS CVOL for the new integrated catalog facility catalog, ICF.CVOL.

Removing an Unavailable Volume from a Nonrecoverable Catalog

In the following example, volume 335001 has become unavailable for conversion using CNVTCAT. Before running CNVTCAT to convert the catalog VSAM.CAT2, the job stream in the example is run to remove objects on the missing volume from the catalog.

```
//RESET  JOB ,
//NONRECOV EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *

  DELETE (SLC00.MVKSDS)            -
         CLUSTER                   -
         NOSCRATCH                 -
         CATALOG(VSAM.CAT2)

  DELETE (SLC00.KSDS50)            -
         CLUSTER                   -
         NOSCRATCH                 -
         CATALOG(VSAM.CAT2)

  DELETE (SLC00.OSDS)              -
         NONVSAM                   -
         NOSCRATCH                 -
         CATALOG(VSAM.CAT2)

  DELETE (335001)                  -
         SPACE                     -
         NOSCRATCH                 -
         CATALOG(VSAM.CAT2)
```

In the above example:

- DELETE CLUSTER (two statements) deletes the VSAM objects in the catalog VSAM.CAT2. The NOSCRATCH operand specifies that actual reference to the volume containing the cataloged object should not take place.
- DELETE NONVSAM deletes a non-VSAM data set from the catalog.
- DELETE SPACE deletes catalog reference to the VSAM data space on the missing volume. The NOSCRATCH operand specifies that reference to the volume containing the space should not take place.

Removing an Unavailable Volume from a Recoverable Catalog

In the following example, volume 338002 has become unavailable for conversion. The recoverable catalog VSAM.CAT1 is rebuilt using RESETCAT.

```
//RESET  JOB ,
//DISC   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSIN DD *

  EXPORT  VSAM.CAT1 DISCONNECT

  DEFINE  USERCATALOG              -
          (NAME(VSAM.CAT1)         -
          CYLINDERS(2,1)           -
          VOLUME(338003)           -
          RECOVERABLE              -
          VSAMCATALOG)             -
          CATALOG(AMASTCAT)

//RESET  EXEC PGM=IDCAMS
//STEPCAT DD DSN=VSAM.CAT1,DISP=OLD
//        DD DSN=VSAM.CAT2,DISP=OLD
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//DD1 DD UNIT=3380,VOL=SER=338001,DISP=OLD
//IDCUT1 DD DSN=WORKFILE,UNIT=3380,
//    VOL=SER=338004,DISP=OLD,AMP='AMORG'
//SYSIN DD *

  RESETCAT CATALOG(VSAM.CAT1)      -
           CRAFILES(DD1)           -
           WORKCAT(VSAM.CAT2)      -
           MASTERPW(MASTMPW1)

  LISTCAT  VOLUME                  -
           CATALOG(VSAM.CAT1)
/*
```

In the above example:

- EXEC PGM = IDCAMS invokes access method services.
- SYSPRINT defines the message output of access method services.
- EXPORT DISCONNECT disconnects catalog VSAM.CAT1 from the master catalog. The aliases for VSAM.CAT1 are also removed.
- DEFINE USERCATALOG defines a VSAM catalog with the same name and recoverable attribute, as the catalog that has just been disconnected. The new VSAM.CAT1 is on a different volume (338003).
- In job step RESET, a RESETCATALOG operation is performed. For requirements on resetting a recoverable catalog, see *VSAM Catalog Administration.*
- RESETCAT re-creates the catalog VSAM.CAT1 in the newly defined catalog. The CRAFILES operand references a DD statement defining all volumes to be used to reconstruct the catalog. The missing volume, 338002 is not referenced on the DD statement DD1.

Converting a Master Catalog

This example assumes that an alternate master catalog named ALTMAST has been built on volume ALTCAT. This catalog has been specified as the active master catalog at IPL time. The job stream shows the access method services commands used to build the target integrated catalog facility master catalog named IMASTCAT on volume NEWCAT, and the access method services commands used to convert the source VSAM catalog AMASTCAT (which is the regular master catalog for this system) on volume PAGEVS to the new catalog.

```
//MASTCNVT JOB ,
//*
//* CREATE THE TARGET MASTER CATALOG
//*
//MASTBLD  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *

  DEFINE  MASTERCATALOG             -
          (NAME(IMASTCAT)           -
          CYLINDERS(10,1)           -
          VOLUME(NEWCAT)            -
          ICFCATALOG)

//*
//* CONVERT THE TARGET MASTER CATALOG
//*
//MASTCNVT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *

 /* MAKE THE SOURCE CATALOG ACCESSIBLE FOR CONVERSION */
  IMPORT  CONNECT                   -
          OBJECTS((AMASTCAT         -
          DEVICETYPE(3380)          -
          VOLUMES(PAGEVS)))         -
          CATALOG(ALTMAST)

 /* CONVERT THE CATALOG */
  CNVTCAT INDATASET(AMASTCAT)       -
          OUTDATASET(IMASTCAT)
```

Chapter 5. Backing Up and Recovering Catalogs

The following three items need to be considered in any procedure for backing up and recovering integrated catalog facility catalogs:

- Basic catalog structure (BCS)
- VSAM volume data set (VVDS)
- Volume table of contents (VTOC) and its index, if an index exists

A Backup and Recovery Strategy

Considerations

The particular backup and recovery procedure defined for an installation must be tailored to meet the needs of that system. Each installation must determine its need for volume backup, the value of quick recovery, the required frequency of backups, and so on.

The pros and cons of volume backup as opposed to data set backup are still relevant in the integrated catalog facility environment, and this section will not attempt to discuss all those issues. However, integrated catalog facility catalogs introduce a new issue: any access to a VSAM data set requires the VVDS, and the VVDS can be backed up only by a volume backup. This implies that any backup strategy must include volume backup to protect the VVDS. There is an alternative, however, and that is to rebuild the VVDS instead of trying to recover it from a backup copy. In some cases, this rebuilding approach will have definite advantages.

The items below should be considered for the backup and recovery of the BCS, the VVDS, and the VTOC.

- Basic catalog structure (BCS)

  Use the EXPORT and IMPORT commands to back up and recover the BCS, even if volume backup is used. You will have greater flexibility and may avoid the need to restore the total volume in order to recover a BCS. The BCS and any of its aliases are copied by EXPORT, and IMPORT can recover the aliases, if they are needed. The BCS can be imported to a different volume, and even to a different device type.

  Use the IGG.CATLOCK facility (described in Chapter 3) if it is available. Catalog locking prevents access to an integrated catalog facility catalog that is being recovered; it does this without terminating user sessions or subsystems. When the IGG.CATLOCK facility is used, attempts to access a locked catalog fail with a return code indicating that the catalog is temporarily unavailable. After recovery, the integrated catalog facility automatically reorients the users or subsystems to the recovered catalog.

- VSAM volume data set (VVDS)

  There is no access method services command to back up the VVDS, so volume backup is the only way to create a backup copy. However, if volume backup is already being planned as the backup procedure for an installation, the VVDS will automatically be captured with each volume backup. Data Facility Data Set Services (DFDSS) can be used to do the volume backup and recovery. See "Data Facility Data Set Services (DFDSS)" on page 96 for more information.

The alternative to volume backups is rebuilding the VVDS. If EXPORT/IMPORT is used to back up data sets and the failure is limited to some VVRs, recovery may require importing only the relevant data sets. Otherwise, the VVDS is rebuilt by importing every VSAM data set that resides on the VVDS volume. This complete rebuild may be time consuming, but it will ensure that the VVDS is accurate and current.

- Volume table of contents (VTOC)

  As with the VVDS, no access method services commands can be used to back up and recover the VTOC. However, backup and recovery of the VTOC can be accomplished with volume backup and volume recovery via DFDSS DUMP and RESTORE.

  WARNING: Although it is possible to copy the VTOC tracks, and to repair a damaged VTOC by other means, the best way to ensure the integrity of the VTOC is to do a full volume backup and restore.

Restrictions

For a catalog to be a candidate for enhanced integrated catalog facility catalog recovery (which uses catalog locking), no job may be allocated to the catalog with a disposition of OLD. This disposition of OLD could come from JOBCAT or STEPCAT statements or DD cards, or it could come from dynamically allocating the catalog as a data set. This dynamic allocation can result from using OUTDATASET with most access method services commands. Specifying a disposition of SHR on a DD card and then referring to the DD card with the OUTFILE keyword bypasses this dynamic allocation.

Conclusions

When developing a backup and recovery procedure for integrated catalog facility catalogs, you should plan to:

- Use EXPORT/IMPORT for the BCS.
- Use the IGG.CATLOCK facility to restrict catalog access.
- Use volume backup and recovery for the VTOC.
- Recover the VVDS by rebuilding it, if volume backup and recovery is inadequate or unfeasible.

Backing Up a Catalog

In this section, backing up a catalog refers only to backing up the BCS.

The VTOC and the VVDS cannot be processed with the EXPORT, IMPORT, or REPRO commands. If they could be processed, "out-of-sync" conditions might occur between high-use RBAs or extent information kept in the VVDS and the VSAM data sets to which they apply. For information on backing up the VVDS, see "Other Backup and Recovery Facilities" on page 96.

Backing Up the BCS Using EXPORT

The EXPORT command can be used to create a backup copy of the catalog, known as the portable data set. The BCS and any of its aliases are copied. If EXPORT is used to back up the BCS, the IMPORT command must be used to restore it.

The TEMPORARY option should be specified with EXPORT. If PERMANENT is specified, a warning message is issued and the BCS is exported with the TEMPORARY option. See "Using the Alternate Master Catalog" on page 43.

An exported backup copy cannot be imported into a master catalog. A master catalog must be referenced as a user catalog for any recovery action to take place.

To ensure the integrity of the copy, access to the BCS is serialized during EXPORT. This serialization prevents update access but allows read access.

Because the BCS is a key-sequenced data set, an integrated catalog facility catalog can be exported and imported to a volume with a different device type. The BCS is backed up, as any key-sequenced data set, but EXPORT of a BCS does not back up the corresponding entries in a related VVDS.

Backing Up the BCS Using REPRO

REPRO NOMERGECAT can be used to copy one BCS (the source) into another BCS (the backup). If REPRO is used to back up a BCS, REPRO must be used to recover it. The copying of a BCS to another BCS requires that:

- The source catalog was previously defined.
- The target catalog was previously defined.
- The target catalog was empty.

The BCS must be copied in its entirety. The VVDS pointers will point to the target catalog.

Access to the BCS is serialized during REPRO to ensure the integrity of the copy. This serialization prevents update access but allows read access.

REPRO does not support the unloading of a BCS to a sequential data set and then reloading that data set.

REPRO may be used as an alternative to EXPORT/IMPORT to reorganize, for example, space and sequence changed entries in a BCS or to back up a BCS. However, use of EXPORT/IMPORT requires that the target BCS containing the backup be kept.

If REPRO is used to reorganize or back up a BCS, both the target and source catalogs may be used to access the same data sets. The unwanted catalog should be removed by using DELETE RECOVERY USERCATALOG or EXPORT DISCONNECT.

Recovering a Catalog

In this section, recovering a catalog refers only to recovering the BCS.

The VTOC and VVDS cannot be processed by using IMPORT or REPRO. For information on recovering the VVDS, and for some additional detail on recovering the BCS, see "Other Backup and Recovery Facilities" on page 96 and "Recovery Procedure Scenarios" on page 97.

Recovering the BCS Using IMPORT

A BCS that was backed up by using the EXPORT command can be recovered by using the IMPORT command. If the BCS had any aliases, those were also copied by the EXPORT command, and can be recovered by specifying the ALIAS parameter with the IMPORT command. A simplified example of importing a BCS and its aliases is shown here, with the use of the IGG.CATLOCK facility (LOCK and UNLOCK) included. The IGG.CATLOCK facility was described in "Protecting the Catalog" on page 30.

- Assuming a RACF system, grant the user who will recover the catalog appropriate RACF authority to the portable data set (the backup) as well as to the catalog to be recovered.
- Grant this user authority to the IGG.CATLOCK facility.

```
//STEP1     EXEC  PGM=IKJEFT01,DYNAMNBR=20
//SYSUDUMP  DD    SYSOUT=A
//SYSTSPRT  DD    SYSOUT=A
//SYSTSIN   DD    *
  SETROPTS GENCMD(*)
  SETROPTS CLASSACT(*)
  RDEFINE FACILITY IGG.CATLOCK AUDIT(ALL) UACC(NONE)
  PERMIT  IGG.CATLOCK CLASS(FACILITY) ID(USERID1) ACC(READ)
/*
```

- Lock the catalog so that access is restricted.

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSABEND  DD    SYSOUT=A
//AMSDUMP   DD    SYSOUT=A
//SYSIN     DD    *
  ALTER CATALOG1/MASTMPW1-
        LOCK-
        CATALOG(CATALOG1/MASTMPW1)
/*
```

- Delete the (failing) BCS with the access method services command DELETE RECOVERY. Although the BCS is deleted, the objects cataloged in it are not scratched.

```
//STEP1     EXEC  PGM=IDCAMS
//CATVOL    DD    DISP=OLD,UNIT=3380,VOL=SER=338001
//SYSPRINT  DD    SYSOUT=A
//SYSABEND  DD    SYSOUT=A
//AMSDUMP   DD    SYSOUT=A
//SYSIN     DD    *
  DELETE CATALOG1/USERMPW1 -
         FILE(CATVOL)      -
         RECOVERY          -
         USERCATALOG
/*
```

- Import the portable data set with the BCS aliases by using the access method services command IMPORT ALIAS LOCK. If the aliases are not needed, do not specify the ALIAS parameter.

```
//STEP1     EXEC  PGM=IDCAMS
//RECEIVE   DD    DSN=CATBACK,DISP=(OLD,PASS,DELETE),
//            SPACE=(TRK,(5,1)),UNIT=3380,VOL=SER=3380001
//SYSPRINT  DD    SYSOUT=A
//SYSABEND  DD    SYSOUT=A
//AMSDUMP   DD    SYSOUT=A
//SYSIN     DD    *
  IMPORT OUTDATASET(CATALOG1/MASTMPW1) -
         ALIAS                         -
         LOCK                          -
         INFILE(RECEIVE)               -
         CATALOG(ICFMAST1/MASTMPW1)
/*
```

- Check for any recent changes.

  The backup or portable copy of an integrated catalog facility catalog reflects the contents of the exported catalog at the time of its export. Any subsequent define or delete operations are not reflected in the catalog when it is imported. The access method services command DIAGNOSE should be used after the import, to assess any activity not reflected by the imported catalog.

  Figure 25 on page 80 shows the activities that may have occurred since the last catalog backup, and the tasks required to bring the backup catalog up to the current level.

- Unlock the catalog with the command ALTER UNLOCK to allow general access.

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSABEND  DD    SYSOUT=A
//AMSDUMP   DD    SYSOUT=A
//SYSIN     DD    *
  ALTER CATALOG1/MASTMPW1 -
        UNLOCK            -
        CATALOG(CATALOG1/MASTMPW1)
/*
```

Recovering Shared Catalogs

When two or more systems share access to a user catalog, and the user catalog is recovered by one of those systems to a different volume or device type, or both, you will need to update or replace the catalog connector record in the master catalog on the other system(s). You can update the connector record in one of two ways: either by performing an IMPORT CONNECT operation specifying the ALIAS parameter or by performing an EXPORT DISCONNECT/IMPORT CONNECT sequence of operations.

Using IMPORT CONNECT ALIAS

Use the IMPORT CONNECT ALIAS operation to preserve the related alias associations, if any exist. Specify the changed volume serial and device type information to update the associated fields in the user catalog's connector record in the master catalog, and any existing aliases that are related to the user catalog will be preserved.

For example, the user catalog ICFUCAT1, which has many aliases, exists on volume V338001 (a 3380), and is shared by SYSTEMA and SYSTEMB. If ICFUCAT1 is successfully recovered by SYSTEMA to volume V338002 (another 3380), it is inaccessible to SYSTEMB because its catalog connector record in SYSTEMB's master catalog now has incorrect volume information. To update the catalog connector record in SYSTEMB's master catalog without losing any of the related aliases, execute the following IDCAMS step on SYSTEMB:

Reconnect the user catalog on the second system by using the access method services command IMPORT CONNECT ALIAS. Specifying the ALIAS parameter preserves the aliases.

```
//STEP1     EXEC   PGM=IDCAMS
//SYSPRINT  DD     SYSOUT=A
//SYSIN     DD     *
  IMPORT CONNECT ALIAS       -
    OBJECTS(ICFUCAT1         -
            DEVICETYPE(3380) -
            VOLUMES(V338002))
/*
```

Note: Aliases do not have to exist to use this method of update.

Using EXPORT DISCONNECT/IMPORT CONNECT

If no related aliases exist for the user catalog on the second system, or their loss is not a concern, an alternative way to update the catalog connector record in the master catalog would be to use the EXPORT DISCONNECT/IMPORT CONNECT sequence of operations. Execute the following IDCAMS steps on SYSTEMB to replace (DELETE and DEFINE) the catalog connector record in SYSTEMB's master catalog:

Disconnect the user catalog on the second system by using the access method services command EXPORT DISCONNECT, then reconnect the user catalog by using the access method services command IMPORT CONNECT.

```
//STEP1     EXEC   PGM=IDCAMS
//SYSPRINT  DD     SYSOUT=A
//SYSIN     DD     *
  EXPORT -
         ICFUCAT1 -
         DISCONNECT

  IMPORT CONNECT -
         ALIAS -
         OBJECTS(ICFUCAT1 -
                 DEVICETYPE(3380) -
                 VOLUMES(V338002))
/*
```

Note: Aliases will be lost with this method of update.

Recovering the BCS Using REPRO

REPRO can be used to copy a BCS into another BCS. Recovery is accomplished when the source BCS (to be copied) is actually the backup.

See "Backing Up the BCS Using REPRO" on page 75 for a description of the REPRO command. REPRO cannot be used to recover an exported BCS.

Activity Causing Downgrading	Data Set Type	Location of Information	Action Needed To Upgrade the Catalog
Add			
Records			
	N	VTOC	None
	V	VVDS	None
Extents			
	N	VTOC	None
	V	VTOC/VVDS	None
Volumes			
	N	BCS/VTOC	Recatalog to pick up volumes
	V	BCS/VTOC/VVDS	(and reuse VTOC, VVDS information)
Data Sets			
	N	BCS/VTOC	Recatalog to pick up volumes
	V	BCS/VTOC/VVDS	(and reuse VTOC, VVDS information)
Remove			
Records			
	N	VTOC	None
	V	VVDS	None
Extents			
	N	VTOC	None
	V	VTOC/VVDS	Not applicable to VSAM
Volumes			
	N	BCS/VTOC	Recatalog data set
	V	BCS/VTOC/VVDS	VSAM volumes are removed with ALTER
Data Sets			
	N	BCS/VTOC	DELETE NOSCRATCH to
	V	BCS/VTOC/VVDS	remove just BCS information
Modify			
Records			
	N	VTOC	None
	V	VVDS	None
Extents			
	N	VTOC	None
	V	VTOC/VVDS	Not applicable to VSAM
Volumes			
	N	BCS/VTOC	Recatalog data set
	V	BCS/VTOC/VVDS	Not applicable to VSAM
Data Sets			
	N	BCS/VTOC	Remove obsolete BCS
	V	BCS/VTOC/VVDS	information and recatalog data set

N = Non-VSAM data set
V = VSAM data set

Figure 25. Activities That Downgrade the Integrated Catalog Facility Catalog

Moving a Catalog

An integrated catalog facility catalog can be moved to another volume (this volume may be a different device type), by using the EXPORT and IMPORT commands. The catalog to be moved is first exported, and then the catalog is imported. The VOLUMES subparameter of the OBJECTS parameter is used on the IMPORT command to specify the target volume. If the target device type has different device characteristics, the new integrated catalog facility catalog may have new data and index component control interval sizes.

Recording EXPORT Information

Information about the exports of integrated catalog facility catalogs can be recorded via SMF type 36 records. The type 36 record identifies the catalog being exported, the time of export, and, if available, the information necessary to retrieve the portable data set and import it. This record contains the information required for recovering a catalog.

The system programmer will need to change the SMF parameters to include recording this new SMF record, which is written only upon successful completion of the EXPORT command. The SMFPRMxx member in the SYS1.PARMLIB contains the SYS parameter that selects the SMF records to be recorded.

The system programmer may then extend the SMF reporting program to include handling this new SMF record. The fields of this record can be mapped by the program using the IFASMFI6 macro and specifying the new SMF record type. The IFASMFI6 macro also handles other record types that record integrated catalog facility catalog updates and VVDS update activity. A reporting program can scan for type 36 records and sort those found by catalog name or by date, and then it can generate reports. These reports can be used to provide historical data about catalog exports, or to determine the backup copy for restoring an integrated catalog facility catalog.

For details about the SMF type 36 record, see *System Management Facilities*.

Problem Prevention

In this section, the emphasis is on the BCS and the VVDS, rather than on VSAM data sets. The following topics are covered:

- Early indication of errors
- Monitoring the usage of the BCS and VVDS
- Reorganization procedures

Early Indication of Errors

The regular use of DIAGNOSE with and without the COMPARE option provides an early indication of problems. If COMPARE is specified, consideration must be given to the potential benefit of early error discovery versus elapsed time and volume mounting requirements. DIAGNOSE should be used each time a BCS is copied.

LISTCAT ALL may also be used to identify some types of errors.

Monitoring Usage

Both the BCS and the VVDS should be monitored using LISTCAT on a regular basis. This allows the catalog administrator to be aware of the occurrence of excessive secondary extents or control interval and control area splits and to take remedial action if required.

Either REPRO or EXPORT and IMPORT may be used to reorganize a BCS.

BCS Reorganization Using EXPORT/IMPORT

Note: If there is a possibility that users will attempt to access the catalog while it is being reorganized, the IGG.CATLOCK facility (LOCK and UNLOCK) should be used to prevent unauthorized access. (See "IGG.CATLOCK Facility" on page 37.) "Recovering the BCS Using IMPORT" on page 76 has an example showing the use of the IGG.CATLOCK facility.

1. EXPORT the BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=STY.EXPORT,UNIT=SYSDA,VOL=SER=SPOOL1,
//              DISP=OLD
//SYSIN     DD    *
  EXPORT -
       ICFCAT1 -
       OUTFILE(DD1) -
       TEMPORARY
/*
```

2. DELETE the BCS with the RECOVERY option:

```
//STEP1     EXEC  PGM=IDCAMS
//DD1       DD    UNIT=SYSDA,VOL=SER=SPOOL1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DELETE -
       ICFCAT1 -
       FILE(DD1) -
       RECOVERY -
       USERCATALOG
/*
```

3. If the control area is less than 1 cylinder and an increase in size is not desired, then DEFINE the BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DEFINE -
       USERCATALOG -
       (NAME(ICFCAT1) -
       VOLUMES(SPOOL1) -
       TRACK(4 4) -
       RECORDSIZE(4086 4086) -
       ICFCATALOG)
/*
```

4. IMPORT the BCS with the INTOEMPTY option:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD2       DD    DSN=ICFCAT1,DISP=OLD,AMP=AMORG,
//              UNIT=SYSDA,VOL=SER=SPOOL1
//DD1       DD    DSN=STY.EXPORT,UNIT=SYSDA,VOL=SER=SPOOL1,
//              DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTFILE(DD2) -
       INTOEMPTY
/*
```

BCS Reorganization Using REPRO

Note: If there is a possibility that users will attempt to access the catalog while it is being reorganized, the IGG.CATLOCK facility (LOCK and UNLOCK) should be used to prevent unauthorized access. (See "IGG.CATLOCK Facility" on page 37.) "Recovering the BCS Using IMPORT" on page 76 has an example showing the use of the IGG.CATLOCK facility.

1. DEFINE the second BCS on the same volume:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DEFINE -
       USERCATALOG -
       (NAME(ICFCAT2) -
       VOLUMES(SPOOL1) -
       TRACK(3) -
       RECORDSIZE(4086 4086) -
       ICFCATALOG)
/*
```

2. REPRO the BCS to be reorganized into the second BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//          DD    DSN=ICFCAT2,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  REPRO -
       INDATASET(ICFCAT1) -
       OUTDATASET(ICFCAT2)
/*
```

3. DELETE the BCS to be reorganized with the RECOVERY option:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//DD1       DD    UNIT=SYSDA,VOL=SER=SPOOL1,DISP=OLD
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DELETE -
       ICFCAT1 -
       FILE(DD1) -
       RECOVERY -
       USERCATALOG
/*
```

4. DEFINE the new BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DEFINE -
        USERCATALOG -
        (NAME(ICFCAT1) -
        VOLUMES(SPOOL1) -
        TRACK(3) -
        RECORDSIZE(4086 4086) -
        ICFCATALOG)
/*
```

5. REPRO the second BCS into the new BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//          DD    DSN=ICFCAT2,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  REPRO -
        INDATASET(ICFCAT2) -
        OUTDATASET(ICFCAT1)
/*
```

6. DELETE the second BCS with the RECOVERY option:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT2,DISP=SHR
//DD1       DD    UNIT=SYSDA,VOL=SER=SPOOL1,DISP=OLD
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DELETE -
        ICFCAT2 -
        FILE(DD1) -
        RECOVERY -
        USERCATALOG
/*
```

VVDS Reorganization

A VVDS is an entry-sequenced data set. It is not subject to splits and the need to reorganize it only occurs as a result of excessive secondary extents. The VVDS only requires additional extents when the record to be added cannot fit into any existing control interval. The need for additional extents should be minimal.

For example, for a VVDS on an IBM 3350 Direct Access Storage with the default allocation of TRACKS(3 2):

- The primary allocation is 12 control intervals.
- If the average VVR length is 350 to 400 bytes, each control interval will contain 4K/400 = 10 VVRs.
- The primary allocation should hold 12 x 10 = 120 VVRs.

However, the first control interval is the VVCR and the second control interval is used only for the VVRs of the VVDS itself; 100 VVRs remain available for user components before a secondary allocation is required.

If the VVDS must be reorganized, for example, for performance reasons, then it must be rebuilt.

To accomplish this:

1. EXPORT each VSAM data set on the volume.
2. DELETE VVDS RECOVERY.
3. IMPORT each data set to rebuild the VVDS.

DIAGNOSE should be run to determine any differences between the rebuilt VVDS and each BCS that had data sets on the volume.

Selecting a Solution to a Problem

Figure 26 lists several tasks you may want to do in backup and recovery procedures. Each solution that is next to the problem is described in the sections that follow Figure 26.

Task	Solution
To remove an unrelated VVDS or the last BCS	"ALTER REMOVEVOLUMES" on page 86
To replace entries in the BCS	"DEFINE Data Set RECATALOG" on page 88
To replace entries for the VVDS into the BCS	"DEFINE VVDS RECATALOG" on page 89
To re-create entries for an alternate index, data, or index	"DEFINE ALTERNATEINDEX RECATALOG" on page 89
To re-create entries for cluster, data, or index	"DEFINE Cluster RECATALOG" on page 89
To re-create path entry	"DEFINE Path RECATALOG" on page 90
To use a data set name not accessible through its sphere record	"DELETE TRUENAME" on page 90
To delete a VVR	"DELETE VVR" on page 90
To delete a base cluster or GDG entries	"DELETE Data Set NOSCRATCH" on page 91
To delete an integrated catalog facility catalog and all its cataloged objects	"DELETE USERCATALOG FORCE" on page 92

Figure 26 (Part 1 of 2). Backup and Recovery Solutions

Task	Solution
To delete the BCS connector record and DSCB	"DELETE USERCATALOG RECOVERY" on page 92
To delete a VVDS entry and pointer in the VVCR	"DELETE VVDS NOSCRATCH" on page 93
To delete a VVDS and its VTOC entry	"DELETE VVDS RECOVERY" on page 93

Figure 26 (Part 2 of 2). Backup and Recovery Solutions

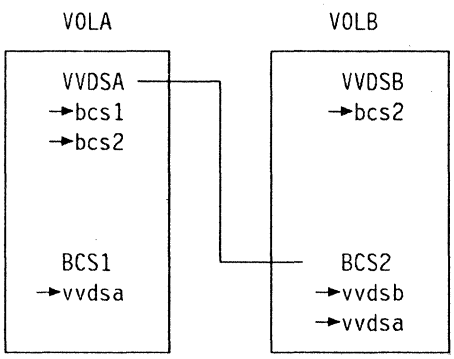
ALTER REMOVEVOLUMES

Note: Run DIAGNOSE before and after ALTER REMOVEVOLUMES.

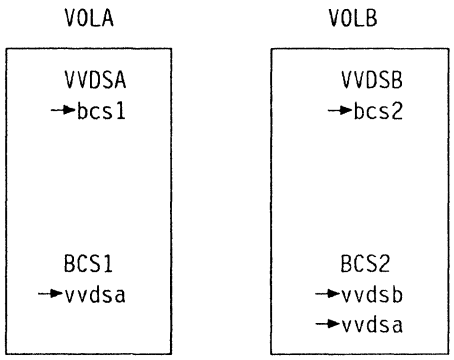
It is important to realize that ALTER REMOVEVOLUMES breaks the relationships between the VVDS and BCSs on other volumes. Assume that:

1. There are two volumes, VOLA and VOLB:
   - VOLA contains VVDSA and BCS1.
   - VVDSA is also known to BCS2 on VOLB.

   The normal relationships, as shown below, are:
   - Each VVDS contains:
     - VVCR entries for every related BCS.
     - VVR entries for each BCS on the VVDS volume (not explicitly shown).
   - Each BCS contains cluster entries for every related VVDS



2. If ALTER REMOVEVOLUMES is used to clean up VOLA, and VVDSA and BCS1 are subsequently redefined, the relationship between the new VVDS on VOLA and BCS2 on VOLB no longer exists.



Specifically:

a. BCS2 on VOLB still contains a cluster entry for VVDSA on VOLA.
b. The new VVDSA on VOLA does *not* contain the name of BCS2 in its VVCR.

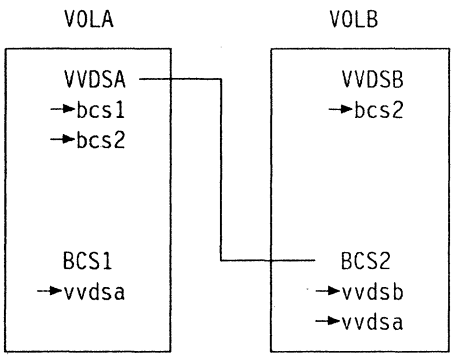
3. It is not possible to use DELETE VVDSA NOSCRATCH to remove the VVDSA cluster entry from BCS2. The DELETE fails with message IDC3009I 50-20, which indicates that the name of BCS2 was not found in the VVCR of VVDSA when it was expected.

   However, this is not a great problem. The first DEFINE into BCS2 for a data set on VOLA will cause message IDC3009I 28-30 to be issued. This message is the result of DEFINE:

   a. Not finding the name of BCS2 in the VVCR of VVDSA, but
   b. Finding a duplicate name when attempting to add the VVDSA cluster to BCS2.

   Distribution of the return codes between modules does not result in the actual error being made available to the last module. Therefore, a more appropriate error message cannot be issued.

   The DEFINE creates an entry for BCS2 in the VVCR of VVDSA and completes successfully. Subsequent DEFINEs do not generate the message, because the relationship between VVDSA and BCS2 has been reestablished.



It may be desirable to remove the VVDS and all BCSs from a volume. This must be done with some care because:

- It is impossible to delete a VVDS if it is not cataloged in a BCS that can be opened. Therefore, if all related BCSs have been deleted or, for some reason, cannot be opened, it is impossible to delete the VVDS.

- It is impossible to delete a BCS if it cannot be opened itself. If the VVDS on the volume has already been deleted, the BCS cannot be opened and it is impossible to delete it.

The problem seldom arises, because it is usually possible to relate the VVDS to a BCS on another volume for at least as long as it takes to delete the original BCS and then the VVDS. Even an uncataloged VVDS can be recataloged and deleted using the procedure in "DELETE VVDS RECOVERY" on page 93.

If it is ever necessary to remove the VVDS and the BCSs from a volume, and impossible to relate the VVDS to a BCS on another volume, even temporarily, ALTER REMOVEVOLUMES is required to remove either the VVDS or the last BCS.

Note: DIAGNOSE should be run before and after the use of ALTER REMOVEVOLUMES.

DEFINE Data Set RECATALOG

This command replaces entries in the BCS based upon information supplied within the command, and information that it finds in the VVR. Both the VVR and the VTOC entries must exist. The occasion to use this command may arise from:

- Using DELETE NOSCRATCH to delete an entry from a BCS
- Using IMPORT to introduce a down-level BCS
- Restoring a down-level volume

The following parameters may be required:

- NAME
- VOLUME
- SPACE
- Type of organization

Usually, only NAME and VOLUME are required. However:

- It may be necessary to specify SPACE even though the values specified were not used to override the values already in the VVR.
- RECATALOG executed with return code 0 when the organization was omitted, but it created entries in the BCS for a key-sequenced data set even when there was no VVR for an index component.

The required parameters are in addition to others that are kept in the BCS and should be respecified, such as ownership, passwords, and expiration date.

It is not necessary to specify component names, because they are obtained from the VVRs whether they were originally specified or generated by an integrated catalog facility catalog.

DEFINE VVDS RECATALOG

This command replaces entries for the VVDS into the BCS based upon:

- Details supplied with the command
- Details in the VVRs describing the VVDS found in the second control interval of the VVDS

Both VVR and VTOC entries must exist.

You use this command after redefining a BCS that contained data sets on other volumes. In this situation:

- The VVDSs on the other volumes know about the BCS.
- The BCS does not know about these VVDSs.

Specifying the names of the previously connected VVDSs defines them in the BCS.

Because a subsequent DEFINE of a data set on such a volume into the BCS also causes the cluster entry for the VVDS on the volume to be defined, it is not essential to perform this function.

DEFINE ALTERNATEINDEX RECATALOG

This command re-creates catalog entries for an alternate index, data, and index components. The VVDS and VTOC must exist, or the operation will fail. To RECATALOG requires that:

- The following parameters in the object's integrated catalog facility catalog entry must be respecified with RECATALOG:

  NAME, RELATE, and VOLUMES

- The following parameters must be specified if they were coded in the original define:

  ATTEMPTS, AUTHORIZATION, CATALOG, CODE, CONTROLPW, FOR, MASTERPW, MODEL, OWNER, READPW, TO, and UPDATEPW

- Parameters originally defined but not required by RECATALOG are ignored.
- A RACF-protected object must be recataloged with the RACF protection attribute held in the VVDS.

DEFINE Cluster RECATALOG

This command re-creates catalog entries for a cluster, a data, and an index component. The appropriate VVDS and VTOC entries must exist. To RECATALOG requires that:

- The following parameters must be specified:

  INDEXED, NONINDEXED, NUMBERED, NAME, and VOLUMES

- The following parameters must be specified if they were coded in the original define:

  ATTEMPTS, AUTHORIZATION, CATALOG, CODE, CONTROLPW, FOR, MASTERPW, MODEL, OWNER, READPW, TO, and UPDATEPW

- A RACF-protected object must be recataloged with the RACF protection attribute held in the VVDS.

DEFINE Path RECATALOG

This command re-creates the path entry. To RECATALOG requires that:

- The following parameters must be specified:

  NAME and PATHENTRY

- The following parameters must be specified if they were coded in the original define:

  ATTEMPTS, AUTHORIZATION, CATALOG, CODE, CONTROLPW, FOR, MASTERPW, MODEL, NOUPDATE, OWNER, READPW, TO, UPDATE, and UPDATEPW

To change a path's attributes, use the ALTER command, not the RECATALOG function of DEFINE PATH.

DELETE TRUENAME

It is possible for the BCS to contain a component that is not accessible through its sphere record. To continue to use the same data set name, it is necessary to issue a DELETE TRUENAME for the components and a DEFINE RECATALOG to re-create the BCS entries.

If the associated base cluster subrecord or alternate index subrecord is missing or inaccessible, DELETE TRUENAME deletes the true name record of the component from the BCS.

After reading the true name record, DELETE uses the contents of the association cell and issues a read for the sphere record of the cluster. If the sphere record is not found, the true name record is deleted. The existence of a standalone true name record may be detected by the use of DIAGNOSE or may be indicated by a duplicate name error on DEFINE. The name so deleted may be used in a subsequent DEFINE.

DELETE VVR

This deletes:

- Unrelated VSAM volume records (VVRs) from the VVDS
- If present, the VTOC entry for the component

An unrelated VVR is one that exists in a VVDS, but has no related cluster, alternate index, data component, or index component entry in the BCS.

DELETE VVR requires the name of the component as the entry name. The true name must be absent to allow DELETE VVR to conclude successfully.

DELETE VVR fails if any one of the following is true:

- The VVR is not present.
- The catalog information in the VVR indicates that the component was not related to the specified catalog.
- The component still exists in the BCS.

Also, the CATALOG parameter must be specified; it is not sufficient to include JOBCAT or STEPCAT DD statements.

DELETE VVR removes unwanted records from the VVDS and from the VTOC if there is a Format-1 DSCB. However, there is no command to remove unwanted entries from the VTOC when a VVR entry is not present.

When a VVDS contains data sets that were cataloged in a BCS that no longer exists, it is not always desirable to use ALTER REMOVEVOLUMES to remove these data sets. If a BCS of the same name as that reflected in the VVR can be established, DELETE VVR can be used to remove them. PRINT with INFILE can be used to obtain the name of the BCS from the VVR and then the BCS can be defined. After deleting the VVR and the DSCB, delete the catalog.

DELETE Data Set NOSCRATCH

This deletes:

- Base cluster or GDG entries in the BCS
- Related entries in the BCS

It does not access the objects VVDS or VTOC entries.

The following remain intact:

- VVDS entries for the object
- VTOC entries for the object

DELETE data set NOSCRATCH processes only sphere and true name records. Therefore, it is not necessary for all components of the sphere to be present. Specifically, it can delete a partial sphere, including one with the "delete-in-progress" bit on.

DELETE data set NOSCRATCH can be used with DEFINE data set RECATALOG to:

- Replace contaminated BCS entries.
- Correct a down-level BCS that has been imported.
- Recover from an interrupted DEFINE or DELETE.

The related entries must also be recataloged.

DELETE data set NOSCRATCH cannot be used to delete an individual component of a cluster.

DELETE data set NOSCRATCH uses the entry name to read the sphere record. It uses the component names in the sphere record to read the true name records.

First the true name records are deleted and then the sphere record. If the sphere record cannot be obtained, DELETE data set NOSCRATCH terminates. The data and index components cannot be deleted individually, because they exist only as portions of the sphere record.

One reason why it might not be able to obtain the sphere records is that the cluster name (key) is damaged or missing. In the former case, if the (damaged) key can be determined, perhaps by dumping the BCS, then DELETE or DELETE data set NOSCRATCH should remove the sphere.

If the true name record cannot be obtained, DELETE data set NOSCRATCH continues processing:

- It deletes the sphere record.
- It issues messages that it deleted both the cluster and the component for which it found a true name record.

It does *not* issue a message that a true name record could not be found. The absence of a true name record does not prevent access to the VSAM data set. However, it does prevent access to the individual component by the component name.

DELETE data set NOSCRATCH is also used to delete non-VSAM entries.

For information on DELETE VVDS NOSCRATCH, see "DELETE VVDS NOSCRATCH" on page 93.

DELETE USERCATALOG FORCE

With DELETE USERCATALOG FORCE, the FILE parameter, if specified, must point to *all* volumes containing objects cataloged in the BCS. When multiple volumes are required, the use of JOBCAT or STEPCAT for dynamic allocation does not provide the necessary allocations. If the FILE parameter does not point to a volume containing the object, an error message is issued by DELETE.

DELETE USERCATALOG FORCE leaves the following intact:

- The VVDS on the catalog volume
- The VVDSs on any related volumes

DELETE USERCATALOG RECOVERY

For a nonempty BCS, this command:

- Scratches the DSCB for the BCS
- Deletes self-describing VVRs
- Deletes the BCS's connector record

The following remain intact after DELETE USERCATALOG RECOVERY:

- The VSAM data set entries in the VVDSs
- The VSAM data set entries in the VTOCs

This command would normally be used prior to importing a backup copy of the BCS.

When the connector record is deleted from the master catalog, all aliases belonging to the BCS are deleted. Therefore, a listing of the aliases should be obtained prior to executing DELETE USERCATALOG RECOVERY.

A JOBCAT or STEPCAT DD statement is required if the FILE parameter is not specified.

DELETE VVDS NOSCRATCH

This command deletes:

- The entry in the specified BCS for a VVDS that does not contain objects cataloged in that BCS
- The back pointer to the BCS in the VVDS control record (VVCR)

DELETE VVDS NOSCRATCH is used when objects on the volume are not and will not be cataloged in the specified catalog.

A check is made to ensure that the catalog does not contain objects on the specified volume. If the check fails, the command fails.

It might be assumed that the NOSCRATCH parameter means there is no requirement for the volume to be mounted. However, this is not the case. DELETE VVDS NOSCRATCH must ensure that the VVDS does not contain VVRs for data sets cataloged in the BCS from which the VVDS is to be deleted. Therefore, the VVDS volume must be mounted.

DELETE VVDS RECOVERY

This command deletes:

- The entry in the specified BCS for a VVDS
- The VTOC entry of the VVDS

The following remain intact after DELETE VVDS RECOVERY:

- The BCS and VTOC entries for the objects that were reflected in the VSAM volume records (VVRs) of the VVDS
- The entries for the VVDS in any other BCSs

DELETE VVDS RECOVERY is used to remove a damaged VVDS before rebuilding it. The VVDS may be rebuilt by importing each object defined on the VVDS volume.

Checks should be made to ensure that the master catalog does not contain entries for objects defined on the VVDS volume or the DELETE will fail.

Deleting the VVDS makes all VSAM objects on the volume inaccessible. This includes any BCSs that were on the volume. Therefore, it is necessary to have a way to recover such BCSs. DELETE UCAT RECOVERY should be used for them prior to DELETE VVDS RECOVERY. This allows the BCSs to be imported or defined.

The IMPORT or DEFINE of the first such BCS or other data set will result in the implicit DEFINE of the VVDS. Also, it is not necessary to DEFINE such a BCS prior to the IMPORT. See "EXPORT/IMPORT in Backup and Recovery" on page 94.

It may happen that a VVDS exists in a VTOC but is not cataloged in any BCS. Such a VVDS cannot be deleted. ALTER REMOVEVOLUMES will remove it, but it may be desirable not to use this command. One solution is to recatalog the VVDS using either of these methods:

- DEFINE VVDS RECATALOG
  1. DEFINE VVDS NONINDEXED RECATALOG to re-create the BCS entries

2. DELETE VVDS SCRATCH

- DEFINE CLUSTER on the VVDS volume

  This creates a definition for the VVDS in the BCS, provided the BCS is not already in the VVCR. (Refer to "ALTER REMOVEVOLUMES" on page 86.) A subsequent DELETE of the cluster empties the VVDS, and, because it is cataloged in a BCS, the VVDS itself may then be deleted.

EXPORT/IMPORT in Backup and Recovery

Data set recovery facilities, including EXPORT/IMPORT, are summarized below:

BCS	VVDS	VTOC	ACTION
P	P	P	IMPORT
P	P	M	IMPORT
P	M	P	IMPORT
P	M	M	IMPORT
M	P	P	DELETE VVR & IMPORT
M	P	M	DELETE VVR & IMPORT
M	M	P	ZAP, SCRATCH & IMPORT
M	M	M	IMPORT

where:

P = entry present
M = entry missing
ZAP = use AMASPZAP to turn off the OS protection bit
SCRATCH = use IEHPROGM to scratch the DSCB

- **BCS missing/VVDS present/VTOC missing (MPM)**

  In situations where:

  - The BCS record for the cluster is missing.
  - The VVR is present.
  - The DSCB is missing.

  In these situations, IMPORT concludes successfully but a second VVR is created. This second VVR may eventually cause problems. For example, the original VVR may now point to an incorrect CCHH. The recovery procedure should include a DELETE VVR prior to using IMPORT.

- **BCS missing/VVDS missing/VTOC present (MMP)**

  IMPORT does not execute if it finds a Format-1 DSCB without corresponding entries in the BCS and the VVDS. It is useful to have some way to remove the Format-1 prior to running IMPORT. Two means for accomplishing this are:

  1. ALTER REMOVEVOLUMES to remove the Format-1 DSCBs.
  2. AMASPZAP to turn off the OS protection bit and IEHPROGM to SCRATCH the DSCB.

For the **basic catalog structure** it is important to have access to a backup copy of the BCS to IMPORT during recovery processing. Otherwise, it may be necessary to rebuild the BCS using DEFINE for each cluster. It is easy to delete unwanted entries from the BCS using commands such as DELETE NOSCRATCH and DELETE TRUENAME.

- **VVDS**

  The VVDS is implicitly defined either:

  - When a copy of a BCS is imported, or
  - When a BCS is defined prior to importing data sets.

  In the first case, it is not necessary to define the BCS before importing the backup copy.

- **Control area size**

  If a BCS is defined with secondary allocation less than 1 cylinder, the control area size is in tracks. If IMBED is specified or allowed to default, DEFINE makes the control area 1 track larger than specified. The use of IMPORT for recovery or reorganization will continue to increase the size of the control area until it reaches a cylinder.

  To avoid this situation, use the following commands in the order shown:

  1. DELETE RECOVERY USERCATALOG
  2. DEFINE USERCATALOG
  3. IMPORT INTOEMPTY

- **DISP**

  IMPORT differentiates between DISP = OLD and DISP = SHR. Specifically:

  - If OUTFILE is used when importing a BCS with the default SHAREOPTIONS of (3 4), DISP = OLD must be used on the DD statement.
  - Using DISP = SHR results in the message IDC3351I I/O rc = 28 being issued.

Whenever a BCS is imported, DIAGNOSE should be run to determine any activity that is not reflected in the imported copy of the BCS.

Interpreting Messages

Some of the most common messages indicating a problem with the BCS or merely that an EXPORT DISCONNECT or an IMPORT CONNECT has been forgotten, is the series:

- IEC331I return code 004 reason code 40
- IEC161I return code 004 reason code 80
- IEC331I return code 004 reason code 86

These messages are issued during OPEN and are basically saying that the catalog is not available. They are usually accompanied by an IDC3009I 4-2 message. Two common reasons for these messages are:

- The VVDS is not available.
- The connection between the BCS and the VVDS is broken.

If this set of messages occurs and it is not merely a missing CONNECT or DISCONNECT, the recovery procedure is to redefine the catalog and import a backup copy. Because the catalog cannot be opened, ALTER REMOVEVOLUMES and AMASPZAP/IEHPROGM are the only ways to remove the catalog DSCB.

Another common message is IDC3009I 50-6. This indicates the absence of a VVR. It may occur as a result of importing a down-level BCS containing a data set that

has been deleted from the VVDS but has entries in the imported BCS. LISTCAT ALL usually identifies such problems.

Other Backup and Recovery Facilities

Data Facility Data Set Services (DFDSS)

Two important uses of DFDSS *volume* restore are the recovery of a VSAM data set cataloged in an integrated catalog facility catalog and the recovery of a VVDS.

- **Volume restore to recover a data set**

  A volume containing a VSAM data set cataloged in an integrated catalog facility catalog can be dumped with DFDSS. The dump can be restored to a scratch volume with the same volume serial number. If the medium of the original volume is fixed, it should be varied offline and a scratch volume with the same volume serial number should be restored. Access to the data set after the volume restore depends on where the data set is cataloged:

  - **Cataloged in a BCS on the restored volume**

    1. REPRO or EXPORT the data set to another volume or tape.
    2. Discard the scratch volume and mount the original volume.
    3. DELETE and DEFINE the data set.
    4. REPRO or IMPORT the data set.

  - **Cataloged in a BCS on another volume**

    1. EXPORT the data set.
    2. Vary offline the scratch volume; vary online the real one.
    3. IMPORT the data set.
    4. Discard the scratch volume.

- **Volume restore to recover a VVDS**

  DFDSS volume restore provides a means for replacing the VVDS. An alternative is to be in the position to rebuild the VVDS by importing every VSAM data set that resides on the volume, but in many instances this is not feasible.

VSAM data sets cataloged in an integrated catalog facility catalog may be backed up and restored using DFDSS Release 2.0. You should use caution when restoring catalogs and multivolume data sets.

You must take the same precautions when restoring a catalog with DFDSS as you do when using the access method services IMPORT command. DFDSS does not access the catalog during the dump and does not catalog the data set after the restore. You must ensure that there is no activity on the data set during the dump or restore for a multivolume data set and that all the segments for a multivolume data set are obtained during dump and restore.

Restoring a VVDS using data set restore is not recommended as a normal recovery method.

Recovery Procedure Scenarios

Three main combinations exist in the relationship between a BCS and a VVDS:

- The BCS and VVDS are on the same volume but are not connected to any other BCSs or VVDSs.
- The BCS and VVDS are on the same volume but the BCS is connected to another VVDS, which, in turn, coresides with a BCS on the other volume.
- Only the VVDS is on the volume, with connections to a BCS on another volume.

Based on these combinations, there is only one basic procedure to recover a BCS; there are three procedures to rebuild a VVDS, depending upon which of the above situations applies. The following descriptions provide some ideas for possible recovery procedures.

With EXPORT copies of the BCS and the VSAM data sets:

- The BCS may be restored or rebuilt.
- The VVDS may be rebuilt.
- A VSAM data set may be restored.

Data set recovery is described under "EXPORT/IMPORT in Backup and Recovery" on page 94.

To recover a BCS using an exported copy, execute the following steps:

1. Use LISTCAT NONVSAM ALL to provide a list of the non-VSAM data set names. Later, when the backup BCS is imported, do another LISTCAT and compare the two lists. If there are differences, use DEFINE or DELETE NOSCRATCH to resolve those differences.
2. If EXPORT was used to create the backup copy of the BCS, the BCS aliases were exported with the BCS, and this step can be ignored. Otherwise, use LISTCAT ENTRIES(catalog name) ALL to provide a list of the aliases associated with the catalog, because the next step will delete those aliases.
3. Do a DELETE RECOVERY USERCATALOG. Note that this does NOT delete the data set entries from either the VVDS or the VTOC.
4. Use IMPORT ALIAS LOCK to import the backup copy of the BCS, and to lock the BCS.
5. If EXPORT was used to create the backup BCS, this step can be ignored. Otherwise, use DEFINE to re-create the aliases, using the alias list created earlier.
6. Use DIAGNOSE to determine differences that may exist between the BCS and each VVDS containing data sets cataloged in the BCS.
7. Use LISTCAT NONVSAM ALL to create a list of non-VSAM data sets. Compare this list with the list generated in the first step.
8. Update the BCS, if necessary:
   a. Use DEFINE RECATALOG to update the BCS with new data sets from information found in the VVDSs.
   b. Use DELETE NOSCRATCH to remove entries from the BCS for data sets that no longer exist in the VVDSs.

c. Use DEFINE and DELETE NOSCRATCH to update the BCS for non-VSAM data sets.

9. Use ALTER UNLOCK to unlock the catalog and make it available for general use.

The environment consists of:

- A BCS named ICFCAT1 on a volume named SPOOL1
- VSAM data sets on SPOOL1 and STVOL3 and non-VSAM data sets on SPOOL1

The steps that were run to set up the environment have been omitted.

1. If possible, LISTCAT to obtain non-VSAM data set names:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  LISTCAT -
       NONVSAM-
       ALL -
       CATALOG(ICFCAT1)
/*
```

2. LISTCAT to obtain aliases to BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  LISTCAT -
       ENTRIES(ICFCAT1) -
       ALL
/*
```

3. DELETE BCS RECOVERY:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DELETE -
       ICFCAT1 -
       RECOVERY -
       USERCATALOG
/*
```

4. IMPORT the BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD2       DD    DSN=ICFCAT1,DISP=OLD,AMP=AMORG,
//                UNIT=SYSDA,VOL=SER=SPOOL1
//DD1       DD    DSN=STY.EXPORT,UNIT=SYSDA,VOL=SER=SPOOL1,
//                DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTFILE(DD2) -
       ALIAS -
       LOCK
/*
```

5. Use the VVDSs to determine whether the BCS requires updates:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=ICFCAT1,DISP=SHR
//DD2       DD    DSN=SYS1.VVDS.VSPOOL1,UNIT=SYSDA,
//                VOL=SER=SPOOL1,AMP=AMORG,DISP=OLD
//DD3       DD    DSN=SYS1.VVDS.VSTVOL3,UNIT=SYSDA,
//                VOL=SER=STVOL3,AMP=AMORG,DISP=OLD
//SYSIN     DD    *
  DIAGNOSE -
       VVDS -
       INFILE(DD2) -
       COMPAREDD(DD1) -
       LIST
  DIAGNOSE -
       VVDS -
       INFILE(DD3) -
       COMPAREDD(DD1) -
       LIST
/*
```

6. Use LISTCAT to obtain non-VSAM data set names and compare with the first list:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  LISTCAT -
       NONVSAM -
       ALL -
       CATALOG(ICFCAT1)
/*
```

7. Update the BCS if necessary:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DEFINE CLUSTER -
       (NAME(name) -
       VOLUMES(SPOOL1) -
       TRACK(1) -        (you must list other
       RECATALOG)         required attributes)
  DELETE -
       NOSCRATCH -
       CLUSTER
  DEFINE -
       NONVSAM -
       (NAME(name) -
       DEVICETYPE(....) -
       VOLUME(SPOOL1))
  DELETE -
       NOSCRATCH -
       NONVSAM
/*
```

8. Unlock the catalog:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  ALTER -
       ICFCAT1 -
       UNLOCK
/*
```

Recover VVDS with Connection Only to BCS on the Same Volume

If a BCS and VVDS are on the same volume, but the VVDS is not connected to another BCS and it is not possible to connect the VVDS to another BCS, then:

1. Use LISTCAT NONVSAM ALL to provide a list of the non-VSAM data set names. Later, when the backup BCS is imported, do another LISTCAT and compare the two lists. If there are differences, use DEFINE or DELETE NOSCRATCH to resolve those differences.
2. Use LISTCAT ENTRIES(catalog name) ALL to provide a list of the aliases associated with the catalog, because the EXPORT DISCONNECT to be done later will delete the aliases.
3. If possible, EXPORT the BCS to ensure that all data sets are subsequently imported.
4. Use ALTER REMOVEVOLUMES, unless there is a VSAM data set cataloged in a VSAM catalog on the volume. In this latter case, use AMASPZAP AND IEHPROGM.
5. Use EXPORT DISCONNECT to disconnect the BCS record in the master catalog.
6. Use IMPORT ALIAS LOCK to import the backup copy of the BCS and to lock the BCS.
7. If EXPORT was used to create the backup BCS, this step can be ignored. Otherwise, use DEFINE to re-create the aliases, using the alias list created earlier.
8. Use DIAGNOSE to determine differences that may exist between the BCS and each VVDS containing data sets cataloged in the BCS.
9. Use LISTCAT NONVSAM ALL to create a list of non-VSAM data sets. Compare this list with the list generated in the first step.
10. Update the BCS, if necessary:
    a. Use DEFINE RECATALOG to update the BCS with new data sets from information found in the VVDSs.
    b. Use DELETE NOSCRATCH to remove entries from the BCS for data sets that no longer exist in the VVDSs.
    c. Use DEFINE and DELETE NOSCRATCH to update the BCS for non-VSAM data sets.
11. Update the VVDS, if necessary, by importing clusters.
12. Use ALTER UNLOCK to unlock the catalog and make it available for general use.

The environment consists of:

- A BCS named ICFCAT1 on a volume named SPOOL1
- VSAM and non-VSAM data sets on SPOOL1

The steps that were run to set up the environment, have been omitted.

1. If possible, use LISTCAT to obtain non-VSAM data set names:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  LISTCAT -
       NONVSAM -
       ALL -
       CATALOG(ICFCAT1)
/*
```

2. Use LISTCAT to obtain aliases to BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  LISTCAT -
       ENTRIES(ICFCAT1) -
       ALL
/*
```

3. If possible, EXPORT the BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=A
//STEPCAT   DD   DSN=ICFCAT1,DISP=SHR
//DD1       DD   DSN=STY.EXPORT,UNIT=SYSDA,VOL=SER=SPOOL1,
//               DISP=OLD
//SYSIN     DD   *
  EXPORT -
       ICFCAT1 -
       OUTFILE(DD1) -
       TEMPORARY
/*
```

4. Remove the BCS and the VVDS and DISCONNECT the catalog:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD2       DD    UNIT=SYSDA,VOL=SER=SPOOL1,DISP=OLD
//SYSIN     DD    *
  EXPORT -
       ICFCAT1 -
       DISCONNECT
  ALTER -
       master catalog name -
       FILE(DD2) -
       REMOVEVOLUMES(SPOOL1)
/*
```

5. IMPORT the BCS and implicitly DEFINE the VVDS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD3       DD    DSN=ICFCAT1,DISP=OLD,AMP=AMORG,
//                UNIT=SYSDA,VOL=SER=SPOOL1
//DD1       DD    DSN=STY.EXPORT,UNIT=SYSDA,VOL=SER=SPOOL1,
//                DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTFILE(DD3) -
       ALIAS -
       LOCK
/*
```

6. IMPORT the clusters that reside on the VVDS volume:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=P,UNIT=SYSDA,VOL=SER=SPOOL1,DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTDATASET(name)
/*
```

7. DIAGNOSE:

   a. Use the BCS to determine if the VVDS requires updates:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=ICFCAT1,DISP=SHR
//DD2       DD    DSN=SYS1.VVDS.VSPOOL1,UNIT=SYSDA,
//                VOL=SER=SPOOL1,AMP=AMORG,DISP=OLD
//SYSIN     DD    *
  DIAGNOSE -
       ICFCATALOG -
       INFILE(DD1) -
       COMPAREDD(DD2) -
       LIST
/*
```

   b. Use the VVDS to determine if the BCS requires updates:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=ICFCAT1,DISP=SHR
//DD2       DD    DSN=SYS1.VVDS.VSPOOL1,UNIT=SYSDA,
//                VOL=SER=SPOOL1,AMP=AMORG,DISP=OLD
//SYSIN     DD    *
  DIAGNOSE -
       VVDS -
       INFILE(DD2) -
       COMPAREDD(DD1) -
       LIST
/*
```

8. Use LISTCAT to obtain non-VSAM data set names and compare with first list:

```
//STEP1     EXEC   PGM=IDCAMS
//SYSPRINT  DD     SYSOUT=A
//SYSIN     DD     *
  LISTCAT -
       NONVSAM -
       ALL -
       CATALOG(ICFCAT1)
/*
```

9. If necessary, update the BCS:

```
//STEP1     EXEC   PGM-IDCAMS
//STEPCAT   DD     DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD     SYSOUT=A
//SYSIN     DD     *
  DEFINE CLUSTER -
       (NAME(name) -
       VOLUMES(SPOOL1) -     (you must list other
       TRACK(1))              required attributes)
  DELETE -
       NOSCRATCH -
       NONVSAM
/*
```

10. If the VVDS requires updates, import the clusters:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=P,UNIT=SYSDA,VOL=SER=SPOOL1,DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTDATASET(name)
/*
```

11. Unlock the catalog:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  ALTER -
       ICFCAT1 -
       UNLOCK
/*
```

Recover VVDS When BCS Is on Volume but VVDS Is Known to Another BCS

To recover a VVDS when neither the BCS on that volume nor the BCS on another volume can access data sets on that volume:

1. Use LISTCAT NONVSAM ALL to provide a list of the non-VSAM data set names. Later, when the backup BCS is imported, do another LISTCAT and compare the two lists. If there are differences, use DEFINE or DELETE NOSCRATCH to resolve those differences.
2. Use PRINT VVDS COUNT(1) to establish which catalogs have connections to the VVDS. These names are required for the next step.

3. Use DIAGNOSE VVDS INCLUDE CATALOG(catalog name) LIST to obtain a list of the data sets that are cataloged in (catalog name) and are on this volume. Rerun this for each BCS that has connections to the VVDS.
4. If possible, EXPORT the BCS on the VVDS volume to provide a check that all data sets are subsequently imported.
5. Use LISTCAT ENTRIES(catalog name) ALL to provide a list of the aliases associated with the catalog, because the next step will delete those aliases.
6. Do a DELETE RECOVERY USERCATALOG so that the catalog may be subsequently imported.
7. Do a DELETE VVDS RECOVERY so that the VVDS may be redefined and rebuilt.
8. Use IMPORT ALIAS LOCK to import the backup copy of the BCS and to lock the BCS.
9. If EXPORT was used to create the backup BCS, this step can be ignored. Otherwise, use DEFINE to re-create the aliases, using the alias list created earlier.
10. Use DIAGNOSE to determine differences that may exist between the BCS and each VVDS containing data sets cataloged in the BCS.
11. Use LISTCAT NONVSAM ALL to create a list of non-VSAM data sets. Compare this list with the list generated in the first step.
12. Update the BCS, if necessary:
    a. Use DEFINE RECATALOG to update the BCS with new data sets from information found in the VVDSs.
    b. Use DELETE NOSCRATCH to remove entries from the BCS for data sets that no longer exist in the VVDSs.
    c. Use DEFINE and DELETE NOSCRATCH to update the BCS for non-VSAM data sets.
13. Update the VVDS, if necessary, by importing clusters.
14. Use ALTER UNLOCK to unlock the catalog and make it available for general use.

The environment consists of:

- A BCS named ICFCAT1 on a volume named SPOOL1
- Another BCS named ICFCAT2 on a volume named STVOL3
- Both catalogs have data sets on STVOL3 and SPOOL1

The VVDS on SPOOL1 is to be rebuilt.

The steps that were run to set up the environment have been omitted.

1. If possible, use LISTCAT to obtain non-VSAM data set names:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  LISTCAT -
       NONVSAM -
       ALL -
       CATALOG(ICFCAT1)
  LISTCAT -
       NONVSAM -
       ALL -
       CATALOG(ICFCAT2)
/*
```

2. Use PRINT to establish the names of the catalogs that have connections to this VVDS, if it can be opened:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  PRINT -
       INDATASET(SYS1.VVDS.VSPOOL1) -
       COUNT(1)
/*
```

3. Use DIAGNOSE to obtain a list of data sets on this volume and the BCSs in which they are cataloged, if the VVDS and BCS can be opened:

```
//STEP1      EXEC  PGM=IDCAMS
//SYSPRINT   DD    SYSOUT=A
//DD1        DD    DSN=SYS1.VVDS.VSPOOL1,UNIT=SYSDA,
//                 VOL=SER=SPOOL1,AMP=AMORG,DISP=OLD
//SYSIN      DD    *
  DIAGNOSE -
       VVDS -
       INFILE(DD1) -
       INCLUDE -
       (CATALOG(ICFCAT1)) -
       LIST
  DIAGNOSE -
       VVDS -
       INFILE(DD1) -
       INCLUDE -
       (CATALOG(ICFCAT2)) -
       LIST
/*
```

4. If possible, EXPORT the BCS:

```
//STEP1    EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//DD1       DD    DSN=STY.EXPORT,UNIT=SYSDA,VOL=SER=SPOOL1,
//                DISP=OLD
//SYSIN     DD    *
  EXPORT -
       ICFCAT1 -
       OUTFILE(DD1) -
       TEMPORARY
/*
```

5. Use LISTCAT to obtain aliases to BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  LISTCAT -
       ENTRIES(ICFCAT1) -
       ALL
/*
```

6. DELETE BCS RECOVERY on the volume containing the damaged VVDS:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DELETE -
       ICFCAT1 -
       RECOVERY -
       USERCATALOG
/*
```

7. DELETE VVDS RECOVERY pointing to the BCS on the other volume:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT2, DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DELETE -
       SYS1.VVDS.VSPOOL1 -
       RECOVERY
/*
```

8. IMPORT the BCS and implicitly DEFINE the VVDS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD3       DD    DSN=ICFCAT1,DISP=OLD,AMP=AMORG,
//              UNIT=SYSDA,VOL=SER=SPOOL1
//DD1       DD    DSN=STY.EXPORT,UNIT=SYSDA,VOL=SER=SPOOL1,
//              DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTFILE(DD3) -
       ALIAS -
       LOCK
/*
```

9. IMPORT the clusters:

   a. From the first BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//DD1       DD    DSN=P,UNIT=SYSDA,VOL=SER=SPOOL1,
//              DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTDATASET(name)
/*
```

   b. From the second BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//STEPCAT   DD    DSN=ICFCAT2,DISP=SHR
//DD1       DD    DSN=D,UNIT=SYSDA,VOL=SER=SPOOL1,
//              DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTDATASET(name)
/*
```

10. Use each BCS to determine whether the rebuilt VVDS requires updates:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=ICFCAT1,DISP=SHR
//DD2       DD    DSN=ICFCAT2,DISP=SHR
//DD3       DD    DSN=SYS1.VVDS.VSPOOL1,UNIT=SYSDA,
//              VOL=SER=SPOOL1,AMP=AMORG,DISP=OLD
//SYSIN     DD    *
  DIAGNOSE -
       ICFCATALOG -
       INFILE(DD1) -
       COMPAREDD(DD3) -
       LIST
  DIAGNOSE -
       ICFCATALOG -
       INFILE(DD2) -
       COMPAREDD(DD3) -
       LIST
/*
```

11. Use each VVDS to determine whether the BCS on SPOOL1 requires updates:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=ICFCAT1,DISP=SHR
//DD2       DD    DSN=SYS1.VVDS.VSTVOL3,UNIT=SYSDA,
//            VOL=SER=STVOL3,AMP=AMORG,DISP=OLD
//DD3       DD    DSN=SYS1.VVDS.VSPOOL1,UNIT=SYSDA,
//            VOL=SER=SPOOL1,AMP=AMORG,DISP=OLD
//SYSIN     DD    *
  DIAGNOSE -
       VVDS -
       INFILE(DD2) -
       COMPAREDD(DD1) -
       LIST
  DIAGNOSE -
       VVDS -
       INFILE(DD3) -
       COMPAREDD(DD1) -
       LIST
/*
```

12. Use LISTCAT to obtain non-VSAM data set names and compare with first list:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  LISTCAT -
       NONVSAM -
       ALL -
       CATALOG(ICFCAT1)
/*
```

13. If necessary, update the BCS:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DEFINE CLUSTER -
       (NAME(name) -
       VOLUMES(......) -
       TRACK(1) -              (you must list other
       RECATALOG)               required attributes)
  DELETE -
       entry name -
       NOSCRATCH -
       CLUSTER
  DEFINE -
       NONVSAM -
       (NAME(name) -
       DEVTYPE(....) -
       VOLUME(......))
  DELETE -
       entry name -
       NOSCRATCH -
       NONVSAM
/*
```

14. If the VVDS requires updates, import the clusters:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=P,UNIT=SYSDA,VOL=SER=SPOOL,
//            DISP=OLD
//SYSIN     DD      *
  IMPORT -
       INFILE(DD1) -
       OUTDATASET(name) -
/*
```

15. Unlock the catalog:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  ALTER -
       ICFCAT1 -
       UNLOCK
/*
```

Recover VVDS When BCS Is Not on the Volume

When only a VVDS is on the volume with connections to one or more BCSs on other volumes:

1. PRINT VVDS COUNT(1) to establish which catalogs have connections to the VVDS. These names are required for the next step.
2. DIAGNOSE VVDS INCLUDE CATALOG(catalog name) LIST to obtain a list of data sets that are cataloged in (catalog name) and are in this VVDS. Rerun this for each BCS that has connections to the VVDS.
3. DELETE VVDS RECOVERY.
4. IMPORT all the data sets that resided on this volume.
5. DIAGNOSE to determine any differences that may exist between the new VVDS and each BCS that had data sets on this volume.
6. Update the VVDS, if necessary, by importing clusters.

The environment consists of:

- A VVDS on a volume named SPOOL1
- A BCS named ICFCAT2 on a volume named STVOL3
- The BCS has data sets on STVOL3 and SPOOL1

Other catalogs have data sets on SPOOL1.

The steps that were run to set up the environment have been omitted.

1. Use PRINT to establish the names of the catalogs that have connections to this VVDS, if it can be opened:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  PRINT -
       INDATASET(SYS1.VVDS.VSPOOL1) -
       COUNT(1)
/*
```

2. Use DIAGNOSE to obtain a list of data sets on this volume and the BCSs in which they are cataloged, if the VVDS can be opened:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=SYS1.VVDS.VSPOOL1,UNIT=SYSDA,
//                VOL=SER=SPOOL1,AMP=AMORG,DISP=OLD
//SYSIN     DD    *
  DIAGNOSE -
       VVDS -
       INFILE(DD1) -
       INCLUDE -
       (CATALOG(ICFCAT2)) -
       LIST
  DIAGNOSE -
       VVDS -
       IFILE(DD1) -
       INCLUDE -
       (CATALOG(catalog name)) -
       LIST
/*
```

3. DELETE VVDS RECOVERY:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//STEPCAT   DD    DSN=ICFCAT2,DISP=SHR
//DD1       DD    UNIT=SYSDA,DISP=SHR,VOL=SER=SPOOL1
//SYSIN     DD    *
  DELETE -
       SYS1.VVDS.VSPOOL1 -
       RECOVERY -
       FILE(DD1)
/*
```

4. IMPORT the clusters that reside on the VVDS volume and are cataloged in ICFCAT2. Repeat this job for other catalogs:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT2,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=P,UNIT=SYSDA,VOL=SER=SPOOL1,DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTDATASET(name)
/*
```

5. Use the BCS to determine whether the VVDS requires updates. Repeat this job for other catalogs:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=ICFCAT2,DISP=SHR
//DD2       DD    DSN=SYS1.VVDS.VSPOOL1,UNIT=SYSDA,
//                VOL=SER=SPOOL1,AMP=AMORG,DISP=OLD
//SYSIN     DD    *
  DIAGNOSE -
       ICFCATALOG -
       INFILE(DD1) -
       COMPAREDD(DD2) -
       LIST
/*
```

6. If the VVDS requires updates, IMPORT the clusters:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT2,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=P,UNIT=SYSDA,VOL=SER=SPOOL1,DISP=OLD
//SYSIN     DD    *
  IMPORT -
       INFILE(DD1) -
       OUTDATASET(name)
/*
```

Examples of Printing and Deleting

1. Using DFDSS to print a VSAM data set:

```
//STEP1     EXEC  PGM=ADRDSSU
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DISP=OLD,UNIT=SYSDA,VOL=SER=SPOOL1
//SYSIN     DD    *
  PRINT -
       INDDNAME(DD1) -
       TRACKS(X'02C6',X'D',X'02C6',X'F')
/*
```

2. Deleting a VVR:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    UNIT=SYSDA,VOL=SER=STVOL3,DISP=OLD
//SYSIN     DD    *
  DELETE -
       entry name -
       VVR -
       FILE(DD1) -
       CATALOG(ICFCAT1)
/*
```

3. Printing a VVDS without using a catalog:

```
//STEP1     EXEC  PGM=IDCAMS
//SYSPRINT  DD    SYSOUT=A
//DD1       DD    DSN=SYS1.VVDS.VSPOOL1,DISP=SHR,
//                UNIT=SYSDA,VOL=SER=SPOOL1,AMP=AMORG
//SYSIN     DD    *
  PRINT -
       INFILE(DD1)
/*
```

4. Deleting a true name record:

```
//STEP1     EXEC  PGM=IDCAMS
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSIN     DD    *
  DELETE -
       entry name -
       TRUENAME
/*
```

5. Using AMASPZAP to print the VVDS:

```
//STEP1     EXEC  PGM=AMASPZAP
//STEPCAT   DD    DSN=ICFCAT1,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
//SYSLIB    DD    DSN=SYS1.VVDS.VSPOOL1,DISP=OLD,UNIT=SYSDA,
//                UNIT=SYSDA,VOL=SER=SPOOL1
//SYSIN     DD    *
  ABSDUMPT ALL
/*
```

Chapter 6. Checking Catalogs for Errors and Synchronization

Listing Catalog Information

- **PRINT**

  The BCS and the VVDS may be printed using several different methods. With the access method services PRINT command, you may print the VVDS without reference to a catalog by specifying:

  - INFILE in the command and
  - AMP=AMORG in the DD statement

- **LISTCAT**

  You can list the entire catalog record, the statistics, and the parameters selected when the data set was defined, by using the LISTCAT command. If ALLOC or ALL is specified, VVDS volumes are required. Some statistics, such as number of extents in a data set, number of records retrieved, added, deleted, and updated, and number of control interval splits, can help you decide when to take action, such as reorganizing a data set or altering the type of processing, to improve performance.

- **IEHLIST VTOC**

  IEHLIST LISTVTOC can provide the names and locations of the BCSs and VVDSs.

- **Data Facility Data Set Services**

  PRINT can print a BCS or VVDS as specified by a fully qualified name.

Using the DIAGNOSE Command

DIAGNOSE is an essential integrated catalog facility catalog administration tool. It is the recommended tool to use to indicate the presence of invalid data or relationships in the BCS and/or VVDS that may cause problems if not corrected.

If errors are indicated in the output of DIAGNOSE, they should be used to determine which recovery procedures are to be used. DIAGNOSE detects inconsistencies between the BCS, VVDS, and VTOC. DIAGNOSE checking can be grouped into:

- Checking an entry's format and content
- Comparing an entry's dependent content for consistency

INCLUDE/EXCLUDE Parameters

Selective checking for particular parts of the BCS or VVDS may be accomplished via:

- INCLUDE—to check only those entries meeting the specification
- EXCLUDE—to check the complement of the entries meeting the specification

Up to 255 names may be specified. When specifying the names for INCLUDE or EXCLUDE, the entity named is used to define the scope of the check. However, because of the record architecture, entries specified to be excluded may have to be scanned to reach the requested entry. Scanning such entries may fail because of record format errors. The result will be errors for entries that were to be excluded. (See *Catalog Diagnosis Reference* for further information.)

The use of INCLUDE or EXCLUDE as a DIAGNOSE parameter influences what is processed. Figure 27 outlines what happens in processing when INCLUDE and EXCLUDE are specified with DIAGNOSE.

BCS or VVDS Entity Type	Normal Processing	Processing with INCLUDE	Processing with EXCLUDE
VSAM CLUSTER	Cluster and all components as well as paths	VSAM cluster and components but no paths	VSAM cluster and components but no paths
VSAM component	Cluster and all components components as well as paths	Only the component	Only the component
VSAM PATH	Path and related AIX or base cluster	Only the path entry	Only the path entry
NON-VSAM	Entry and any aliases	Entry only	Entry only
NON-VSAM ALIAS	Alias and the base entry	Alias entry only	Alias entry only
GDG BASE (GDG)	GDG base and GDSs as well as aliases	GDG base and GDSs	GDG base and GDSs
Generation data set (GDS)	GDG base and GDSs as well as aliases	GDS only	GDS only

Figure 27. Using INCLUDE and EXCLUDE as DIAGNOSE Parameters

COMPARE Parameter

Dependency checking is accomplished by specifying COMPARE. For example, the VVDS is checked for an entry that is consistent with the BCS entry. This is not a complete check of the external data set or its entry. It is only a consistency check between the two. Because only VSAM data sets have VVDS entries, only VSAM entries receive consistency checks. If COMPARE is specified for:

- A BCS, the VVDS is checked for dependency.
- A VVDS, the BCS and the VTOC are checked for dependency.

If a VVR is checked, either as the result of diagnosing the VVDS, or because the COMPARE parameter was used, the corresponding DSCB is always checked, as the VVR and the DSCB are treated as a unit. A check of the VVR results in a comparison of the DSCB extent information with the VVR extent information.

Up to 36 names may be coded for the COMPARE parameter and you may check 36 VVDSs for consistency against an integrated catalog facility catalog or 36 BCSs against a VVDS. It is the entries in the input data set that determine which entries in the compare data set are to receive compare checking. This may prevent some of the entries in the compare data set from being checked.

If dependency checking (COMPARE) is not specified, DIAGNOSE does not use or confirm external pointers to the BCS for the specified data set.

Note: DIAGNOSE cannot identify the situation in which a VVDS should be specified, so that a further comparison can occur.

DIAGNOSE Output

The analysis facility in DIAGNOSE results in messages. These messages help those responsible for portions of the integrated catalog facility catalog to make decisions about the validity of these areas and to decide which recovery techniques would be suitable, should errors be identified.

The messages provided by DIAGNOSE can result in the following summaries:

- A list of all entries that had no errors
- A list of entries that had errors
- A list of volume serial numbers that were found to be associated with the BCS that were not encountered during a BCS entry scan
- A list of BCS names found to be associated with a given VVDS that were not encountered during a VVDS entry scan
- A list of COMPARE members that were not encountered during processing
- A list of INCLUDE or EXCLUDE members that were not encountered during processing

Unless the INCLUDE or EXCLUDE parameter is used with DIAGNOSE, all records and entries in the input data set (BCS or VVDS) are processed.

Storage Estimate for DIAGNOSE

When running DIAGNOSE, consider the following information to estimate preliminary storage requirement:

- The access method services overhead is approximately 40K bytes.
- DIAGNOSE ICFCATALOG takes 32K bytes to open a catalog and 16K bytes to open each VVDS specified in COMPARED or COMPAREDS.
- DIAGNOSE VVDS takes 16K bytes to open a VVDS and 32K bytes to open a catalog specified in COMPARED or COMPAREDS.
- Each entry takes 45 bytes for entry record and approximately 40 bytes for VOLSER, DLST, NLST, PLST, ILST, CLST, and RLST.
- To avoid a storage problem, you should not create a catalog that has a huge number of entries. As an example, a 4-megabyte region size must not have a catalog of more than 30000 entries.

Messages from DIAGNOSE

The output of DIAGNOSE generally consists of three error messages. The first, message IDC21364I, gives you the following information:

- The name (and type) of data entry being checked
- The key of the record for a BCS or the RBA of the record for a VVDS
- The offset to the start of the cell causing the error
- A reason that describes the error

The second message is usually IDC21365I, which provides a display of the records in error. If this record was previously displayed, message 'IDC01371I RECORD DISPLAY SUPPRESSED, ALREADY DUMPED' is issued instead.

The records that DIAGNOSE displays depend on the type of error condition encountered. Any of the following three types might be displayed:

- The BCS record
- The VVDS record
- The Format-1 DSCB

With the exception of the Format-1 DSCB, the records are composed of various subrecords or cells.

Use the "offset" field (produced in message IDC21364I), to determine the start of the cell in error. For a listing and description of the various cell types, see "Data Areas" in *Catalog Diagnosis Reference.*

Note: In the case of a Format-1 DSCB, the first 44 bytes of the record are not displayed.

The third message produced by DIAGNOSE is IDC21363I, which is basically a summary of all the errors found by DIAGNOSE.

DIAGNOSE Message Record Notation

DIAGNOSE messages identify the entries and records involved in certain error situations. Catalog records have 45-byte binary keys, each composed of a 44-byte EBCDIC character portion and a binary pad byte. When DIAGNOSE prints the two portions of the 45-byte key, they are separated by a slash (/).

For example:

```
SYS1.VVDS.VCATALG /01
```

The actual key is 45 bytes of binary data, but the left portion has been translated to EBCDIC and printed as such. The last byte of the key is 01. This notation allows extension records to be noted and allows discrimination between the base and any extensions. Excessive blanks have been removed.

For VVDS records, the hexadecimal RBA of the record is used as its 'KEY' or identifier.

For example:

```
X'00002000'
```

VTOC records have 44-byte EBCDIC keys; no special notation is needed.

DIAGNOSE Message Entry Notation

The entry name, when part of a DIAGNOSE message, is followed by the entry type in parentheses. An example of an entry name is:

```
SYS1.VVDS.VCATALG  (D)
```

The "D" in parentheses is the entry type: Data component. For information on entry types, see "Sample Diagnosis Output" in *Catalog Diagnosis Reference*.

Interpreting DIAGNOSE Output

The messages produced by DIAGNOSE can be divided into the following message types:

- DIAGNOSE error messages
- Execution error messages
- Summary messages

Figure 28 lists all the DIAGNOSE messages by number, the appropriate condition code for each message, and the associated message type for each message.

DIAGNOSE Message Number	Condition Code Associated With Message	Message Type
IDC01360I	0	Summary
IDC01371I	0	Execution
IDC11361I	4	Summary; may be syntax
IDC11362I	4	Summary
IDC11367I	4	Summary
IDC11373I	4	Summary
IDC11374I	4	Summary
IDC11375I	4	Summary
IDC21363I	8	Summary
IDC21364I	8	Execution
IDC21365I	8	Execution
IDC21372I	8	Execution; may be syntax
IDC31366I	12	Syntax
IDC31368I	12	Syntax
IDC31369I	12	Execution
IDC31370I	12	Execution; may be syntax
IDC31376I	12	Execution
IDC31377I	12	Execution

Figure 28. DIAGNOSE Messages

Note to Figure 28:

Condition Code:

0 Not an error condition; informational only
4 Possible error condition, processing continues
8 Error condition, processing continues
12 Severe error, processing terminates

DIAGNOSE Invocation Error Messages

Before evaluating the target object, DIAGNOSE detects errors caused by incorrect or misleading command syntax. Correct the syntax errors, and rerun the DIAGNOSE job.

The following are DIAGNOSE messages that result from incorrect command syntax.

IDC11361I THE FOLLOWING {INCLUDE|EXCLUDE} ELEMENTS WERE NOT ENCOUNTERED

Explanation: This message indicates that names given in the INCLUDE or EXCLUDE list were not found during processing. It can be caused by a misspelled name, a damaged VSAM volume data set (VVDS), or damage to the basic catalog structure (BCS).

For a BCS, check for a spelling error and consider executing DIAGNOSE against the VVDS. For a VVDS, check for a spelling error and consider executing DIAGNOSE against the BCS. If there is no spelling error, one or more entries from the INCLUDE/EXCLUDE list have not been found. This is an error only if the BCS or VVDS should contain the listed entries.

IDC31366I INPUT DATA SET IS NOT AN ICFCATALOG A VVDS

Explanation: This message indicates that the data set to be diagnosed is neither an integrated catalog facility catalog nor a VVDS of the type indicated in the DIAGNOSE command syntax. If the data set is an integrated catalog facility catalog, the syntax should be 'DIAGNOSE ICFCATALOG'; if the data set is a VVDS, the syntax should be 'DIAGNOSE VVDS'.

For a VVDS, the name of the data set should be SYS1.VVDS.Vnnnnnn, where nnnnnn is the volser of the VVDS volume. For a BCS (ICFCATALOG is coded), DIAGNOSE imposes no naming restrictions. This message results from a damaged volume table of contents (VTOC), a misspelled name, or an uncoded DSNAME. Ensure that a DSNAME is coded on any DD statements for VVDSs or BCSs.

IDC31368I CATALOG MAY NOT BE SPECIFIED WITH ICFCATALOG

Explanation: This message warns that 'DIAG ICFCATALOG INCL(CATALOG(...))' or 'DIAG ICFCATALOG EXCL(CATALOG(...))' should not be coded for a BCS scan. The CATALOG parameter can be coded only for a DIAG VVDS.

IDC31370I UNABLE TO OBTAIN INFORMATION ON {dsname|ddname}

Explanation: This message indicates that DIAGNOSE read the VTOC with the OBTAIN supervisor call (SVC 27) and that OBTAIN has failed. There are four possible causes:

- Spelling error
- Wrong volume specified on the DD card
- Entry is not in the VTOC
- VTOC is damaged

Make sure you have specified the name of an integrated catalog facility catalog, not a VSAM catalog.

IDC21372I xxxxxxxxxxxxxx IS NOT AN ICFCATALOG A VVDS

Explanation: This message indicates that a compare parameter specified an incorrect data set. This error can be caused by a spelling error or a damaged DSCB in the VTOC.

Execution Error Messages

Execution error messages are caused by unexpected conditions that are found during the analysis phase of DIAGNOSE processing when the input data set is being read. These messages can mean that the input data set is damaged. To correct these errors, perform a recovery procedure against the catalog. (See "Catalog Recovery Procedures" on page 133.) You may have to recover a BCS or a VVDS or both.

DIAGNOSE performs checking operations against the specified BCS or VVDS in the following order:

1. Check an entry or record format.
2. Check for any associations (in the BCS only).
3. Check for miscellaneous length and context.
4. Check for BCS and VVDS dependencies (if the COMPARE DD option is specified).

An error in any particular step of the checking analysis causes message IDC21364I to be issued. When an error in an entry is detected, subsequent checks against that entry are then bypassed, and processing continues with the next catalog entry.

You should note that an error in any step of the analysis may mask additional errors. For example, a format error may mask association errors.

The type of an error is also noted (through reason codes) in the message text. At the end of a DIAGNOSE run, message IDC21363I is printed and all entries with errors are listed along with their pertinent reason codes.

If 'INCLUDE' or 'EXCLUDE' is specified, association checking may be bypassed. (See "INCLUDE/EXCLUDE Parameters" on page 115.) The INCLUDE and EXCLUDE tailoring options are generally employed to bypass entries diagnosed in previous runs or entries that are badly damaged.

IDC21364I ERROR DETECTED BY DIAGNOSIS:
VVDS ENTRY: aaaa
ICFCAT
RECORD: kkkk
OFFSET: dddd
REASON: tttt

Explanation: You should use the reason code to determine the scope and nature of the damage found by DIAGNOSE in structural problems in either the BCS or VVDS. Consider executing another DIAGNOSE command to give an analysis of both the BCS and any VVDSs for the damaged entry. See reason code descriptions below to determine which action you need to take.

For additional information on catalog or VVDS record structure, see Chapter 2, "Integrated Catalog Facility Catalog Structure" on page 5, and Appendix A, "The Integrated Catalog Facility Catalog Cell Structure" on page 149.

The section below provides extended descriptions of the execution error message reason codes and their associated recovery procedures.

Reason Code	Message, Explanation, and Recovery Procedure
1	**CELL LENGTH IS ZERO** **Explanation**: Each record is composed of cells, each cell having a cell type and a cell length. The indicated entry has a cell length of zero, which is an error. This error may be caused by incorrect positioning (caused by some other error) or the length may actually be zero. Use **OFFSET** associated with IDC21364I to find the cell in error. **Recovery Procedure**: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.
2	**CELL TYPE NOT RECOGNIZED** **Explanation**: Various catalog records can only contain certain cell types. The record being analyzed contained a cell type that either did not belong in the record or was not a legal cell type. This error may be caused by an incorrect length in the preceding cell that resulted in incorrect positioning. **Recovery Procedure**: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.
3	**RECORD TYPE NOT RECOGNIZED** **Explanation**: The first cell in each record has a cell type field, which is also the record type (or ID). The ID of the record named is invalid for integrated catalog facility catalogs, and is therefore not recognized. **Recovery Procedure**: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

4 **UNEXPECTED RECORD ID ENCOUNTERED**

Explanation: While processing an entry, a cell type was encountered that signalled the beginning of a new record. This may be caused by an invalid length value or a damaged entry.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

5 **COMPONENT LENGTH IS ZERO**

Explanation: Component length should never be zero. There are certain entry types that do not use the field set aside for component length. DIAGNOSE, therefore, only checks for zero component length in the following entry types: cluster ('C'), data ('D'), index ('I'), alternate index (AIX) ('G'), and generation data set (GDS) ('H').

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

6 **CELL LENGTH TOO LARGE**

Explanation: A cell length was found that is inconsistent with component or record length. This may be caused by the preceding or current cell.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

7 **CELL LENGTHS SUM AND COMPONENT LENGTH DISAGREE**

Explanation: The sum of all cell lengths in a component was not equal to the component length; it should have been. Either the component length is wrong or one of the cell lengths is wrong.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

8 **REPEATING CELL NOT VALID**

Explanation: Certain cells such as volume cells may occur more than once in an entry; most cells may not. The indicated cell occurred more than once and is an invalid occurrence.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

9 **RECORD LENGTH INCORRECT**

Explanation: The record length at the front of the record is not the same as the length stored by VSAM record management.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

10 **INCOMPLETE EXTEND DETECTED**

Explanation: The EXTEND function did not execute to completion. The named entry was interrupted in the middle of an EXTEND/EOV operation.

Recovery Procedure: The data set should be recovered. Records existing in a data set prior to the aborted extent should still be accessible if the data set is opened for input only. These records can be retrieved by using the REPRO command.

When the records have been retrieved, the data set can be deleted or redefined, and the records reinserted by using the REPRO command.

11 **INCOMPLETE DELETE DETECTED**

Explanation: The delete function did not execute to completion. This may indicate existence of partial record structures in the BCS.

Recovery Procedure: Rerun the delete function against the data set to complete the deletion process.

12 **CATLG AND VVDS NAMES UNEQUAL**

Explanation: There are four name fields in a VVDS record. The BCS entry and the VVDS entry do not have precisely the same names (length fields must also be the same) for one of the following four VSAM volume record (VVR) fields:

- VVRBSENM—Record name
- VVRKEY—Subrecord name
- VVRCMPNM—Component name
- VVRCATNM—Catalog name

Recovery Procedure: If the catalog names disagree, remove the entries in the BCS by using the DELETE command with the NOSCRATCH option. At this point, if the VVR contains the desired catalog name, the data set can be cataloged into the desired BCS by using the DEFINE command with the RECATALOG option; otherwise, catalog the data set into the catalog indicated in the VVR. (It will be necessary to define this catalog if it does not exist.) The data set can then be cataloged into the desired catalog by using the REPRO command with the MERGECAT option for this entry.

If names other than catalog names disagree, and the VVR is correct, remove the entries from the BCS by using the DELETE command with the NOSCRATCH option.

The data set can now be recataloged using the DEFINE command with the RECATALOG option.

If the VVR is incorrect, remove the data set by using the EXPORT command. At this point, the data set can be imported using the IMPORT command with the desired NEWNAMES parameter.

13 **VVDS AND VTOC EXTENT SEQ. NO. UNEQUAL**

Explanation: VTOC sequence numbers start at 0; VVDS record sequence numbers start at 1. The VVR number should always be equal to the VTOC number plus 1. An exception to this rule is the VVR sequence number for key range data sets.

Recovery Procedure: See "Recovering Damaged VVDS Entries" on page 134.

14 **CATLG AND VVDS VOLFLG UNEQUAL**

Explanation: VOLFLG is a field in the VVDS record for the named entry. This field has prime and overflow indicators. The indicators in the BCS and VVDS, for the named entry, are not equal.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

15 **CATLG AND VVDS KEYS UNEQUAL**

Explanation: BCS and VVDS records both have high and low key fields. The fields and their lengths must be equal; they were not.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

16 **VVDS AND VTOC STARTING CCHH UNEQUAL**

Explanation: For 'Z' records (a primary VVR), the VTOC extents and VVR extents are compared and an extent mismatch is detected.

Recovery Procedure: See "Recovering Damaged VVDS Entries" on page 134.

17 **VTOC ENTRY NOT FOUND**

Explanation: The data set control block (DSCB) for the named entry could not be found in the VTOC.

Recovery Procedure: See "Recovering Damaged VVDS Entries" on page 134.

18 **VVDS ENTRY NOT FOUND**

Explanation: BCS entries for VSAM data sets have one or more VVDS records (VVRs). To find a VVR, DIAGNOSE starts with the control interval (CI) indicated in the BCS volume cell. If the search for that CI fails, the entire VVDS is scanned, looking for the correct VVR. The search fails if DIAGNOSE cannot find a VVR with the following three matching fields:

- Component name
- Key range qualifier, if present
- Record type ('Z' – primary; 'Q' – secondary)

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

19 **CATLG ENTRY NOT FOUND**

Explanation: When executing a DIAGNOSE VVDS command with the COMPARE option, the BCS that is named in the VVR is opened and a VSAM GET DIRECT command is issued for the given record name. After the record is read, a search for the needed component is begun. These VVR fields are used during the search:

- VVRBSENM–Record name

- VVRKEY—Subrecord name
- VVRCMPNM—Component name
- VVRCATnm—Catalog name

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

20 **ASSOCIATION NOT FOUND**

Explanation: Certain types of BCS entries may be paired with other BCS entries. This pairing of one record with another is called an "association." Associations between entries are connected by name and are indicated by an association cell in an entry. Reason code 20 occurs when an indicated association name cannot be found elsewhere in the BCS. Figure 29 illustrates an example of an association and its logical connections.

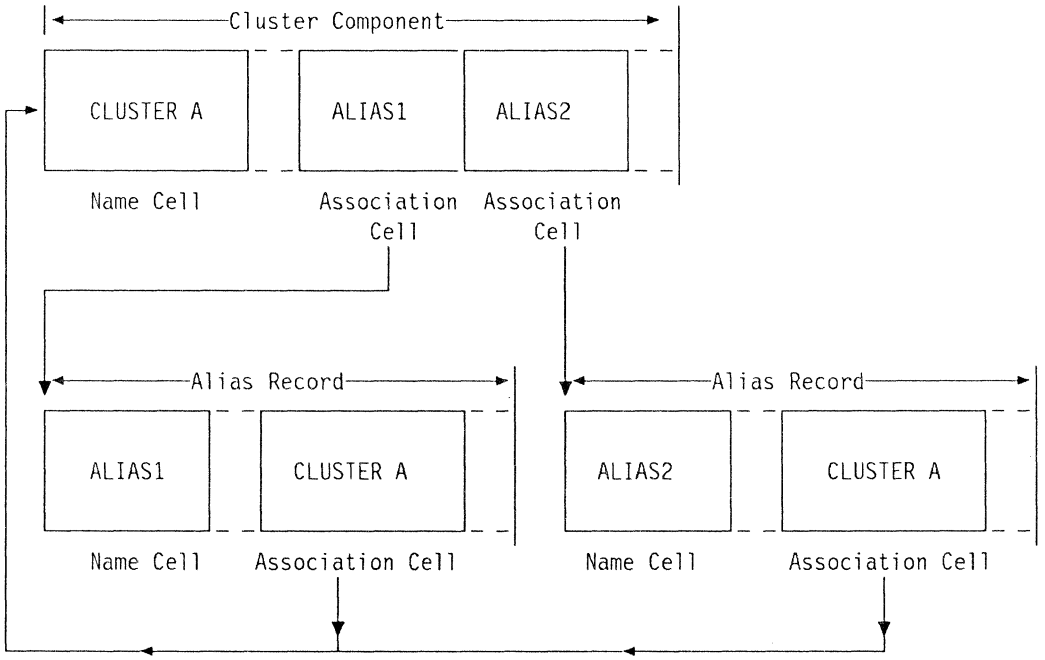


Figure 29. Example of an Association and Its Logical Connections

The following associations may occur:

- ALIAS ('X') entries with USERCATALOG ('U') connector, NONVSAM ('A'), or GDS ('H') entries.
- USERCATALOG ('U') connector, NONVSAM ('A'), or generation data set (GDS) ('H') entries with ALIAS ('X') entries.
- PATH ('R') entries with cluster ('C') or AIX ('G') entries.
- cluster ('C') or AIX ('G') entries with PATH ('R') entries.

Recovery Procedure: See "Recovering Damaged BCS Entries" on page 133.

21 **ASSOCIATION LOOP FAILURE**

Explanation: For an explanation of "association," see reason code 20. If an association can be found, but the association does not point back to the original entry, an association loop failure exists. For example, if a NONVSAM ('A') entry points to an ALIAS ('X') entry, then the ALIAS ('X') entry must point back to the NONVSAM ('A') entry.

Recovery Procedure: See "Recovering Damaged BCS Entries" on page 133.

22 **TRUENAME NOT FOUND**

Explanation: VSAM data ('D'), index ('I') and AIX ('G') components have implicit associations known as truenames. For an explanation of associations, see reason code 20. Truename associations are not described by an association cell because they are implicit. Reason code 22 is issued when a truename ('T') entry is not found.

Recovery Procedure: See "Recovering Damaged BCS Entries" on page 133.

23 **TRUENAME LOOP FAILURE**

Explanation: For an explanation of truename, see reason code 22. The truename ('T') entry must always point back to the original cluster ('C') component. If it does not, a truename loop failure exists.

Recovery Procedure: See "Recovering Damaged BCS Entries" on page 133.

24 **REQUIRED CELL MISSING, CELL TYPE**

Explanation: Depending on the component or entry type, certain cells may be required. The cell type that appears in the message is missing; it shouldn't be. For example, VSAM data ('D') and index ('I') entries must have a volume cell ('04'). If a data ('D') entry did not have a volume cell ('04'), reason code 24 would be issued.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

25 **UNUSED**

Explanation: This reason code is unused but reserved.

26 **CELL TYPE INVALID IN CONTEXT**

Explanation: The indicated cell is a valid cell type, but it is not a cell type that is valid for the type of entry being scanned. For example, although a volume cell ('04') is a valid cell, it may not occur in an ALIAS ('X') entry.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

27 **ENTRY MISSING FROM GAT CELL**

Explanation: Each GDS ('H') entry should be reflected in the generation aging table (GAT) cell ('05') of the generation data group (GDG) ('B') entry. Reason code 27 is the result of a GDS ('H') entry that is not reflected in the GAT cell.

Recovery Procedure: See "Recovering Damaged BCS Entries" on page 133.

28 **GAT CELL ENTRY NOT FOUND**

Explanation: Each entry in a GAT cell ('05') represents a GDS ('H') entry within the current GDG ('B') record. Reason code 28 results from a GAT cell entry for which there is no GDS ('H') entry within the current GDG ('B') record.

Recovery Procedure: See "Recovering Damaged BCS Entries" on page 133.

29 **ENTRY MISSING FROM REL CELL**

Explanation: Each AIX ('G') entry should be reflected in the RELation cell ('06') of the cluster ('C') record. Reason code 29 is the result of an AIX ('G') entry that is not reflected in the REL cell.

Recovery Procedure: See "Recovering Damaged BCS Entries" on page 133.

30 **REL CELL ENTRY NOT FOUND**

Explanation: Each entry in a RELation cell ('06') represents an AIX ('G') entry within the cluster ('C') record. Reason code 30 results from a REL cell entry for which there is no AIX ('G') entry within the current cluster ('C') record.

Recovery Procedure: See "Recovering Damaged BCS Entries" on page 133.

31 **UNUSED**

Explanation: This reason code is unused but reserved.

32 **UNUSED**

Explanation: This reason code is unused but reserved.

33 **INCOMPLETE UPDATE DETECTED**

Explanation: The current entry was undergoing a subrecord update/move operation that did not complete. This record or subrecord may be damaged.

Recovery Procedure: See "Recovering Damaged VVDS Entries" on page 134.

34 **VVDS AND VTOC ENDING CCHH UNEQUAL**

Explanation: Direct access storage device (DASD) extent information is kept in both the VTOC DSCB and the VVR. The VTOC and VVDS information should agree but do not.

Recovery Procedure: See "Recovering Damaged VVDS Entries" on page 134.

35 **VVDS AND VTOC EXTENT COUNTS UNEQUAL**

Explanation: DASD extent information is kept in both the VTOC DSCB and the VVR. The VTOC and VVDS information should agree but do not.

Recovery Procedure: See "Recovering Damaged VVDS Entries" on page 134.

36 **LENGTH OF NAME INVALID**

Explanation: Variable-length names occur in various places in both VVDS and BCS records. These names should have lengths ranging from 1 to 45. Incorrect length values may cause a variety of problems, including program checks and data overlays. A name with an incorrect length was found.

Recovery Procedure: Determine if the entry belongs to the BCS or VVDS, and see "Catalog Recovery Procedures" on page 133 for the appropriate recovery procedure.

IDC21365I ICFCATALOG RECORD DISPLAY: RECORD: kkkk
VVDS
VTOC

Explanation: This message appears because an error has been detected and 'DUMP' was coded or defaulted to on the DIAGNOSE command. Message IDC21365I should be used in conjunction with message IDC21364I which precedes it and details the error. The record displayed may be used in determining both the current recovery procedure and the extent of the damage. You may want to code NODUMP on the DIAGNOSE command to suppress any record displays. Information on record formats may be found in *Catalog Diagnosis Reference* or in Chapter 2, "Integrated Catalog Facility Catalog Structure" on page 5, and Appendix A, "The Integrated Catalog Facility Catalog Cell Structure" on page 149.

IDC31369I MAXIMUM ERROR LIMIT REACHED PROCESSING I/O ERROR ON INPUT DATA SET TRUNCATED

Explanation: Analysis of the BCS or VVDS has stopped because of the number of errors encountered during processing. The ERRORLIMIT parameter of the DIAGNOSE command may be coded to raise or lower the point at which processing stops. To ignore the number of errors, code ERRORLIMIT(0); the default value is 16. A badly damaged BCS or VVDS with ERRORLIMIT(0) may generate sizeable output.

An I/O error was encountered on either the BCS or VVDS during DIAGNOSE execution. This indicates that the BCS or VVDS has structural damage of such a nature that it cannot be read. The BCS or VVDS will need to be completely restored. This error may indicate a hardware error.

IDC31370I UNABLE TO OBTAIN INFORMATION ON {dsname|ddname}

Explanation: See "DIAGNOSE Invocation Error Messages" on page 120 for an explanation.

IDC31376I INPUT ICFCATALOG HAS NO VVDS ENTRIES

Explanation: DIAGNOSE processing has stopped because no VVDS (SYS1.VVDS.Vvolser) entries have been found in the input BCS. This indicates BCS damage, because there should be a VVDS entry for the volume on which the catalog resides.

Import the BCS from a backup copy to restore it. Then, using the appropriate access method services commands, bring the BCS up to date and rerun DIAGNOSE.

IDC31377I FIRST CATALOG ENTRY NOT FOUND

Explanation: Processing has terminated because DIAGNOSE has failed to position to the first BCS entry. This indicates that the self-describing cluster entry for the catalog is absent or damaged.

Import the BCS from a backup copy to restore it. Then, using the appropriate access method services commands, bring the BCS up to date and rerun DIAGNOSE.

IDC01371I RECORD DISPLAY SUPPRESSED, ALREADY DUMPED

Explanation: The record display that would normally appear after message IDC21364I has been suppressed because the record appeared earlier in the DIAGNOSE output. A record display may result in the display of more than one entry.

IDC11361I THE FOLLOWING {INCLUDE|EXCLUDE} ELEMENTS WERE NOT ENCOUNTERED

Explanation: See "DIAGNOSE Invocation Error Messages" on page 120.

Summary Messages

The summary DIAGNOSE messages listed below are produced by DIAGNOSE at the conclusion of the analysis phase and after the input data set has been read. These messages may be informational, may warn of a possible error, or may indicate that errors have occurred.

IDC01360I THE FOLLOWING ENTRIES HAD NO ERRORS

Explanation: This message results from using the LIST option with the DIAGNOSE command, and lists all entries without errors. IDC01360I and IDC21363I indicate all the entries processed: the first message lists entries with no errors; the second message lists entries with errors. Any entries not found in either list were *not* encountered. The value of this message lies in its use in determining what DIAGNOSE processed. Use this information to compile a list of processed entries. NOLIST is the DIAGNOSE default.

IDC11361I THE FOLLOWING {INCLUDE|EXCLUDE} ELEMENTS WERE NOT ENCOUNTERED

Explanation: See "DIAGNOSE Invocation Error Messages" on page 120.

IDC11362I THE FOLLOWING CATALOG REFERENCED VOLUMES WERE NOT ENCOUNTERED

Explanation: This may indicate a damaged BCS or VVDS and may have resulted from the use of INCLUDE or EXCLUDE causing entries to be skipped.

For a BCS:

There are entries in the BCS for volumes (VVDSs) that are not referenced (in the volume cell) by any of the entries processed. This indicates that certain entries in the BCS may be missing volume information (that is, missing volume cells).

To recover the missing volume information, recatalog, by using the DEFINE command with the RECATALOG option, any data sets that should reference the missing volume. After recataloging, any candidate volumes must be made available to the recataloged data set by using the ALTER command with the ADDVOLUMES parameter.

For a VVDS:

The VVDS has entries for referenced catalogs in the VVCR (the VVDS control record). None of the VVDS entries processed referenced the listed catalog names. There may be missing VVDS entries.

After you have determined which data sets are missing from the volume, use IMPORT to recover the data sets in their entirety. Multivolume data sets will require that all volumes be involved in the recovery, not just the affected volume.

IDC21363I THE FOLLOWING ENTRIES HAD ERRORS

Explanation: These entries had errors. The names listed here, along with the names listed under message IDC01360I, constitute all the entries processed. Any name listed under IDC21363I also has a corresponding IDC21364I message and reason code.

Use the information given with message IDC21364I to determine the nature and scope of the damage and the appropriate recovery procedure. Consider running

DIAGNOSE again to give an analysis of both the BCS and any VVDS(s) for the damaged entry.

IDC11367I THE FOLLOWING VVDS REFERENCED CATALOGS WERE NOT ENCOUNTERED

Explanation: The following integrated catalog facility catalog names are referenced by the VVDS as catalog entries but were not found in any entries for scanned VVDSs.

Run DIAGNOSE BCS for the extraneous catalog name. Determine whether data sets are missing from the volume. If they are not, then use the DELETE command with the NOSCRATCH option to remove the VVDS entry from the extraneous catalog. See summary message IDC11362I for more information.

THE FOLLOWING COMPARE ELEMENTS WERE NOT ENCOUNTERED

Explanation: The names that follow were expected to be encountered but were not. This message may result from a user error.

Check the spelling carefully and ensure that a DSNAME has been coded on the DD statement; or if a volser has been coded, ensure that it is correct.

For a BCS:

Determine whether any data sets in the catalog are missing volume entries. If so, recatalog these data sets by using the DEFINE command with the RECATALOG option to pick up missing volumes. If not, change the DIAGNOSE command to make no reference to the missing compare name.

For a VVDS:

Determine whether there are VVDS entries that should be referencing the named catalog. See message IDC11367I above for recommended action.

IDC11374I THESE ADDITIONAL CATALOG REFERENCED VOLUMES WERE ENCOUNTERED

Explanation: There are BCS entries referencing volumes (by means of a volume cell) for which the BCS has no record (that is, no SYS1.VVDS.Vvolser entry). This may indicate BCS damage. There should be a VVDS entry for the listed volume. Either the SYS1.VVDS.Vvolser entry is missing or the BCS entry is referencing a volume it no longer uses.

To add the volume entry, use the DEFINE command with the RECATALOG option for SYS1.VVDS.Vvolser.

To remove the volume entry, use the ALTER command with the REMOVEVOLUMES option for the entry referencing the extraneous volume.

IDC11375I THESE ADDITIONAL VVDS REFERENCED CATALOGS WERE ENCOUNTERED

Explanation: There are VVDS entries referencing catalogs for which the VVDS has no record (in the VVCR). This may indicate VVDS damage. There should be an entry in the VVDS control record (VVCR) for the listed catalog. Either this entry is misssing or a VVDS record is referring incorrectly to a catalog. Run DIAGNOSE against the BCS for the extraneous catalog.

To add the catalog name to the control record (VVCR), define a dummy data set by using the DEFINE command, and specify the catalog name needed in the

catalog parameter. Then delete the dummy data set by using the DELETE command; the catalog name will remain in the control record.

To remove the VVDS entry, if message IDC11375I is the result of an extraneous VVDS entry, use the DELETE command and specify VVR.

Catalog Recovery Procedures

This section describes recommended procedures for recovering damaged BCS entries and damaged VVDS entries. For additional information on catalog or VVDS record structure, see Chapter 2, "Integrated Catalog Facility Catalog Structure" on page 5, and Appendix A, "The Integrated Catalog Facility Catalog Cell Structure" on page 149.

Recovering Damaged BCS Entries

Three steps are used to recover a damaged BCS entry:

1. Remove the sphere or base record, if it exists.

   The damage detected may not be in a sphere or base record. If it is not, you will need to determine the sphere or base record key so that the sphere or base record can first be deleted. Truename and other association records contain the name of the sphere or base record in their association cell. An extension record belongs to the sphere record that has as its name the first 44 characters of the extension record name.

2. Remove any remaining association records.

   The DIAGNOSE command can be reexecuted after removing the sphere or base record to identify any unwanted truename or association entries in the BCS. These entries can subsequently be removed by using the DELETE command with the TRUENAME option.

3. Reintroduce the removed entries into the catalog.

   After the damaged entries have been removed, the data sets can be redefined. For VSAM data sets, the RECATALOG option of the DEFINE command should be specified.

Considerations of Generation Data Group Entries: The method of recovering damaged generation data group (GDG) entries is identical to that of recovering other damaged BCS entries, except that the generation data sets (GDSs) will need to be reintroduced into the catalog after the generation data group (GDG) has been redefined.

The current generations of the data set will need to be found and reintroduced into the generation data group (GDG) *in the proper order*. The LISTCAT command can assist in determining the current generation data sets (GDSs).

Example of a Recovery Process: Given the scenario outlined above, a sample recovery process should proceed as follows.

By using the DIAGNOSE command, an unwanted association named ALIAS.NONVSAM.DATASET has been found. With the base record not in the catalog, this association can now be removed.

```
//DELJOB   JOB
//DELSTEP  EXEC=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  DELETE                          -
       (ALIAS.NONVSAM.DATASET) -
       TRUENAME
/*
```

If these steps fail, delete the catalog with the DELETE command, specifying the RECOVERY option. The catalog can then be imported from a backup copy or redefined with the data sets recataloged.

Recovering Damaged VVDS Entries

Three steps are used to recover a damaged VVDS entry:

1. Remove the entries in the BCS for the data set, if they exist.

   Before the damaged VVRs can be removed, the entries in the BCS must be removed. See "Recovering Damaged BCS Entries" on page 133 for more details on removing BCS entries.

2. Remove the damaged VVRs from the VVDS.

   After the BCS entries have been removed, the VVRs can be removed by using the DELETE command specifying VVR. This will also remove the Format 1 DSCB from the device.

3. Restore the data set from a backup copy.

   The data set can now be reintroduced into both the BCS and VVDS by importing it from a backup copy.

If a backup copy of the data set does not exist and the data set can be opened, an attempt to recover some of the data may be made. Depending upon the extent and type of damage in the VVR, you may be unable to recover any data, or any data that has been recovered by you may be damaged or out of sequence.

Sample DIAGNOSE Output

Figure 30 on page 135 illustrates the messages produced by DIAGNOSE. This example analyzes a BCS and uses the following DIAGNOSE parameters:

- IFILE—Pass the BCS data set name on DD statement CATDD.
- CMPRDD—Compare BCS and VVDS data for entries which reference the VVDS specified by DD statement VVDSDD.
- LIST—List entries *without* errors as well as entries with errors.
- DUMP—If an entry is in error, display the record in which the entry resides.
- ELIMIT(1)—After one error, stop the DIAGNOSE of the BCS.

```
    /*******************************************/ -
    /*                                         */ -
    /* DIAGNOSE THE BCS  OF DIAGCAT1:          */ -
    /*   - PASS THE BCS NAME ON CATDD.         */ -
    /*   - COMPARE THE BCS AND THE VVDS OF     */ -
    /*      SYS1.VVDS.V333001,                 */ -
    /*   - LIST ENTRIES WITHOUT ERRORS AS      */ -
    /*      WELL AS ANY ENTRIES IN ERROR.      */ -
    /*   - DUMP ENTRIES IN ERROR.              */ -
    /*   - AFTER ONE ERROR, STOP THE DIAGNOSE. */ -
    /*                                         */ -
    /*******************************************/ -
    DIAG -
         ICFCAT -
              IFILE(CATDD ) -
                   CMPRDD(VVDSDD) -
                     LIST -
                     DUMP -
                     ELIMIT(1)

IDC21364I ERROR DETECTED BY DIAGNOSE:
  ICFCAT ENTRY: SAMPLE.KSDS1.DATA (D)
  RECORD: SAMPLE.KSDS1 /00
  OFFSET: X'0073'                                   <-- Offset to start of cell causing error
  REASON: 12 - CATLG AND VVDS NAMES UNEQUAL

IDC21365I ICFCAT RECORD DISPLAY:
  RECORD: SAMPLE.KSDS1 /00          <-- Length of record being displayed

000000  00DB0034 C3004600 2DE2C1D4 D7D3C54B   D2E2C4E2 F1404040 40404040 40404040   *....C....SAMPLE.KSDS1          *
000020  40404040 40404040 40404040 40404040   40404040 40000012 01E5E2C1 D4E3C5E2   *                    ....VSAMTES*
000040  E3408033 7F00000F 0019C400 490012E2   C1D4D7D3 C54BD2E2 C4E2F14B C4C1E3C1   *T ..".....D....SAMPLE.KSDS1.DATA*
000060  00001201 FFFFFFFF FFFFFFFF 0080337F   00000F00 1E0400F3 F3F3F0F0 F1305020   *..............".......333001.&.*
000080  09820000 00200000 00000000 00000000   00001AC9 004A0013 E2C1D4D7 D3C54BD2   *.....................I....SAMPLE.K*
0000A0  E2C4E2F1 4BC9D5C4 C5E70000 1201FFFF   FFFFFFFF FFFF0080 337F0000 0F001E04   *SDS1.INDEX.............".......*
0000C0  00F3F3F3 F0F0F130 50200982 00000020   00000000 00000000 000000             *.333001.&...................    *

IDC21365I VVDS RECORD DISPLAY:
  RECORD: X'00002000'

000000  01170040 E9000000 000012E2 C1D4D7D3   C54BD2E2 C4E2F14B C4C1E3C1 000DE2C1   *... Z.....SAMPLE.KSDS1.DATA..SA*
000020  C1C1C1C1 4BD2E2C4 E2F10008 C4C9C1C7   C3C1E3F1 0DE2C1D4 D7D3C54B D2E2C4E2   *AAAA.KSDS1..DIAGCAT1.SAMPLE.KSDS*
000040  F1000035 21402000 00000600 00000100   00018000 00000000 00280000 00004400   *1....  .....................D...*
000060  00FFFFFF FFFFFFFF FF000000 00000000   00000000 00000000 62608000 60000000   *.....................-.-.....*
000080  00004000 00001400 00000000 00000002   00000000 64000000 00000000 00000000   *.....................-..-.....*
0000A0  00000000 00000000 00000000 00000000   00000000 01000000 00000000 00000000   *.....................-..-.....*
0000C0  00000000 00000000 00000028 00000000   00000000 00003000 00003E23 80010000   *.....................-..-.....*
0000E0  00000000 00000008 28000000 02000014   00014000 13000000 00000000 00000000   *.....................-..-.....*
000100  00001400 01000000 06000000 06000100   00000000 0027FF                      *.....................-..-.....*

 IDC31369I MAXIMUM ERROR LIMIT REACHED, PROCESSING TRUNCATED
                                                    <-- Result of "ELIMIT(1)"
IDC01360I THE FOLLOWING ENTRIES HAD NO ERRORS:

  00000000000000000000000000000000000000000000 (C)
  DIAGCAT1 (D)
  CATINDEX.T81EF870.VID80337.T9127F1D (I)
  CATINDEX.T81EF870.VID80337.T9127F1D (T)
  DIAGCAT1 (T)
  SAMPLE.KSDS1 (C)
                                                    <-- Result of "LIST"

IDC21363I THE FOLLOWING ENTRIES HAD ERRORS:

  SAMPLE.KSDS1.DATA (D) - REASON CODE: 12
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 8

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 8
```

Figure 30. Sample DIAGNOSE Output

Analysis of the DIAGNOSE Sample Output

Analysis of the sample output is based primarily on the presence of these messages:

- IDC21363I—There are entries with errors.
- IDC21364I—The entry in error is SAMPLE.KSDS1.DATA, the data component of a VSAM cluster.
- IDC21365I—The cluster is SAMPLE.KSDS1.
- In the sample program, two additional messages were received: IDC31369I and IDC01360I.
  - IDC31369I—This indicates that processing stopped because the error limit had been reached.
  - IDC01360I—This lists successfully processed entries.

IDC21363I THE FOLLOWING ENTRIES HAD ERRORS

Explanation: This summary message tells us that DIAGNOSE has detected one or more errors. Because ELIMIT(1) was coded, DIAGNOSE stopped reading the BCS after one error (message IDC31369I is also printed). The entry in error is SAMPLE.KSDS1.DATA, a VSAM data component, and the associated reason code is 12.

By using the information about message IDC21363I, which appears under the section "Interpreting DIAGNOSE Output," we can determine that reason code 12 is a compare error, signifying that the BCS and VVDS do not agree. If we are not able to determine that the error is confined to the BCS, we will import the data set to recover from this error. See "Catalog Recovery Procedures" on page 133 on how to recover damaged VVDS entries.

IDC21364I ERROR DETECTED BY DIAGNOSE:
ICFCAT ENTRY: aaaa
RECORD: kkkk
OFFSET: dddd
REASON: tttt

Explanation: The error detected is 'CATLG AND VVDS NAMES UNEQUAL', and the associated reason code is 12. The offset provided points us to the beginning of a cell type '04', a volume cell. By looking at the cell, we determine that the volume being processed was '333001' and thus the VVDS is 'SYS1.VVDS.V333001'. We now know that the entry for 'SAMPLE.KSDS1.DATA' in the BCS does not agree with the VVDS entry in 'SYS1.VVDS.V333001'.

An inspection of the VVDS entry shows that the name 'SAMPLE.KSDS1' appears as 'SAAAAA.KSDS1'. To recover, we must correct the VVDS record. This is done by importing the data set 'SAMPLE.KSDS1'. See "Catalog Recovery Procedures" on page 133 on how to recover damaged VVDS entries.

IDC21365I ICFCATALOG RECORD DISPLAY: RECORD: kkkk VVDS

Explanation: The record displays provided may be used in conjunction with the information supplied with message IDC21364I. If we are not familiar with the format of integrated catalog facility records or want to concern ourselves with this problem at a higher level, we should use message IDC21363I for problem analysis.

IDC31369I MAXIMUM ERROR LIMIT REACHED PROCESSING TRUNCATED

Explanation: This message informs us that processing did not reach a normal conclusion, but was terminated prematurely because the ELIMIT of 1 was reached. There is, therefore, one entry under message IDC21363I and one entry under message IDC21364I. Message IDC01360I should be consulted to determine which entries, if any, had no errors.

IDC01360I THE FOLLOWING ENTRIES HAD NO ERRORS

Explanation: This message resulted from the LIST parameter in the DIAGNOSE command, which lists the entries that successfully passed the various DIAGNOSE checks. Notice that the cluster entry 'SAMPLE.KSDS1' is correct, although the data component has an error. Notice also that the index component was not analyzed because of ELIMIT(1).

Chapter 7. Communicating with Catalog Address Space (CAS)

MODIFY CATALOG

System programmers can communicate with Catalog Address Space (CAS) via CATALOG options of the system command MODIFY. Using these options, you can directly control some of the recovery capabilities designed into CAS and explore additional recovery possibilities. These options are particularly useful for providing transparent recovery in cases that may have previously required a system IPL or the cancellation of a user job. In addition, these options can help you diagnose problems by providing information about CAS and its environment.

While they are processing, some of the CATALOG options require control of specific system resources. For example, MODIFY CATALOG,LIST requires the CAS local lock. If the resource required by the MODIFY routine is not available, the CAS MODIFY subtask routine waits a limited time for the resource. If the request is not completed in the allotted time, the CAS MODIFY subtask abends with ABENDA1A. A different MODIFY subtask can then be attached. This abend prevents the MODIFY subtask from being lost for further processing.

The CAS MODIFY subtask is protected by its own ESTAE routine. The MODIFY subtask is also covered by an End of Task Exit Routine (ETXR), which attaches a new MODIFY task to replace the terminating one so that the MODIFY function can continue to be available.

A complete description of the MODIFY command can be found in *System Commands*. The output messages are defined in *System Messages, Volume 2*. Summaries of the options for communicating with CAS as well as examples and special considerations are presented in this chapter.

The MODIFY command can be entered at any console which can submit operator commands and also via JCL. CAS will only accept one MODIFY command at a time. If the CAS MODIFY task is active and another MODIFY command is entered, the second command will be rejected.

MODIFY CATALOG Command Formats and Examples

MODIFY CATALOG,ABEND

This option terminates a CAS task. The task is abended with ABEND91A, and any catalog request in process at the time of the abend is redriven one time. See the MODIFY CATALOG,END command format for unlimited redrive capability.

When the task ID is known, the END command format would be the preferred method of terminating CAS service task processing.

Example 1: If a CAS task is involved in an ENQ lockout, you can use this option to terminate the CAS service task, and release its resources. The timing of the resource acquisition on the redrive will probably be different, and the simultaneous request of resources (which caused the original lockout) may not occur. If the redrive of the request results in the same ENQ lockout, you can then cancel the job.

Example 2: If a CAS task goes into a wait instead of continuing normally, you can use this option to terminate that task, and the redrive of the catalog request may be successful. This action may avoid the cancellation of a job.

The MODIFY CATALOG,ABEND formats are:

```
MODIFY CATALOG,ABEND(ID)
MODIFY CATALOG,ABEND(00000000)
MODIFY CATALOG,ABEND(MODIFY)
```

Where:

ID = The task ID.

00000000 = CAS task TCB address.

MODIFY = To cause CAS modify task (IGG0CLGA) to terminate and a new CAS modify task to be attached.

MODIFY CATALOG,CLOSE

This option closes an integrated catalog facility catalog dynamically, without affecting any existing allocations. All of the CAS private storage associated with the catalog is freed. The catalog will be reopened with a new set of control blocks the next time a request is processed for that catalog. The rebuilding of the control blocks is transparent to the users of the catalog. You can dynamically change catalog attributes such as share options and the number of strings with this option. In the past, changing these attributes would have required an IPL (for the master catalog) or the termination and restart of a job or online system (for a user catalog).

Example 1: CAS can determine that damaged integrated catalog facility control blocks are probably causing some abnormal occurrences. In these cases, CAS causes an automatic rebuilding of the control blocks, and the rebuilding is transparent to users of the system. Some situations are beyond CAS's detection capability, but you can detect them. For example, you may determine that damaged in-storage catalog control blocks are probably causing recurrent abends when accessing a particular catalog. Using this command, you can initiate the rebuilding process.

Example 2: Assume that in your system, once the system is up and all data bases are opened, there is little or no catalog activity. This command format could be used to free all CAS private storage associated with the catalogs.

The MODIFY CATALOG,CLOSE format is:

MODIFY CATALOG,CLOSE(NNNNNN...)

Where:

NNNNNN = integrated catalog facility catalog name.

MODIFY CATALOG,END

This option is the preferred method of ending the processing of a CAS service task. This option terminates a CAS service task. The task is abended with ABEND91A (REDRIVE option) or ABEND71A (NOREDRIVE option). With the REDRIVE option, any catalog request in process at the time of the abend is redriven an unlimited number of times. With the NOREDRIVE option, any catalog request in process at the time of the abend is failed with catalog RC246.

Example 1: If a CAS task is involved in an ENQ lockout, you can use this option to terminate the CAS service task, and release its resources. The timing of the resource acquisition on the redrive will probably be different, and the simultaneous request of resources (which caused the original lockout) may not occur. If the redrive of the request results in the same ENQ lockout, you can then cancel the job. A resource monitor program may tell you that CAS is holding the critical resource involved in an ENQ lockout, and that the CAS task is processing on behalf of a user jobname. The command format MODIFY CATALOG,LISTJ(jobname) will provide you with the task ID to be used in this command.

Example 2: If a CAS task goes into a wait instead of continuing normally, you can use this option to terminate that task, and the redrive of the catalog request may be successful. This action may avoid the cancellation of a job. The command format MODIFY CATALOG,LIST will provide you with the task ID to be used in this command.

The MODIFY CATALOG,END formats are:

MODIFY CATALOG,END(ID)
MODIFY CATALOG,END(ID),REDRIVE
MODIFY CATALOG,END(ID),NOREDRIVE

Where:

ID = The task ID.

REDRIVE = The default. REDRIVE will cause the catalog request currently being processed by the service task to redrive under a different task (transparent to the user).

NOREDRIVE = Will cause the catalog request currently being processed by the service task to be failed with catalog return code 246 (RC246).

MODIFY CATALOG,ENTRY

This option provides the starting addresses, the FMID's, and the PTF/APAR levels of all the modules in catalog load modules IGG0CLX0 (resident in CAS) and IGG0CLHA (resident in the link pack area). The output message IEC349I displays the information. The output of this command is probably best viewed on the system log due to its size, if all entry points are requested.

Example 1: You can use this option to provide the storage address of one specific catalog module for a Serviceability Level Indication Processing (SLIP) trap. For more information on SLIP traps see SYSTEM COMMANDS.

The MODIFY CATALOG,ENTRY formats are:

```
MODIFY CATALOG,ENTRY(MMMMMMMM)
MODIFY CATALOG,ENTRY
```

Where:

MMMMMMMM = CSECT name.

If CSECT name is omitted, all CSECTs are listed.

MODIFY CATALOG,LIST

This option lists currently active CAS service tasks, their related jobnames, their elapsed time, and unique ID. The output message IEC347I displays the information.

Example 1: Assume you have an ENQ lockout condition. Your resource monitor program indicates that the resource essential to the lockout is being held by a CAS task on behalf of a user jobname. Using the command MODIFY CATALOG,LIST(jobname) you can determine the ID of the CAS service task. Then using the command format MODIFY CATALOG,END(id) you can terminate the CAS task to release the resource and redrive the user catalog request.

The MODIFY CATALOG,LIST formats are:

```
MODIFY CATALOG,LIST
MODIFY CATALOG,LIST(ID)
MODIFY CATALOG,LIST(00000000)
MODIFY CATALOG,LIST(jobname)
```

Where:

ID = The task ID

00000000 = The task TCB address. Either the task ID or TCB address can be used to list one single task.

jobname = List all CAS service tasks currently active for the specified jobname.

MODIFY CATALOG,OPEN

This option lists the name, volume serial number, current allocation count, and various status flags for every catalog currently allocated on the system. The output message IEC348I displays the information.

Example 1: In recovering a volume, you can use this option to determine which catalogs are allocated on that volume and how many users are currently allocated to those catalogs. You can use the command formats MODIFY

CATALOG,CLOSE(catname), and MODIFY CATALOG,VCLOSE(volser) to close the catalogs and the VVDS on that volume.

The MODIFY CATALOG,OPEN formats are:

```
MODIFY CATALOG,OPEN
MODIFY CATALOG,OPEN(VVVVVV)
```

Where:

VVVVVV = Volser which can be used to limit the list to allocated catalogs on a specific volume.

MODIFY CATALOG,REPORT

This option lists various CAS status fields. The output message IEC359I displays the information. The values displayed for 'ALIAS LEVELS', "SYS% TO SYS1 CONVERSION', and 'SERVICE TASK LOWER LIMIT' can be set automatically at IPL via the SYSCATnn member of SYS1.NUCLEUS.

The MODIFY CATALOG,REPORT format is:

```
MODIFY CATALOG,REPORT
```

MODIFY CATALOG,RESTART

This option restarts CAS in a new address space. The CAS mother task is abended with ABEND81A, and any catalog requests in process at the time are redriven.

The restart of CAS in a new address space is transparent to all users. However, even though all requests are redriven successfully and receive a return code 0, the system may produce indicative dumps on the console, the system log, and on user job logs. Currently, there is no way to suppress these indicative dumps.

CAS is designed to recover from cross-memory failures that can occur during CAS restart. CAS recognizes and recovers from the following abend codes, which may occur during CAS restart: ABEND052, ABEND058, ABEND066, ABEND070, ABEND073, and ABEND0Dx. You can ignore any indicative dumps produced by the system for these abend codes. Only the final catalog return code, which should be 0, is significant.

Example 1: If CAS address space has a storage shortage due to freemain failures, you can use this option to restart CAS in a new address space.

Example 2: If any of the other options cannot resolve a catalog-caused system failure, you can use this option to restart CAS in a new address space. This option may eliminate the need to do a system IPL or to terminate and restart some online system.

The MODIFY CATALOG,RESTART format is:

```
MODIFY CATALOG,RESTART
```

MODIFY CATALOG,VCLOSE

This option closes a VVDS without affecting any existing allocations. The next time a request is processed for that VVDS, the VVDS will be reopened with a new set of control blocks.

Example 1: You determine that in-storage control block damage may be causing recurrent abends when accessing a particular VVDS. This option allows you to initiate the control block rebuild process.

The MODIFY CATALOG,VCLOSE format is:

MODIFY CATALOG,VCLOSE(VVVVVV)

Where:

VVVVVV = The volume serial number.

MODIFY CATALOG Messages

IEC347I LIST ACTIVE CATALOG TASKS

Explanation: MODIFY command was addressed to Catalog Address Space requesting a listing of active CAS service tasks.

Problem Determination: This is an informational message.

User Response: None.

Operator Response: None.

Programmer Response: None.

System Action: Multi-line Write to Operator giving hexadecimal address of each CAS service task and the JOBNAME for which the task is currently processing (or NONE if service task not currently processing a catalog request). The ELAPSED TIME (HH.MM.SS) that the request has been active in CAS is displayed. The CAS task ID is displayed. This ID can be used in the command format MODIFY CATALOG,LIST(00) to display one specific task, and also in the command format MODIFY CATALOG,END(00) to terminate one specific CAS service task. When a list of ALL active CAS tasks is requested (MODIFY CATALOG,LIST), and more than one active task is listed, the oldest request active in CAS is indicated (O). This indication is not given for any other form of LIST. When the CAS task is waiting (e.g., for ENG) as indicated by a nonzero CCXCASST field, the wait status is indicated (W). Abending task is indicated (A). ENQ wait is indicated (E). Recall wait is indicated (R).

IEC348I ALLOCATED CATALOGS

Explanation: MODIFY command was addressed to Catalog Address Space requesting a listing of open catalogs. "C" indicates that the integrated catalog facility catalog has been "closed" to free record management storage in CAS. Catalog will be "opened" at the next access with a new set of record management control blocks. Catalog allocation is unaffected. CAXWA, ACB, and RPL storage in CSA is also unaffected. The catalog may have been closed via the modify command format of F CATALOG,CLOSE(nnnnn...), or it may have been closed by CAS recovery functions to cause control block rebuild and/or free CAS resources. "D" indicates that catalog has been deleted. The indicators "C" and "D" are mutually exclusive. The user count (hex) gives the number of times the

catalog has been allocated. vvvvvv = volser, nnnnnn = catalog name (up to 44 characters).

Problem Determination: This is an informational message.

User Response: None.

Operator Response: None.

Programmer Response: None.

System Action: Multi-line Write to Operator giving volume serial, use count, and name of each allocated catalog.

Note: User count is in hex.

IEC349I DISPLAY ENTRY POINT

Explanation: MODIFY command was addressed to Catalog Address Space requesting a display of module(s) entry point address(es).

mmmmmmmm = module name
00000000 = hexadecimal starting address
ffffff = FMID value
llllll = maintenance level

Problem Determination: This is an informational message.

User Response: None.

Operator Response: None.

Programmer Response: None.

System Action: Multi-line Write to Operator giving hexadecimal address(es) of the module(s) requested, the module name(s), and maintenance level.

IEC350I CATALOG ADDRESS SPACE MODIFY COMMAND AVAILABLE

Explanation: MODIFY command interface task for Catalog Address Space has been initialized and is ready to accept MODIFY commands.

Problem Determination: This is an informational message.

User Response: None.

Operator Response: None.

Programmer Response: None.

System Action: None.

IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE

Explanation: Command has been directed to the Catalog Address Space. The CAS MODIFY task has received the command.

Problem Determination: This is an informational message.

User Response: None.

Operator Response: None.

Programmer Response: None.

System Action: None.

IEC352I CATALOG ADDRESS SPACE MODIFY COMMAND COMPLETED

Explanation: MODIFY command has been directed to the Catalog Address Space. The CAS MODIFY task has completed the requested function.

Problem Determination: This is an informational message.

User Response: None.

Operator Response: None.

Programmer Response: None.

System Action: Requested function is performed. If restart of CAS was requested, the CAS jobstep task is abended with ABEND81A. CAS is then restarted. If abend of service task was requested, the service task is abended with ABEND91A.

IEC353I CATALOG ADDRESS SPACE MODIFY UNSUCCESSFUL

Explanation: MODIFY command has been directed to the Catalog Address Space. The CAS MODIFY task is unable to process the request since the format, keyword, or address input is invalid, input catalog name is not open, or input volser does not contain an open VSAM Volume Data Set (VVDS).

Problem Determination: This is an informational message.

User Response: None.

Operator Response: Enter MODIFY command with corrected format, keyword, or address.

Programmer Response: None.

System Action: None.

IEC354I CATALOG ADDRESS SPACE INPUT COMMAND NOT MODIFY

Explanation: Command has been directed to the Catalog Address Space. The CAS MODIFY task is unable to process the request since the command was not MODIFY.

Problem Determination: This is an informational message.

User Response: None.

Operator Response: Enter MODIFY command to request catalog function.

Programmer Response: None.

System Action: None.

IEC359I CATALOG REPORT OUTPUT

Explanation: MODIFY command has been directed to the Catalog Address Space. General information CAS status report is produced.

ffffffff = FMID value
aaaa = current ASID for CAS
llll = service task lower limit (constant set at IPL)
hhhh = service task high water mark (never less than lower limit)
cccc = service tasks currently attached (may be less than lower limit if CAS has not yet processed llll number of requests or if tasks have abended. CAS will create new tasks as requests are processed to maintain lower limit. CAS will

gradually reduce the number of tasks to maintain lower limit if the rate of catalog requests decreases)

Problem Determination: This is an informational message.

User Response: None.

Operator Response: None.

Programmer Response: None.

System Action: None.

MODIFY CATALOG Abend Codes

ABEND81A

Explanation: Abend of CAS mother task (Jobstep task IGG0CLX0). Requested by MODIFY command to restart CAS. All catalog requests in process are redriven.

Programmer Response: None.

ABNED91A

Explanation: Abend of CAS service task. Requested by MODIFY command to restart service task. Catalog request in process is redriven.

Programmer Response: None.

ABENDA1A

Explanation: Abend of CAS MODIFY task IGG0CLGA. MODIFY task exceeded allowed timer interval for completion of requested MODIFY function. The time allowed to the MODIFY task is limited to prevent loss of the function due to unavailable system resources. A new task will be ATTACHed to restore MODIFY function.

Programmer Response: None.

ABENDC1A

Explanation: Abend of CAS MODIFY task IGG0CLGA was requested via MODIFY command. A new task will be ATTACHed to restore the specified CAS function. This abend is used to terminate the specified task when requested via MODIFY CATALOG,ABEND(MODIFY) command.

Programmer Response: None.

Appendix A. The Integrated Catalog Facility Catalog Cell Structure

Basic Catalog Structure (BCS) Cells

The cell is the smallest block of information and may contain the name, volume, owner, security, and association information. Related cells become components and may be data or index components or GDGs. The related components may become a cluster or alternate index. The following cells are found in the BCS:

AIX Name (Type 'G') – identifies an alternate index (AIX) and contains:

- Flag: AIX is member of the upgrade set
- AIX condensed key

Alias Name (Type 'X') – identifies an alias of a non-VSAM data set (or GDG base) and contains the alias name.

Association (Type X'03') – identifies and contains the condensed association keys:

- Paths associated with a cluster or AIX
- Cluster and/or AIX associated with a path
- Aliases associated with a non-VSAM data set
- Data set associated with an alias
- Base cluster, and AIX if any, associated with a true name

Cluster Name (Type 'C') – identifies a VSAM base cluster and contains the cluster name.

Data Name (Type 'D') – identifies a data component of a base cluster or AIX and contains:

- Interruption recognition flags:
  - Delete-in-progress
  - Update-extend in progress
  - Subrecord move and update in progress
- Condensed key

GDG Extension Name (Type 'J') – identifies an extension record for a GDG and has the same format as the VSAM extension name cell.

Generation Aging Table (Type X'05') – identifies data sets in a generation data group and contains:

- Attribute flags:
  - Delete oldest/all GDS entries when GDG limit exceeded
  - Do not/do scratch F1 DSCB of DASD GDS if mounted
- Maximum GDS entries in GDG base
- For each current GDS:
  - Generation number of GDS
  - Version number of GDS

Generation Data Group Name (Type 'B') – identifies and contains the name and a pad character.

Generation Data Set Name (Type 'H') – identifies a non-VSAM data set that is part of a generation data group and contains:

- Generation number of GDS
- Version number of GDS

Integrated Catalog Facility Connector Name (Type 'U') – identifies a VSAM or integrated catalog facility user catalog connector record and contains:

- Catalog type flag: VSAM/integrated catalog facility
- Catalog name

Index Name (Type 'I') – identifies an index component of a base cluster or AIX and has the same format as the data name cell.

Non-VSAM Name (Type 'A') – identifies a non-VSAM data set and contains the non-VSAM name.

Ownership (Type X'01') – identifies the ownership information for a component and contains:

- Owner identification
- Flags:
  - RACF protection
  - Index component data set
  - Reusable data set
  - Erase specified (cluster only)
  - Swap space (cluster only)
  - Page space (cluster only)
- Creation date
- Expiration date
- Creation century
- Expiration century

Path Name (Type 'R') – identifies a path name of a base cluster or AIX and contains:

- Path attribute flags:
  - Upgrade if related to AIX
  - Update if related to cluster
  - Relation: AIX/cluster
- Path name

Relation (Type X'06') – identifies AIXs associated with a base cluster and contains, for each AIX:

- Upgrade AIX flag
- AIX condensed key

Security (Type X'02') – identifies security information for a VSAM component and contains:

- Passwords
- Password prompting code
- Maximum number of attempts

- User security verification routine name
- User authorization record length
- User security verification data length
- User security verification data

True Name (Type 'T') – identifies a data, index, or AIX component that is not the first component of a record and contains the data, index, or AIX name.

Volume (Type X'04') – identifies volume information for a VSAM or non-VSAM data set and contains:

- Volume serial number
- Device type
- Flags:
  - Prime volume—allocated space
  - Candidate volume no space
  - Overflow key range only
  - Converted VSAM data set volume
  - Non-VSAM volume cell
  - Key range qualifier present
  - Primary VVR CI
  - Sequence set with data
- Relative byte address of VVR
- Non-VSAM F1 DSCB TTR
- Non-VSAM file sequence no. (tape)
- Key range qualifier (VSAM only)
- Low key range key
- High key range key

VSAM Extension Name (Type 'E') – identifies an extension record for a VSAM cluster and contains:

- Extension key
- Sphere name

Note: For the complete format of the cells described above, see *Catalog Diagnosis Reference*.

VSAM Volume Data Set (VVDS) Cells

VVR Header – identifies a primary and secondary VVRs and contains:

- Length of entire VVR
- Length of header cell
- Type of VVR
  - Z (primary VVR)
  - Q (secondary VVR)
- Flags:
  - Self-describing VVR for VVDS
  - Catalog self-describing VVR
  - Data/index component type
- Key range qualifier
- Component name length
- Component name
- Cluster name length
- Cluster name

- Catalog name length
- Catalog name
- Base cluster name length
- Base cluster name

VVR Data Set Information – identifies attributes of a VSAM data component or index component and contains:

- Attribute flags:
  - Speed
  - Unique
  - Reusable
  - Erase
  - Inhibit update
  - Temporary export
  - Track overflow
  - Share attributes
  - Internal system data set
  - Component not usable
- Open indicator
- Minimum buffer size
- Primary space allocation
- Secondary space allocation
- Space option flags
- Data set high used RBA
- Data set high allocated RBA
- Average logical record length
- Exception exit
- Data set high key RBA
- Cluster attribute flags:
  - Timestamps exist in this cell
  - Verify required (catalog only)
  - Cluster describes swap space flag
  - Cluster describes page space flag
- Upgrade AIX flag
- Timestamp (catalog only)
- Alias table timestamp

VVR AMDSB – identifies the access method data statistics block of a VSAM data component or index component and contains:

- Attribute flags:
  - KSDS/ESDS/LDS
  - Write check
  - Sequence set with data
  - Replication
  - Use volumes in list order
  - Key range data set
  - RRDS
  - Spanned records are allowed
  - Nonunique/unique keys
  - Cylinder fault MSS or stage
  - Bind/nobind on MSS
  - Wait/nowait on relinquish
  - Load mode/data set loaded
- Alternate key RKP

- RKP
- Key length
- % Free CI in CA
- % Free bytes in CI
- Number of CI's per CA
- Free CI's per CA
- Free bytes per CI
- CI size
- Maximum record size
- RBA of high level index record
- Record slots per CI
- RBA of first sequence set record
- Max relative record number
- Pointer to first ARDB
- Number of concurrent requests
- Number of index buffers
- Number of data buffers
- System time stamp
- Number of index levels
- Number of extents
- Number of logical records
- Number of deleted records
- Number of inserted records
- Number of updated records
- Number of retrieved records
- Bytes of free space in data set
- Number of CI splits
- Number of CA splits
- Number of EXCPs

VVR Volume Information – identifies information for a particular volume for a VSAM data component or index component and contains:

- Flags:
  - Prime volume
  - Extents in VVR out of sync with DSCB
  - Overflow volume
- Number of extents on volume for data set
- High-key RBA
- High-used RBA
- High-allocated RBA
- Block size
- Number of blocks per track
- Tracks per allocation unit
- Extent type flags:
  - Sequence set with data
  - Extents not preformatted
  - Converted VSAM data set volume
- Tracks per cylinder
- Bytes per track
- Bytes per allocation unit
- Low key length
- Low key on volume
- For each extent:
  - Starting CCHH
  - Ending CCHH
  - Number of tracks
  - Starting RBA
  - Ending RBA

Appendix B. Sample Conversion From VSAM to Integrated Catalog Facility Catalog

This sample job stream is used to convert a VSAM catalog to an integrated catalog facility catalog.

```
//STEP01  EXEC PGM=IDCAMS

//*************************************************************
//*  DEFINE AN INTEGRATED CATALOG FACILITY CATALOG            *
//*************************************************************

//SYSPRINT DD   SYSOUT=A
//VOL      DD   VOL=SER=333801,UNIT=3380,DISP=OLD
//SYSIN DD *
         DEFINE USERCATALOG    -
                (NAME(ICFCAT1)    -
                 CYLINDERS(1 1)    -
                 VOLUMES(333801)    -
                 MASTERPW(ICFPW)  -
                 FREESPACE(10 10)  -
                 RECORDSIZE(4086 4086)   -
                 ICFCATALOG  -
                 FILE(VOL))  -
                 CATALOG(ICFMCAT/MPW1) /*master catalog/master
                                         password*/

/*
//STEP02  EXEC PGM=IDCAMS

//*************************************************************
//*  CONVERT FROM VSAM TO INTEGRATED CATLOG FACILITY CATALOG   *
//*************************************************************

//SYSPRINT DD   SYSOUT=A
//VOL      DD   VOL=SER=333801,UNIT=3380,DISP=OLD
//SYSIN DD *
         CNVTCAT -
                INDATASET(VSAMCAT1/VSAMPW) -
                OUTDATASET(ICFCAT1/ICFPW) -
                FILE(VOL)
/*
```

```
//STEP03  EXEC PGM=IDCAMS
//*****************************************************************
//*  LIST THE CATALOG                                             *
//*****************************************************************

//SYSPRINT DD   SYSOUT=A
//VOL      DD   VOL=SER=333801,UNIT=3380,DISP=OLD
//SYSIN DD *
         LISTCAT -
                ALL -
                FILE(VOL) -
                CATALOG(ICFCAT1/ICFPW)  /*user catalog*/
/*
//STEP04  EXEC PGM=IDCAMS

//*****************************************************************
//*  DEFINE THE ALIASES                                           *
//*****************************************************************

//SYSPRINT DD   SYSOUT=A
//VOL      DD   VOL=SER=333801,UNIT=3380,DISP=OLD
//SYSIN DD *
         DEFINE ALIAS -
           (NAME(.......) -
            RELATE(ICFCAT1))
```

Appendix C. Alternate Master Catalog Job Stream

This sample job stream is used to produce an integrated catalog facility alternate master catalog on an IBM 3380 Direct Access Storage volume, including volume IPL capability.

```
//INITIPL  JOB

//*************************************************************
//*  THIS JOB INITIALIZES AN IPL VOLUME (ALTVOL) WITH IPLTXT.  *
//*  THE VOLUME MUST BE OFFLINE.                               *
//*************************************************************

//INITIX   EXEC PGM=ICKDSF
//SYSPRINT DD SYSOUT=A
//IPLTXT   DD  DSN=SYS1.ASAMPLIB(IEAIPL00),DISP=SHR,UNIT=3380,
//             VOL=SER=SYSLIB
//SYSIN    DD *
  INIT    UNITADDRESS(CUU) NOVERIFY INDEX(0,2,5) VOLID(altvol) -
          PURGE IPLDD(IPLTXT)
/*

//*************************************************************
//* NOW VARY altvol ONLINE AND MOUNT.                         *
//*************************************************************

//*************************************************************
//* THE FOLLOWING JOB IS OPTIONAL AND                         *
//* MAY BE USED TO BRING SELECTED SYSTEM DATA SETS            *
//* ONTO THE NEW VOLUME (altvol).                             *
//*************************************************************

//ICFSMB  JOB
//ALLOC   EXEC PGM=IEFBR14
//SYSPRINT DD  SYSOUT=A
//DUMP00   DD  DSN=SYS1.DUMP00,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(40),,CONTIG)
//DUMP01   DD  DSN=SYS1.DUMP01,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(40),,CONTIG)
//NUCLEUS  DD  DSN=SYS1.NUCLEUS,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(32,,27),,CONTIG),
//             DCB=(RECFM=U,BLKSIZE=13030)
//LPALIB   DD  DSN=SYS1.LPALIB,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(40,4,391),,CONTIG),
//             DCB=(RECFM=U,BLKSIZE=13030)
//CMDLIB   DD  DSN=SYS1.CMDLIB,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(10,2,111),,CONTIG),
//             DCB=(RECFM=U,BLKSIZE=13030)
```

```
//IMAGELIB DD  DSN=SYS1.IMAGELIB,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(3,,27),,CONTIG),
//             DCB=(RECFM=U,BLKSIZE=1024)
//SVCLIB   DD  DSN=SYS1.SVCLIB,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(3,,27),,CONTIG),
//             DCB=(RECFM=U,BLKSIZE=13030)
//LINKLIB  DD  DSN=SYS1.LINKLIB,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(59,5,500),,CONTIG),
//             DCB=(RECFM=U,BLKSIZE=13030)
//MACLIB   DD  DSN=SYS1.MACLIB,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(80,8,200),,CONTIG),
//             DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80)
//MANX     DD  DSN=SYS1.MANX,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(10),,CONTIG),
//             DCB=(RECFM=VBS,BLKSIZE=4096,LRECL=4096)
//MANY     DD  DSN=SYS1.MANY,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(10),,CONTIG),
//             DCB=(RECFM=VBS,BLKSIZE=4096,LRECL=4096)

//*************************************************************
//*       For the following three data sets, MVS8 is          *
//*       used instead of SYS1, since SYS1.VTAMLIB,           *
//*       SYS1.PARMLIB, and SYS1.PROCLIB are enqueued         *
//*       by data set name and those data sets are in use.    *
//*************************************************************

//PARMLIB  DD  DSN=MVS8.PARMLIB,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(7,2,55),,CONTIG),
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//VTAMLIB  DD  DSN=MVS8.VTAMLIB,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(7,5,75),,CONTIG),
//             DCB=(RECFM=U,BLKSIZE=13030)
//PROCLIB  DD  DSN=MVS8.PROCLIB,UNIT=3380,VOL=SER=altvol,
//             DISP=(,KEEP),SPACE=(CYL,(7,,167),,CONTIG),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN    DD  DUMMY
/*
//*************************************************************
//*   COPY REQUIRED SYSTEM DATASETS TO altvol.                *
//*************************************************************

//*
//COPY    EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=A
//SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(20,5))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(20,5))
//NUCLEUS  DD  DSN=SYS1.NUCLEUS,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=U,BLKSIZE=32760),DISP=SHR
//NUC      DD  DSN=SYS1.NUCLEUS,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
//LPALIB   DD  DSN=SYS1.LPALIB,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=U,BLKSIZE=32760),DISP=SHR
//LPA      DD  DSN=SYS1.LPALIB,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
```

```
//PARMLIB  DD  DSN=MVS8.PARMLIB,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80),DISP=SHR
//PARM     DD  DSN=SYS1.PARMLIB,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
//CMDLIB   DD  DSN=SYS1.CMDLIB,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=U,BLKSIZE=32760),DISP=SHR
//CMD      DD  DSN=SYS1.CMDLIB,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
//IMAGELIB DD  DSN=SYS1.IMAGELIB,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=U,BLKSIZE=1024),DISP=SHR
//IMAGE    DD  DSN=SYS1.IMAGELIB,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
//SVCLIB   DD  DSN=SYS1.SVCLIB,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=U,BLKSIZE=32760),DISP=SHR
//SVC      DD  DSN=SYS1.SVCLIB,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
//PROCLIB  DD  DSN=MVS8.PROCLIB,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),DISP=SHR
//PROC     DD  DSN=SYS1.PROCLIB,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
//LINKLIB  DD  DSN=SYS1.LINKLIB,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=U,BLKSIZE=32760),DISP=SHR
//LINK     DD  DSN=SYS1.LINKLIB,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
//MACLIB   DD  DSN=SYS1.MACLIB,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=FB,BLKSIZE=3120,LRECL=80),DISP=SHR
//MAC      DD  DSN=SYS1.MACLIB,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
//VTAMLIB  DD  DSN=MVS8.VTAMLIB,UNIT=3380,VOL=SER=altvol,
//             DCB=(RECFM=U,BLKSIZE=32760),DISP=SHR
//VTAM     DD  DSN=SYS1.VTAMLIB,UNIT=3380,VOL=SER=sysres,
//             DISP=SHR
//SYSIN    DD  *
  C I=NUC,O=NUCLEUS
  C I=LPA,O=LPALIB
  C I=PARM,O=PARMLIB
  C I=CMD,O=CMDLIB
  C I=IMAGE,O=IMAGELIB
  C I=SVC,O=SVCLIB
  C I=PROC,O=PROCLIB
  C I=LINK,O=LINKLIB
  C I=MAC,O=MACLIB
  C I=VTAM,O=VTAMLIB
/*
//*************************************************************
//*            RENAME MVS8.PROCLIB TO SYS1.PROCLIB.           *
//*            RENAME MVS8.VTAMLIB TO SYS1.VTAMLIB.           *
//*            RENAME MVS8.PARMLIB TO SYS1.PARMLIB.           *
//*************************************************************
```

```
/*
//RENAME EXEC  PGM=IEHPROGM
//SYS2PROC DD  DISP=SHR,UNIT=3380,VOL=SER=altvol
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
  RENAME DSNAME=MVS8.PROCLIB,VOL=3380=altvol,             C
         NEWNAME=SYS1.PROCLIB
  RENAME DSNAME=MVS8.VTAMLIB,VOL=3380=altvol,             C
         NEWNAME=SYS1.VTAMLIB
  RENAME DSNAME=MVS8.PARMLIB,VOL=3380=altvol,             C
         NEWNAME=SYS1.PARMLIB
/*
//*************************************************************
//*    DEFINE AN ALTERNATE INTEGRATED CATALOG FACILITY CATALOG *
//*    ON VOLUME altvol.                                       *
//*    EXPORT WILL DISCONNECT THE NEW CATALOG                  *
//*    IF IT WAS PREVIOUSLY DEFINED.                           *
//*    STGINDEX AND PAGE DATA SETS ARE ALSO DEFINED.           *
//*************************************************************

//EXPDEF  EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD  SYSOUT=A
//CATLG    DD  UNIT=3380,VOL=SER=altvol,DISP=OLD
//SYSIN    DD  *
  EXPORT -
         CATALOG.Valtvol -
         DISCONNECT
  DEFINE MASTERCATALOG -
        (NAME(CATALOG.Valtvol) -
         FILE(CATLG) -
         VOLUMES(altvol) -
         CYLINDERS(18 1) -
         ICFCATALOG)
/*
//DEFPGE  EXEC PGM=IDCAMS COND=(0,NE)
//*************************************************************
//*   DEFINE SYS1.STGINDEX AND PAGE DATASETS.                 *
//*************************************************************

//STEPCAT  DD  DSNAME=CATALOG.Valtvol,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//DD1      DD  VOL=SER=altvol,UNIT=3380,DISP=OLD
//SYSIN    DD  *
  DEFINE CLUSTER -
             (NAME(SYS1.STGINDEX) -
              FILE(DD1) -
              KEYS(12 8) -
              CYLINDERS(6) -
              BUFFERSPACE(5120) -
              RECORDSIZE(2041 2041) -
              VOLUME(altvol) -
              REUSE) -
         DATA (CONTROLINTERVALSIZE(2048)) -
         INDEX (CONTROLINTERVALSIZE(1024))
```

```
  DEFINE PAGESPACE -
             (NAME(SYS1.PAGE01) -
              FILE(DD1) -
              CYLINDERS(30) -
              VOLUME(altvol) -
              UNIQUE)
  DEFINE PAGESPACE -
             (NAME(SYS1.PAGE02) -
              FILE(DD1) -
              CYLINDERS(40) -
              VOLUME(altvol) -
              UNIQUE)
  DEFINE PAGESPACE -
             (NAME(SYS1.PAGE03) -
              FILE(DD1) -
              CYLINDERS(100) -
              VOLUME(altvol) -
              UNIQUE)
/*
//*************************************************************
//*   CATALOG ALL NON-VSAM DATA SETS IN THE NEW CATALOG       *
//*   NOTE: VOLUME IDs BELOW ARE:                             *
//*         (******) altvol - ALT. CATALOG VOLUME             *
//*         (sysres) MASTER CATALOG VOLUME                    *
//*         (sysrs2) ANOTHER VOLUME WITH SOME SYSTEM DATA SETS *
//*************************************************************

//*
//DNVSAM  EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
  DEFINE NONVSAM -
        (NAME(SYS1.LINKLIB) VOL(******) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(user.JOBS) VOL(usrvol) DEVT(3380))  -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(user1.LINKLIB) VOL(sysrs2) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.LPALIB) VOL(******) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.MLPALIB) VOL(sysrs2) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.NUCLEUS) VOL(******) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.PARMLIB) VOL(******) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.PROCLIB) VOL(******) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
```

```
DEFINE NONVSAM -
      (NAME(SYS1.DUMP00) VOL(******) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.DUMP01) VOL(******) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(IPO1.PROCLIB) VOL(sysres) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.NCPDUMP) VOL(sysres) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.RACF)    VOL(sysrs2) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(USER.PROCLIB) VOL(sysrs2) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.VTAMLIB) VOL(******) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.VTAMLST) VOL(sysrs2) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.VTAMOBJ) VOL(sysrs2) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.BRODCAST) VOL(sysrs2) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.CMDLIB) VOL(******) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS2.CMDLIB) VOL(sysrs2) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.DCMLIB) VOL(sysres) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.HELP) VOL(sysres) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.LOGREC) VOL(******) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.IMAGELIB) VOL(******) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
DEFINE NONVSAM -
      (NAME(SYS1.MACLIB) VOL(******) DEVT(3380)) -
       CAT(CATALOG.Valtvol)
```

```
  DEFINE NONVSAM -
        (NAME(SYS1.SAMPLIB) VOL(sysres) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.SVCLIB) VOL(******) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.UADS) VOL(sysrs2) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.TELCMLIB) VOL(sysrs2) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(PRIME.PROCLIB) VOL(sysrs2) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.MANX) VOL(******) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
  DEFINE NONVSAM -
        (NAME(SYS1.MANY) VOL(******) DEVT(3380)) -
         CAT(CATALOG.Valtvol)
/*
//*************************************************************
//*  DEFINE OS CVOL ALIASES IN THE NEW CATALOG.               *
//*  EACH OS CVOL NEEDS A SEPARATE STEP WITH A SEPARATE STEPCAT*
//*************************************************************
//*
//*
//DEFCAT  EXEC PGM=IDCAMS COND=(0,NE)
//STEPCAT  DD  DSN=CATALOG.Valtvol,DISP=OLD
//SYSPRINT DD  SYSOUT=A
//SYSCTLG  DD  DSN=SYSCTLG,DISP=SHR,VOL=SER=sysrs2,
//             UNIT=3380,SPACE=(CYL,(8),,CONTIG)
//SYSIN    DD  *
  DEFINE NONVSAM -
        (NAME(SYSCTLG.Vsysrs2) -
         DEVICETYPE(3380) -
         VOLUMES(sysrs2))
  DEFINE ALIAS -
        (NAME(usernm) -
         RELATE(SYSCTLG.Vsysrs2))
/*
//DEFAIDS EXEC PGM=IDCAMS COND=(0,NE)
//STEPCAT  DD  DSN=CATALOG.Valtvol,DISP=OLD
//SYSPRINT DD  SYSOUT=A
//SYSCTLG  DD  DSN=SYSCTLG,DISP=SHR,VOL=SER=usrvol,
//             UNIT=3380,SPACE=(CYL,(8),,CONTIG)
//SYSIN    DD  *
  DEFINE NONVSAM -
        (NAME(SYSCTLG.Vusrvol) -
         DEVICETYPE(3380) -
         VOLUMES(usrvol))
  DEFINE ALIAS -
        (NAME(usernm) -
         RELATE(SYSCTLG.Vusrvol))
  DEFINE ALIAS -
        (NAME(AIDS) -
         RELATE(SYSCTLG.Vusrvol))
/*
```

```
//*************************************************************
//*     INITIALIZE LOGREC ON altvol.                          *
//*************************************************************
//LOGREC EXEC PGM=IFCDIP00
//SERERDS DD   DSN=SYS1.LOGREC,DISP=(NEW,KEEP),
//             VOLUME=(,RETAIN,SER=altvol),
//             SPACE=(TRK,(71),,CONTIG),UNIT=3380
/*
//*************************************************************
//*    THE FOLLOWING JOBS UPDATE SYS1.NUCLEUS WITH            *
//*    A SYSCATxx MEMBER FOR THE ALTERNATE CATALOG.           *
//*************************************************************
/*
//*************************************************************
//*    SCRATCH OLD SYSCATXX (IF PRESENT) FROM SYS1.NUCLEUS    *
//*    AND REPLACE IT WITH THE NEW MEMBER.                    *
//*************************************************************
                                                          72
//*
//SCRTCHI8 EXEC PGM=IEHPROGM
//DD2      DD  DISP=OLD,VOL=SER=altvol,UNIT=3380
//SYSPRINT DD  SYSOUT=A
//SYSIN    DD  *
  SCRATCH DSNAME=SYS1.NUCLEUS,VOL=3380=altvol,             C
  MEMBER=SYSCATI8
//*
//*************************************************************
//*    CHANGE COL 7 IN GENER CONTROL CARD                     *
//*    ACCORDING TO CATALOG TYPE:                             *
//*      INTEGRATED CATALOG FACILITY  = C'1' (X'F1')          *
//*      VSAM  = blank (X'40')                                *
//*    COL 8 is not used.  (it will be ignored if used.)      *
//*    CHANGE COLs 9 and 10 IN GENER CONTROL CARD             *
//*    ACCORDING TO VALUE OF CAS SERVICE TASK LOWER LIMIT:    *
//*      C'00' to C'C8' (defaults to C'18')                   *
//*************************************************************
/*
//GENERI8 EXEC PGM=IEBGENER
//SYSUT2   DD  DSN=SYS1.NUCLEUS,
//         DISP=(MOD,KEEP),
//         VOL=SER=altvol,UNIT=3380
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  *
altvol1 0CCATALOG.Valtvol
//*  col 7 is C'1' for integrated catalog facility catalog, col 8 is
//*  not used, cols 9 and 10 are C'18' for CAS service task lower
//*  limit value
/*
//SYSIN    DD  *
  GENERATE MAXNAME=1
  MEMBER NAME=SYSCATI8
/*
```

```
//*************************************************************
//*     UPDATES PARMLIB(IEAAPF00) TO SHOW                     *
//*     VTAMLIB IS ON altvol.                                 *
//*************************************************************

//UPDTAPF EXEC  PGM=IEBUPDTE,PARM=MOD
//SYSPRINT DD  SYSOUT=A
//SYSUT1 DD  DSN=SYS1.PARMLIB,UNIT=3380,VOL=SER=altvol,DISP=SHR
//SYSUT2 DD  DSN=SYS1.PARMLIB,UNIT=3380,VOL=SER=altvol,DISP=SHR
//SYSIN    DD  DATA
./ REPL LIST=ALL,NAME=IEAAPF00
 user.NCPLINK sysres,
 IPO1.LINKLIB sysres,
 user.RESLIB usrvol,
 user.PGMLIB usrvol,
 SYS1.JES3LIB sysrs2,
 SYS1.VTAMLIB altvol
./ ENDUP
/*
//*************************************************************
//*   LISTS THE VTOC ON THE ALTERNATE CATALOG VOLUME.         *
//*************************************************************

//LVTOC   EXEC PGM=IEHLIST
//SYSPRINT DD  SYSOUT=A
//DD1      DD  UNIT=3380,DISP=SHR,VOL=SER=altvol
//SYSIN    DD  *
  LISTVTOC DUMP,VOL=3380=altvol
  LISTVTOC FORMAT,VOL=3380=altvol
/*

//*************************************************************
//*    LISTS THE CATALOG ON THE ALTERNATE CATALOG VOLUME.     *
//*************************************************************

//LISTCAT EXEC  PGM=IDCAMS
//STEPCAT DD  DSN=CATALOG.Valtvol,DISP=SHR
//SYSPRINT  DD  SYSOUT=A
//SYSIN  DD  *
  LISTCAT -
       CATALOG(CATALOG.Valtvol) -
       ALL
/*
//*************************************************************
//*  FOLLOWING SUCCESSFUL COMPLETION OF THIS JOBSTREAM,       *
//*          IPL WITH CLPA AND REPLY TO CATALOG PROMPT        *
//*                  R 00,I8                                  *
//*************************************************************
```

Appendix D. Operand Notation for SHOWCAT

The SHOWCAT Macro

The SHOWCAT (show, or display, the catalog) macro has three forms: standard, list, and execute. Although the VSAM catalog and integrated catalog facility catalog have different structures, the SHOWCAT macro supports both VSAM and integrated catalog facility catalogs. Thus, all references in this appendix to VSAM catalogs also apply to integrated catalog facility catalogs.

You can use the IGGSHWPL macro to generate a DSECT statement and labels for the fields in the parameter list for SHOWCAT.

Using the SHOWCAT Macro

The SHOWCAT (show, or display, the catalog) macro enables you to retrieve information from a catalog independently of an open data set 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; one entry points to one or more other entries, which point to yet others. Figure 31 on page 168 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 in Figure 31 on page 168. You first issue SHOWCAT on the name of an object.

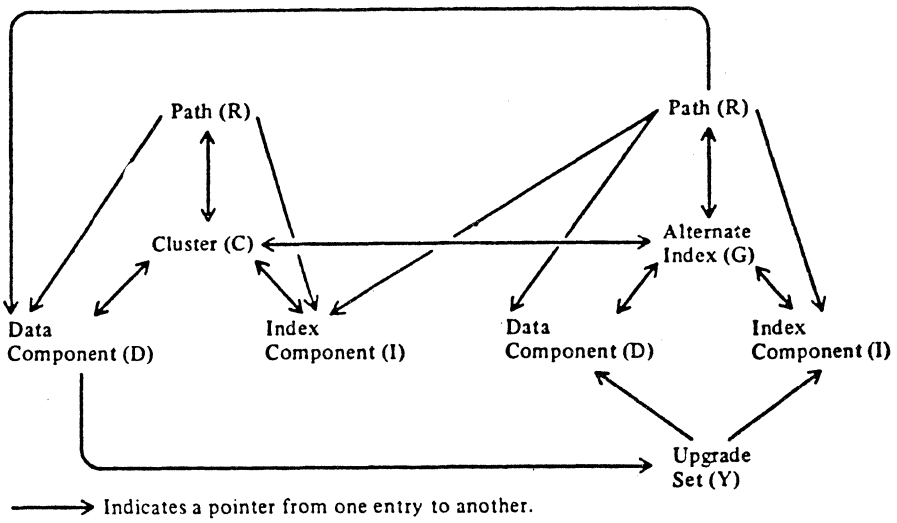


Figure 31. Interrelationship Among Catalog Entries.

The information VSAM returns to you includes the control-interval numbers of catalog records in entries that describe associated objects. You then issue SHOWCAT on a control-interval number to retrieve information from one of these other entries.

The first time you issue SHOWCAT, VSAM searches VSAM catalogs in the following order to locate the entry that describes the object you name:

1. The STEPCAT or JOBCAT user catalog or catalogs (catalogs can be concatenated under STEPCAT or JOBCAT).
2. The master catalog.
3. When the object has a qualified name, the catalog, if any, whose name or alias is the same as the first-level qualifier of the object's name.

VSAM returns to you the address of the access-method control block that defines the catalog. In subsequent use of SHOWCAT, you can specify that address, which then causes VSAM to search only that catalog.

Standard Form of SHOWCAT

The format of the list form of SHOWCAT is:

SHOWCAT	[**ACB** = *address*] [**AREA** = *address*] [{**CI** = *address*|**NAME** = *address*}]

ACB = *address*
specifies the address of the access-method control block that defines the catalog that contains the entry from which information is to be displayed. You issue the first SHOWCAT without ACB specified and VSAM supplies it to you for the next SHOWCAT (see the description of the work area under the AREA operand). Specifying ACB enables VSAM to go directly to the correct catalog without searching other catalogs first. You should always specify ACB when you specify CI instead of NAME.

AREA = *address*
specifies the address of the work area in which the catalog information is to be displayed. The first 2 bytes of the area must give the length of the area, including the 2 bytes; the minimum is 64. If the area is too small, VSAM returns as much information as possible.

You can use the IGGSHWPL macro to generate a DSECT statement and labels for the fields in the work area.

The format of the work area is:

Offset	Length	Symbolic Name	Description
0(0)	2	SHWLEN1	Length of the area, including the length of this field (provided by you).
2(2)	2	SHWLEN2	Length of the area actually used by VSAM, including the length of this field and the preceding field.
4(4)	2	SHWACBP	The address of the ACB that defines the catalog that contains the entry from which information is displayed.
8(8)	1	SHWTYPE	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 contain one set of information for C, G, R, and Y types and another set for D and I types: The format of the work area for **C, G, R,** and **Y types** is:

Offset	Length	Symbolic Name	Description
(9)	1	1SHWATTR	For C and Y types: reserved.
			For G type:
		SHWUP	**x**... .... The alternate index may (1) or may not (0) be a member of an upgrade set. One way of verifying this 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 31 on page 168 shows 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).
fit = .			**.xxx xxxx** Reserved.
			For R type:
		SHWUP	**x**... .... The path is (1) or isn't (0) defined for upgrading alternate indexes.
			.xxx xxxx Reserved.
10(A)	2	SHWASS0	The number of association pointers that follow.
		SHWACT	Each association pointer 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, G, I. With R: C, D, G, I. With Y: D, I. Figure 31 on page 168 shows how the catalog entries for all these objects are interrelated.
12(C)	1	SHWATYPE	Type of object the entry describes.
13(D)	3	SHWAC1	The control-interval number of its first record.

Offset	Length	Symbolic Name	Description
16(10)			Next association pointer, and so on. For type Y, if the area is too small to display an association pointer for each associated object, VSAM displays as many pointers as possible and returns a code of 4 in register 15. For types C and G, if the area is too small, VSAM displays as many pointers as possible, but returns as a code of 0 in register 15 because fields for the main associated objects can always be displayed (in the smallest allowed work area). For type R, fields for all associated objects (five possible) can always be displayed. (An associated pointer occupies 4 bytes (1 byte for the associated entry type and 3 bytes for its control-interval number). However, for all types except Y, 4 additional bytes are required as work space for the SHOWCAT processor. Thus, for example, if you provide 80 bytes for associated objects, as many as 10 association pointers can be displayed for type C or G and 20 for type Y.)

The format of the work area for **D** and **I** types is:

Offset	Length	Symbolic Name	Description
9(9)	1		Reserved.
10(A)	2	SHWDSB	Relative position of the prime key in records in the data component.
		SHWRKP	For the data component of an entry-sequenced data set, there is no prime key and this field is 0.
12(c)	2	SHWKEYLN	Length of the prime key.
14(E)	4	SHWCISZ	Control-interval size of the data or index component.
18(12)	4	SHWMREC	Maximum record size of the data or index component.
22(16)	2	SHWASS	The number of association pointers that follow.
		SHWACT	Each association pointer 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 31 on page 168 shows how the catalog entries for all these objects are interrelated.
24(18)	1	SHWATYPE	Type of object the entry describes.
25(19)	3	SHWACI	The control-interval number of its first record.
28(1C)			Next association pointer, and so on. Fields for all associated objects can always be displayed.

{CI = *address*|**NAME** = *address* **}**
specifies the address of an area that identifies the catalog entry that contains the desired information.

CI = *address*
specifies that the area is 3 bytes long and contains the control-interval number (RBA divided by 512) of the first record in the catalog entry. You can issue the first SHOWCAT with NAME specified, and then VSAM supplies control-

interval numbers to you for other SHOWCATs (see the description of the work area under the AREA operand). The type of object named must be C, D, G, I, R, or Y. The 3-byte area must be separate from the work area, even though VSAM returns a control-interval number in the work area.

NAME = *address*
specifies that the area is 44 bytes long and contains the name of the object described by the entry. The name is left-justified and padded with blanks. The type of object named must be C, D, G, I, or R.

List Form of SHOWCAT

The format of the list form of SHOWCAT is:

SHOWCAT	[**ACB** = *address*] [**AREA** = *address*] [{**CI** = *address*|**NAME** = *address*}] **MF** = **L**

MF = **L**
indicates that this is the list form of SHOWCAT.

AREA and {CI|NAME} are optional in the list form of SHOWCAT, but, if they are not so specified, they must be specified in the execute form.

Note: For a detailed description of ACB, AREA, and CI|NAME parameters, refer to the information contained in "Standard Form of SHOWCAT" on page 168.

Execute Form of SHOWCAT

The format of the execute form of SHOWCAT is:

SHOWCAT	[**ACB** = *address*] [**AREA** = *address*] [{**CI** = *address*|**NAME** = *address*}] **MF** = ({**E**|**B**},*address*)

MF = ({**E**|**B**},*address*)
indicates that this is the execute form of SHOWCAT.

E
indicates that the parameter list, whose address is given in address, 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, address) 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, and a register from 2 through 12. Register 1 is used to pass the parameter list to VSAM (MF = E).

Note: For a detailed description of ACB, AREA, and CI|NAME parameters, refer to the information contained in "Standard Form of SHOWCAT" on page 168.

Expressions That Can Be Used for SHOWCAT

The values for an operand of SHOWCAT can be expressed in a variety of ways, as:

- An absolute numeric expression, for example, BUFFERS = (2048(10)).
- A code or a list of codes separated by commas and enclosed in parentheses; for example, FIX = BFR or FIX = (BFR,IOB).
- A register (in parentheses) from 2 through 12 that contains an address or numeric value; for example, BFRNO = (3). In the execute form of a macro, you can use register 1 for the address of the parameter list. Equated labels can be used to designate a register; for example, BFRNO = (BFR#), where the equate statement, BFR# EQU 3, has been included in the program.
- An expression valid for a relocatable A-type address constant; for example, AREA = RETURN + 4.

The expressions that can be used depend on the operand. Only absolute numeric expressions, codes, and relocatable A-type address constants are valid for the list form of a macro.

Figure 32 indicates the expressions allowed for each operand of SHOWCAT:

Operands	Absolute Numeric	Code	Register	A-Type Address
SHOWCAT (STANDARD)				
ACB			X	X
AREA			X	X
CI			X	X
NAME			X	X
SHOWCAT (LIST)				
ACB				X
AREA				X
CI				X
MF		X		
NAME				X
SHOWCAT (EXECUTE)				
ACB			X	
AREA			X	
CI			X	
MF				
B		X		
E		X	X	
address			X	
NAME				

Figure 32. Operand Expressions for the SHOWCAT Macro

Return Codes from SHOWCAT

VSAM returns a code in register 15 that indicates whether the SHOWCAT request was successful:

Code	Meaning
0(0)	VSAM completed the task.
4(4)	The area specified in the AREA operand is too small to display all pairs of fields for the associated objects.
8(8)	There is insufficient virtual storage to complete the task. (A GETMAIN failed.)
12(C)	Either the ACB address is invalid, or the VSAM master catalog does not exist, or it is not open.
16(10)	The address specified in the AREA operand is outside the partition or address space of the program that issued SHOWCAT.
20(14)	The named object or control interval does not exist.
24(18)	There was an I/O error in gaining access to the catalog.
28(1C)	The control-interval number is invalid.
32(20)	The catalog record does not describe a C, D, G, I, R, or Y type of object.
36(24)	The interrelationship among catalog entries is in error. For example, another type.
40(28)	There was an unexpected error code returned from catalog management to the SHOWCAT processor.

Appendix E. VSAM Catalogs

This appendix applies only to the VSAM catalog.

A VSAM catalog is structured as a VSAM key-sequenced cluster with two key ranges. Like a VSAM key-sequenced cluster, the catalog consists of an index component and a data component. The space the catalog occupies is divided into fixed-length control intervals of 512 bytes each.

The Data Component

The catalog's data component is divided into two areas: a high-key range and a low-key range. One high-key range record and one low-key range record (plus, possibly, one or more extension records in the low-key range) exist for each cataloged object:

- A high-key range record contains the full 44-byte name of the cataloged object and the 3-byte control interval number of the object's low-key range record. Each high-key range record is 47 bytes long; up to 10 records can exist in a control interval in the high-key range.
- A low-key range record contains the information necessary to describe and locate the cataloged object. Each low-key range record is 505 bytes long; only one record exists in a control interval in the low-key range. Low-key range records are described in detail in *Catalog Diagnosis Reference*.

The high-key range and low-key range also contain records that describe the catalog itself.

Control intervals that contain data records are grouped into control areas. A control area consists of control intervals that contain high-key range records or low-key range records; high-key range records do not exist in the same control area with low-key range records.

The Index Component

The catalog's index component consists of 505-byte index records, one per control interval. Control intervals containing index records are not grouped into control areas.

Each index record consists of one or more variable-length index entries. Each entry contains a compressed key value and a pointer to a lower level in the catalog (that is, a control interval in either a lower index level or in one of the data key ranges). A compressed key is the 44-byte cataloged object's entry name, minus characters from the front and back of the entry name that are not needed to distinguish the key value (entry name) from the preceding and following key values. (See *VSAM Administration Guide* for details on indexes, index processing, and key compression.)

The index has two parts—an index set and a sequence set:

- The sequence set is the lowest level of the index. A sequence set record exists for each control area. The sequence set record has an entry for each control

interval in the control area. Each entry points to a control interval and the control interval's highest key value (compressed entry name).

- The index set is all levels of the index higher than the sequence set level. An index set record points to lower-level index set records or to sequence set records. Each entry points to the lower-level record and the highest key value (compressed entry name) of the keys in the lower-level record.

Size of a Control Area and Location of the Sequence Set

The size of the control area is limited by how many entries can fit in a sequence set record. The length of an entry varies (because of the variable-length compressed entry name), but approximately 40 entries can fit in a sequence set record. A control area exists as a whole number of contiguous tracks on a direct access device. A catalog's control interval does not span tracks. Therefore, the size of a control area depends on the number of entries that will fit into a sequence set record, and on the type of direct access device:

- For an IBM 3330, 3330 Model 11, or 2305 Model 2, a control area is 2 tracks and contains 40 data control intervals (20 control intervals per track).
- For an IBM 3350, a control area is 2 tracks and contains 54 data control intervals (27 control intervals per track).
- For an IBM 3340/3344, a control area is 4 tracks and contains 48 data control intervals (12 control intervals per track).
- For an IBM 3380, a control area is 1 track and contains 46 data control intervals.

The track preceding each control area contains the control area's sequence set record. The sequence set record's track is also divided into control intervals (the number of control intervals per track depends on the device type). The sequence set record is copied in each control interval on the track to improve performance, reducing the amount of rotational delay during access.

A VSAM Catalog's Use in Data and Space Management

VSAM catalogs are a central information point for all VSAM data sets and the direct access storage volumes containing them. The information describing a volume and the data sets on it allows VSAM to allocate and deallocate data sets on the volumes without the volumes being mounted on a device of the system. The catalogs also provide VSAM with information to authorize access to data sets, compile usage statistics, and relate RBAs to physical locations. Defining a VSAM data set automatically builds the appropriate catalog entry containing all the necessary information.

All VSAM data sets on a volume must be cataloged in the same VSAM catalog, and that catalog must be the one that owns the volume. This may be either the master catalog or a user catalog. A VSAM data set has an entry in only one catalog.

Information Contained in the Records of a Catalog

VSAM catalog records describe direct access volumes in terms of the allocation of data spaces, the location of available space, and data set records. VSAM can allocate and deallocate space on cataloged volumes that are not mounted. However, when allocating space to a data set, if there is not sufficient space available in the data space or data spaces on a volume, you must use the access method services DEFINE space command to get the additional space the data set needs.

Information in a Data Set Record

Data set records provide information for the connection between a data record's RBA and a storage volume's physical attributes. Besides the type of storage device and a list of volume serial numbers, a VSAM catalog keeps other data set information, including:

- A pointer to the location of each extent of the data set
- Statistics on the results of operations performed on the data set and its records, such as the number of insertions and deletions and the amount of free space remaining
- Attributes of the data set determined when it was defined, such as control interval size, physical record size, number of control intervals in a control area, and, for a key-sequenced data set, location of the key field
- Password protection information
- An indication of the connection between: the index and the data components of a key-sequenced data set; the index and data components of an alternate index cluster; the alternate index and the base cluster of a path; and an alternate index upgrade set and its base cluster
- Information used to determine whether a key-sequenced data or index component has been processed without the other
- Information about the volume(s) on which the data set is stored

Information in a Volume Record

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

- The volume serial number and device characteristics
- The location of data spaces on a volume
- The location and size of free areas available for allocation to data sets

From this information, you can derive:

- The count of data spaces and data sets on a volume
- The location of data sets within data spaces on a volume
- An indication of the data spaces associated with a data set

Allocation of Catalog Space

When a catalog is created, it is built in the first data space on the volume. You can specify an exact amount of space for it or allow access method services to determine the space. When access method services determines the total amount of catalog space, you must specify the amount of space necessary for the catalog entries and access method services determines the space for the rest of the catalog.

Utilization of Catalog Space

Because the index set, sequence set, and low-key range control intervals each contain only one catalog record, the space allocated to these parts of the catalog can be completely filled before a secondary allocation of space is required. However, the catalog's high-key range control intervals can contain up to 10 records each, and are subject to control interval splitting and control area splitting:

- Control interval split: When VSAM manipulates catalog records, it tries to add a record to a high-key range control interval that contains 10 records and the control interval is split into two control intervals. Five of the control interval's records are moved into an unused control interval, so the control interval now contains five high-key range records. The new record is then added to the appropriate control interval. Each control interval now has sufficient free space to accept more high-key range records, but the free space may not be used until entries with the appropriate key values are added to the catalog.
- Control area split: When all control intervals in one of the high-key range's control areas are used and a control interval must be split, the control area is split. Before the control area is split, all its control intervals except one may be only half full. When the control area is split, half the control intervals are moved into an unused control area. The appropriate control interval is split and the high-key range record is added in key sequence to the appropriate control interval.

Whenever a control interval or control area is split, the appropriate sequence set and index set records are updated or split to reflect the new data structure.

VSAM Catalogs and Volume Ownership

VSAM accesses and manages space on a volume in units called data spaces, which may be:

- One or more data sets stored in a data space
- One data set stored in one or more data spaces on one or more storage volumes
- Data sets and data spaces extended beyond their original size
- Data spaces extended automatically by whole numbers of tracks or cylinders
- Data spaces as large as a volume with a maximum of 16 extents

VSAM data spaces are defined in a VSAM catalog. That catalog controls (owns) the volumes that contain the data spaces defined in the catalog. The catalog also owns candidate volumes for VSAM data sets defined in the catalog. VSAM ownership of a volume is established with the DEFINE USERCATALOG or DEFINE SPACE commands or when the UNIQUE attribute is specified with DEFINE CLUSTER, DEFINE ALTERNATEINDEX, or DEFINE PAGESPACE.

All VSAM clusters, alternate indexes, page spaces, and data spaces stored on a volume must be cataloged in the catalog that owns the volume. Only one VSAM catalog can own the volume.

VSAM volume ownership does not affect non-VSAM data sets that reside on the volume. Non-VSAM data sets can exist on a volume owned by a VSAM catalog, and can (*but should not*) be cataloged in a catalog that does not own the volume. Because OS CVOLs are considered non-VSAM data sets, an OS CVOL can also exist on a volume owned by a VSAM catalog.

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

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

  `Z999999n.VSAMDSPC.Taaaaaaa.Tbbbbbbb`

  where:

  n = 2 if no catalog resides in the data space

  n = 4 if a user catalog resides in the data space, or if the master catalog resides in the data space and the master catalog was created under OS/VS2 MVS Release 2 or a later release

  n = 6 if the master catalog resides in the data space

  aaaaaaabbbbbbb = the timestamp value

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

  `VSAMDSET.Tbbbbbbb.DFDyyddd.Taaaaaaa.Tbbbbbbb`

  where:

  yyddd = the date (year and Julian day)

  aaaaaaabbbbbbb = the timestamp value

If you do not specify a name for the data and index components of a suballocated VSAM data set, VSAM generates a name for these components.

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

Each volume owned by a VSAM catalog contains two timestamps that are written in the Format-4 DSCB when the volume is first cataloged. Both timestamps are updated but only the second one is checked. The first timestamp is maintained for compatibility with earlier VSAM releases. The volume timestamps are updated as follows:

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

Using the DEFINE Command for VSAM Catalogs

VSAM uses catalogs as a central information point for all VSAM data sets and the direct access volumes on which they are stored. Users may:

- Define a VSAM object in a VSAM catalog by using the access method services DEFINE command.
- Use the DEFINE command to define non-VSAM objects in a VSAM catalog.
- Use JCL, such as DISP=(,CATLG), to define non-VSAM objects into a VSAM catalog.

When you issue the DEFINE command to catalog an object, access method services builds one or more catalog entries to describe the object.

The VSAM objects you can define are:

- Master catalog (this command creates a user catalog)
- User catalog
- Data space
- Cluster, or VSAM data set
  - ESDS (entry-sequenced data set)
  - KSDS (key-sequenced data set)
  - RRDS (relative record data set)
- Alternative index
- Path
- Non-VSAM data set
- Alias
- Generation data group (GDG)
- Page space

Many parameters may be specified for the various DEFINE commands. The most important ones are:

- FILE
- FREESPACE
- IMBED/REPLICATE
- KEYRANGES
- NAME
- RECORDSIZE
- SHAREOPTIONS
- SPEED/RECOVERY
- SUBALLOCATABLE/UNIQUE

Creating a VSAM Catalog

A master catalog, which is the system catalog, is established at system generation time and must be permanently mounted. This master catalog is the only one allowed on the system and you cannot issue DEFINE MASTERCATALOG to create a master catalog. If you issue DEFINE MASTERCATALOG, VSAM creates a user catalog. You can, however, establish a user catalog as a master catalog at IPL time. See "Changing a User Catalog to a VSAM Master Catalog" on page 228 for a description of this procedure.

In the Mass Storage System, a user catalog can be defined on a Mass Storage System volume. If it is, it is retained until closed on the direct access storage staging drive to which it is staged. A catalog can contain entries for both objects stored on MSS volumes and objects stored on direct access storage volumes.

The master catalog points to user catalogs. The master catalog contains a user-catalog entry that identifies the user catalog's volume. Only one user catalog can reside on a volume. All VSAM objects on a volume must be defined in the volume's user catalog. A user catalog can control (catalog the objects of) more than one volume.

Specify passwords when defining a user catalog, even when the master catalog is not password protected, to prevent an unauthorized user from deleting your user catalog. The catalog's master password is required to delete it, but any password specified for the catalog becomes a master password. (See "How to Code Subparameters" in *Access Method Services Reference*.)

A user catalog can be deleted and redefined in the same step, but it cannot be used in that step. A TSO session is considered one step, so you would have to log off and log on to use a catalog that you had deleted and redefined.

Note: When a new user catalog has been defined, the first object cataloged causes the initialization of the entire catalog to take place. It also reserves the volume on which the catalog resides.

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

DEFINE USERCATALOG creates a suballocatable data space on a volume not owned by any other VSAM catalog. If the catalog entries are changed often by deleting or adding VSAM objects, it might be advisable to reserve its own data space for the catalog to prevent secondary allocation being built to form the catalog's prime allocation.

To define a user catalog that occupies its own data space and is recoverable, the following might be used:

```
//PRIMER    JOB WTSC,IBM,MSGLEVEL=1
//PRIM0022  EXEC PGM=IDCAMS
//VOL1      DD VOL=SER=WTVSAM,DISP-OLD,UNIT=DISK
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *
          DEF UCAT  (                    -
                    NAME (PRIMER.UCAT1)  -
                    VOL (WTVSAM)         -
                    CYL (3 1)            -
                    MASTERPW (UCATMRPW)  -
                    UPDATEPW (UCATUPDT)  -
                    RECOVERABLE          -
                    )
/*
```

The number of tracks required can be calculated using the worksheet in Figure 33 on page 186.

Defining a VSAM Recoverable Catalog

If you define a VSAM catalog with the RECOVERABLE option, the space assigned to the catalog is calculated differently. For a recoverable catalog, VSAM creates a catalog recovery area (CRA) on each volume owned by the catalog. (See "Space Allocation for the Catalog Recovery Area (CRA)" on page 188.)

The catalog recovery area is an entry-sequenced data set that exists on each volume owned by a recoverable catalog, including the volume on which the catalog itself resides. The CRA contains copies of the catalog's records, and can be used to recover a damaged catalog. Refer to "RECOVERABLE" in *Access Method Services Reference*.

Allocating the Volume

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

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

```
//ddname DD DISP=OLD,
//          VOLUME=SER=(volser1ffl,volser2,...")
```

VSAM Objects in a Data Space

All VSAM objects (VSAM catalogs, clusters, alternate indexes, and page spaces) are stored in VSAM data spaces. A unique component is defined as the only component in its data space. VSAM allocates the data space for a unique component when you define the data set. A nonunique object can share a data space with other VSAM objects such as VSAM catalogs, clusters, page spaces, and alternate indexes. A data space used to suballocate space for a data set must be defined before defining a data set.

A data space can contain many nonunique data sets, and can be stored in more than one data space on the same volume or on different volumes.

Space Allocating Space VSAM Objects

VSAM allocates space for a new data set or extends the space for an existing data set. For a new data set, the DEFINE fails if space is not available.

For an existing data set, if not enough space is available, VSAM tries to extend the data space. If the data space cannot be extended (already has the maximum number of extents or not enough space on the volume), VSAM tries to set up a new data space on the same volume.

If the data space cannot be extended because the secondary allocation amount of a suballocated data set is greater than the secondary allocation amount of the data space, VSAM builds a new data space. The primary and secondary allocation amounts are the same as the secondary allocation amount for the data set. The total amount of space allocated to VSAM data spaces on a volume might exceed the predicted limit. If there is not enough space on the volume and you identified another volume for expansion when defining the data set, VSAM tries to set up a data space on that volume.

How Space Is Assigned to a VSAM Catalog

When a catalog is defined, VSAM builds a data space on the volume and then builds the catalog.

To share the data space with other VSAM data sets, you can specify the space parameter for the data and, optionally, the index components of the catalog. The catalog occupies the space specified as a subparameter of DATA and INDEX. The amount of space is specified as a subparameter of USERCATALOG.

The following is a summary of the three ways space can be specified when defining a catalog.

- If DATA space subparameters are not coded, the entire data space is allocated to the catalog in multiples of allocation units. Any remaining space is available for suballocation by other VSAM data sets. INDEX space subparameters are invalid unless DATA space subparameters are specified.
- If both DATA and INDEX space subparameters are coded, the sum of the space subparameters determines the amount of space available to contain the catalog. VSAM determines the proportion of this space to be allocated to the catalog's data and index components. The remainder of the data space is available for suballocated VSAM data sets.
- If DATA space subparameters are coded but INDEX space subparameters are not coded, the DATA space subparameters determine the size of the low-key range of the catalog's data component. The remainder of the data space is available for suballocated VSAM data sets. VSAM determines the amount of space required for the high-key range of the data component, the sequence set, and the high-level index and adds this amount to the DATA space subparameter specification.

CYLINDERS, RECORDS, and TRACKS parameters are subparameters of USERCATALOG used to specify the size of a data space. For example:

```
DEFINE USERCATALOG -
TRACKS(prim secn)...)
```

- TRACKS specifies the number of tracks to be allocated to the data space.

You may, optionally, specify the size of the catalog itself by using the CYLINDERS, RECORDS, or TRACKS parameter as a subparameter of DATA, or DATA and INDEX. For example,

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

- TRACKS specifies the number of tracks to be allocated to the data space.
- RECORDS specifies the number of records in the low-key range of the catalog's data component (see "Estimating the VSAM Catalog's Space Requirements" on page 185 for a discussion of this value).

For example, if you specify as a subparameter of USERCATALOG, 100 cylinders of a 3380 (the primary allocation for the catalog's data space is 100 cylinders) and for the catalog's data and index components 5 cylinders (the sum of the DATA and INDEX primary allocation amounts is 5 cylinders), 95 cylinders remain in the data space and are available for suballocated VSAM data sets.

Catalog space is allocated in tracks even when you specify cylinders or records. The number of tracks is a multiple of an allocation unit. An allocation unit is the size of a control area. In each allocation unit, one track contains the control area's replicated sequence set record. For example, the size of an allocation unit is:

- 3 tracks for an IBM 3330, 3330 model 11, 3350, or 2305 model 2 (2 tracks for the control area and 1 track for the sequence set record).
- 5 tracks for an IBM 3340/3344 (4 tracks for the control area plus 1 track for the sequence set record).
- 2 tracks for an IBM 3380 (1 track for the control area and 1 track for the sequence set).

When the amount of space you specify is not a multiple of the device's allocation unit, the amount of space is rounded downward and the additional tracks are not used. If you specify less than the minimum allocation unit, the amount of space is rounded upward and a minimum-sized catalog is allocated.

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

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

- The number of tracks is rounded down to 18 tracks, which is six allocation units of 3 tracks each (the 19th track is not used and not available unless the catalog's data space is suballocated).
- One allocation unit (3 tracks) is allocated to the catalog's index set.
- Because 10% of the remaining 15 tracks is less than 3 tracks, one allocation unit is allocated to the high-key range of the data component. The first track contains a replicated sequence set record. The next 2 tracks contain a control area with 40 control intervals, which can hold a maximum of 400 records. (Records in the high-key range are 47 bytes long.)

- The remaining 4 allocation units (12 tracks) are allocated to the low-key range of the data component. Each allocation unit includes 1 track for the replicated sequence set record and two tracks for a control area with 40 control intervals, which can hold 40 records. (Records in the low-key range are 505 bytes long.) The low-key range can hold a maximum of 160 records. From 12 to 15 of these records (13 for a 3330) are used to describe the catalog itself.

When the maximum number of records is reached in the low-key or high-key range, the catalog must be extended to hold additional records. When you define the catalog, you can specify an amount of space for secondary allocation.

In some cases, 20% of the data component space is allocated to the high-key range rather than the normal 10%. If the SPACE subparameter is specified on both DATA and INDEX, and the specified INDEX quantity is exactly 20% of the specified DATA quantity, then the high-key range will be allocated 20% of the space available for the data component.

```
DEFINE USERCATALOG( -
NAME(USERCAT) -
FILE(ddX) -
VOLUME(USER01) -
CYLINDERS(60 5)) -
DATA( -
CYLINDERS(50 5)) -
INDEX( -
CYLINDERS(10))
```

Sixty cylinders are to be allocated to the catalog's data space. Because space is specified on both DATA and INDEX, the index value is exactly 20% of the data value; the high key range will be allocated 20% of the space available for the data component.

Estimating the VSAM Catalog's Space Requirements

Before you define a catalog, you should estimate the amount of space needed for the catalog's data component. To determine the approximate number of records required for the low-key range of the catalog, use the worksheet in Figure 33 on page 186.

Variable Quantities	Formulas	Estimates
Basic requirement = 10 records		10
A = number of key-sequenced clusters	Ax3	
A^1 = number of key-sequenced clusters with alternate indexes to be upgraded	A^1x4	
B = number of entry-sequenced clusters	Bx2	
B^1 = number of entry-sequenced clusters with alternate indexes to be upgraded	B^1x3	
C = number of relative-record clusters	Cx2	
D = number of alternate indexes	Dx3	
E = number of path entries	E	
F = number of nonVSAM data set entries	F	
G = number of generation data group entries	G	
H = number of alias entries	H	
I = number of pagespaces	I*2	
J = number of volumes, depending on device type, owned by the catalog:		
J^1 = number of 2305 volumes	J^1x2	
J^3 = number of 3330, 3340/3344, 3375, and 3380 Models A04, AA4, B04, AD4, and BD4	J^3x4	
J^4 = number of 3330 Model 11 and 3350 volumes	J^4x6	
J^5 = number of 3380 Models AE4 and BE4	J^5x8	
K = for each key-sequenced cluster and alternate index (KSDS) with space on more than two volumes, add "1" for each additional group of one to five volumes:		
K^1 = number of KSDSs with 3 to 7 volumes	K^1	
K^2 = number of KSDSs with 8 to 12 volumes	K^2x2	
K^3 = number of KSDSs with 13 to 17 volumes	K^2x3	
L = for each entry-sequenced cluster and relative-record cluster (ESDS) with space on more than five volumes, add "1" for each additional group of one to eight volumes:		
L^1 = number of ESDSs with 6 to 13 volumes	L^1	
L^2 = number of ESDSs with 14 to 21 volumes	L^2 x2	
M = for each group of four data spaces on a volume, add "1"	M	
N = number of entry records required for the catalog's data component (total of A through M above)	N	

Figure 33. Worksheet for Estimating a VSAM Catalog's Space Requirements

To define the catalog's space parameters, use one of the formats shown below:

1. To assign all the space to the catalog itself, specify:

```
DEFINE USERCATALOG -
(TRACKS(prim secn)...)
```

   where:

   - prim = the amount of space for the primary extent (allocation) of the catalog's data space. Calculate the minimum value of prim as:

     ```
     prim = (aN + b) tracks
     ```

     where N is the value derived from the worksheet (Figure 33) and the values of a and b are derived from the table below:

Catalog resides on:	a	b
2305 Model 2	0.09	3
3330 or 3330-11	0.09	3
3340 or 3344	0.125	5
3350	0.0667	3
3375	0.03	2
3380 (all models)	0.023	2

The calculated value of prim should be rounded upward to a whole number of tracks.

- secn = the amount of space for each secondary extent of the catalog's data space.

  This example of defining a catalog's space parameters shows the space specification in terms of TRACKS using the worksheet in Figure 33 on page 186. The value for prim and secn as subparameters of CYLINDERS may also be specified.

2. To have other VSAM objects in the catalog's data space, specify:

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

where:

- prim = the total amount of space for the primary extent of the catalog's data space. The amount of space assigned to the catalog itself is taken from the prim space as calculated by VSAM from the value specified by prec. To determine the amount of space that VSAM calculates for the catalog itself, use the following algorithm:

  ```
  (aN + b) tracks
  ```

  as described for Format-1 above. When specifying a value for prim larger than the requirements for the catalog itself, the remainder of the space is available for the suballocation of other VSAM objects. Using this format, the value for prim as a subparameter of CYLINDERS rather than TRACKS may also be specified.

- secn = the amount of space to be used for each secondary extent of the catalog's data space.

- prec = the number of catalog entry records in the low-key range, or the value of N from the worksheet (Figure 33 on page 186). VSAM uses this value to calculate the total primary allocation requirements for the catalog itself.

- srec = the number of catalog-entry records to be used by VSAM to calculate the catalog's secondary-extent allocation.

The VSAM Catalog's Secondary Allocation Amount

Specify a secondary allocation amount for the catalog's data component. Do not use the ALTER command to add or change a secondary allocation amount after the catalog has been defined.

The secondary allocation quantity is allocated each time one of the catalog's key ranges needs more space—up to a total of 13 times. The secondary allocation quan-

tity you specify for the catalog's data component applies to the catalog's high-key range and low-key range (including their sequence set tracks). When the high-key range or low-key range has no space left, the specified secondary allocation quantity is allocated to the key range that requires the space. When both key ranges need more space at the same time, two separate extents are allocated. Each extent is equal to the secondary allocation quantity.

Use the secondary allocation to minimize the size of the low-key range and to provide for an extension of the high-key range. Generally, a large allocation quantity for the low-key range prevents overuse of the high-key range. Specifying a secondary allocation instead of a large low-key range uses space in the high-key range when necessary. The size of the low-key range is not affected. Secondary allocation also applies to the low-key range when it requires space.

The high-level index of a catalog never requires extension.

Space Allocation for the Catalog Recovery Area (CRA)

How space is assigned in the catalog depends on whether the catalog is defined with the RECOVERABLE option. For a recoverable catalog, VSAM creates a catalog recovery area (CRA) on each volume owned by the catalog. The primary allocation for a CRA is the number of tracks equivalent to one cylinder. VSAM ensures that these tracks are contiguous, but they may or may not start on a cylinder boundary. Space allocation is determined as follows:

- On the volume containing the VSAM catalog, the primary allocation for CRA is allocated from the catalog's data space.
- On volumes without a VSAM catalog, a primary space allocation is obtained by VSAM for the CRA when the first data space is defined on the volume.
  - If the first data space is defined via the UNIQUE attribute of DEFINE CLUSTER or ALTERNATEINDEX, VSAM obtains 1 cylinder of space on the volume for the CRA.
  - If the first data space is defined via DEFINE SPACE, VSAM first attempts to obtain space for the CRA's primary and secondary allocation. However, if VSAM fails to obtain the space, the CRA is allocated from the space specified by the space parameters. If the first DEFINE SPACE for a volume specifies the CANDIDATE attribute, VSAM obtains 1 cylinder for the CRA's primary and secondary allocation.

After a primary allocation of the CRA is filled, it can expand to include a maximum of 15 additional extents. The allocation for each secondary extent is 1 cylinder.

Index Options

Five options influence performance through the use of the index of a key-sequenced data set. Each option improves performance, but some require that you provide additional virtual storage or auxiliary storage space. The options are:

- Index-set records in virtual storage
- Size of index control interval
- Index and data set on separate volumes
- Replication of index records (REPL option)
- Sequence-set records adjacent to control areas (IMBED option)

Index-Set Records in Virtual Storage

To retrieve a record from a key-sequenced data set or store a record in it using keyed access, VSAM needs to examine the index of that data set. Before your processing program begins to process the data set, it must specify the amount of virtual storage it is providing for VSAM to buffer index records. Enough space for one I/O buffer for index records is the minimum, but a serious performance problem would occur if an index record were continually deleted from virtual storage to make room for another and then retrieved again later when it is required. Ample space to buffer index records can improve performance by preventing this situation.

You ensure that index-set records will be in virtual storage by specifying enough virtual storage for index I/O buffers when you begin to process a key-sequenced data set. VSAM keeps as many index-set records in virtual storage as the space will hold. Whenever an index record must be retrieved to locate a data record, VSAM makes room for it by deleting from the space the index record that VSAM judges to be least useful under the circumstances then prevailing. It is generally the index record that belongs to the lowest index level or that has been used the least.

Size of the Index-Set Control Interval

The second option you might consider is ensuring that the index-set control interval is large enough to cover a full control area. Thus, the index-set control intervals might be larger than actually required to contain the pointers to the sequence-set level. However, this option also keeps to a minimum the number of index levels required, thereby reducing search time and improving performance. This option increases rotational delay and transfer time.

Index and Data on Separate Volumes

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

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

Replication of Index Records

You can specify that each index record be replicated (written on a track of a direct access volume as many times as it will fit). Replication reduces the time lost waiting for the index record to come around to be read (rotational delay). Average rotational delay is half the time it takes for the volume to complete one revolution. Replication of a record reduces this time; for example, if 10 copies of an index record fit on a track, average rotational delay is only 1/20th of the time it takes for the volume to complete one revolution.

On a 3340, the time usually is reduced by 50%. On a 3380, the time is reduced to 1/n, where n is the number of times the index is replicated on the track.

Because there are usually few control intervals in the index set, the cost in terms of direct access storage space is small. If the entire index set is not being held in storage and there is significant random processing, replication is a good choice. If not, replication does very little. Because its cost is small and it is an attribute that cannot be altered, it may be desirable to choose this option.

Sequence-Set Index Records Adjacent to Control Areas

When the data set is defined, you can specify that the sequence-set index record for each control area is to be embedded on the first track of the control area. This reduces disk-arm movement, because it is not necessary to do separate seeks to locate both the sequence-set index record and the data record. One arm movement enables VSAM to retrieve or store both the index record and the contents of the control interval in which the data record is stored.

When the IMBED option is chosen, sequence-set records are replicated, regardless of whether you also chose the REPL option. This means that one track of each control area is used for sequence-set records. In some situations, this may be too much space for index in relation to the data. For example, the space required for the sequence set is 1/12th of the data space on a 3340, and 1/15th of the data space on a 3380. IMBED must be specified explicitly to get the performance benefits of a replicated, embedded sequence set.

Index Option Summary

On a 3340, place the index and data on separate volumes and do not replicate or embed them. Provide index buffer space to hold the entire index set plus one sequence set when doing random processing. If direct access storage space is not a problem, if index and data cannot be put on different volumes, or if index buffer space is not available, specify REPL IMBED for random processing.

On a 3380, specify REPL IMBED and ensure that the allocation unit is in cylinders.

Copying a VSAM Catalog

Use the REPRO command to do any of the following:

- Copy a nonrecoverable catalog from one volume to another.
- Make a backup copy of a catalog.
- Reload a backup copy of a catalog.

The REPRO command can be used to copy a catalog from one volume to another; however, the input and the output objects must be catalogs. Define a catalog on the device that will contain the newly copied catalog. For example, it might be copied to move it to a faster device type or to optimize the catalog's allocation.

The receiving catalog and the source catalog do not have to be the same device type. The receiving catalog must be empty and cannot contain any entries other than the entries that describe the catalog and its data space.

To copy a nonrecoverable catalog to a recoverable catalog, you must export each VSAM data set (from the nonrecoverable catalog), then import them into the newly defined (recoverable) catalog.

To copy a recoverable catalog to a nonrecoverable catalog, either export each VSAM data set from the source catalog and then import them into the receiving catalog or use the EXPORTRA/IMPORTRA commands to accomplish the same thing on a volume basis.

To copy a recoverable catalog to another recoverable catalog, use the method above.

If the specified input and output catalogs are the same catalog, the free chain rebuild-in-place function is performed. This is the only function performed in this case. All DD statements referring to the catalog must have DISP = SHR coded, including the JOBCAT/STEPCAT, for this function. In addition, the rebuild-in-place function requires the INFILE(ddname)/OUTFILE(ddname) form of REPRO. The INDATASET/OUTDATASET (dynamic allocation) form of REPRO may not be used.

The free chain rebuild-in-place function is allowed for both recoverable and nonrecoverable catalogs.

To use the REPRO command to copy a catalog, access method services must be authorized. See "Authorized Program Facility (APF)" in *System Macros and Facilities* for information about program authorization.

Copy Catalog Preparation for a VSAM Catalog

Determine the amount of space to be allocated in the receiving catalog and how space will be allocated in the catalog to be copied.

You have to dump the catalog to determine the size of the catalog's index set or high-key range. You can however, determine the size (number of data records) of the low-key range. When you define a catalog with DATA(RECORDS...), the number of records is the number of data records the low-key range contains, unless an amount of space is specified as a subparameter of INDEX. One of the catalog's self-describing records (in the low-key range) is called the catalog control record (CCR). The CCR describes the free (or unused) control intervals in the catalog, and contains statistics about how the catalog is used. To print the CCR, issue the PRINT command, specifying the name of the catalog as the input data set.

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

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

The CCR is printed in the "dump." The record's first 44 bytes are its key value. The next byte is the character 'L', which identifies the record as a CCR-type record. The next 9 bytes are three 3-byte fields whose hexadecimal values specify:

Displacement	Symbol	Contents
45 (X'2D')	A	The highest control interval number that can be assigned.
48 (X'30')	B	The next control interval number to be assigned.
51 (X'33')	C	The number of control intervals never used and the number of control intervals that are unused because the entry they contained was deleted.

To determine the number of records in the low-key range, you also need the information in the catalog's data component record. To list the data component record, issue the LISTCAT command as shown below:

```
LISTCAT ENTRIES(catname)ALL
```

The job must include a JOBCAT or STEPCAT DD statement to describe and allocate the catalog. The name of the catalog's data component record is shown in the listing as VSAM.CATALOG.BASE.DATA.RECORD.

The number of records in the catalog is the same as the number of control intervals in the low-key range that currently contain catalog records (both base and extension records) or free records. The formula to derive the number of records varies depending on the number of extents in the low-key range. Using the LISTCAT output, find the VOLUME group for the low-key range of the data component. This information is listed under the VOLUME heading where the LOW-KEY is X'00' and the HIGH-KEY is X'3F'. Extents are shown under the EXTENTS heading for this VOLUME group. Look at the EXTENTS information to determine whether the low-key range has one or more extents.

1. If the low-key range has one extent, the value of B obtained from the CCR is the number of records to use.
2. If the low-key range has two or more extents, use the formulas shown below to obtain the number of records:

   For all low-key range extents except the last:

   ```
   ((HIGH-RBA + 1) - LOW-RBA) / 512
   ```

   For the last extent:

   ```
   b=((B x 512) - LOW-RBA) / 512
   a+b=the number of records to be used
   ```

   where:

   HIGH-RBA and LOW-RBA are the values obtained from the LISTCAT output for each extent, and B is the value obtained from the CCR.

Add the constant 19 to the number of records derived from either 1 or 2 above. This allows for the 15 self-describing records of the target catalog plus the 4 free records that are always reserved for VSAM catalog use at the end of the low-key range. The final result should be specified as the value for primary when you specify:

```
DATA(RECORDS(primaryffl  secondary"))
```

on the DEFINE USERCATALOG command.

The number of records derived from the formulas above includes any formatted free records.

- If the low-key range has only one extent, all control intervals that have never been used, plus all control intervals that are unused because the entry they contained was deleted, are formatted free records. The value contained in the CCR field described as C above includes both types of free records.
- If the low-key range has more than one extent, only those control intervals that are unused because the entry they contained was deleted, are formatted free records. This is the value contained in the CCR field described as C above. All formatted free records are copied from the source catalog to the target catalog by the copy catalog function.

When specifying the space allocation for the receiving catalog in terms of RECORDS as described above, the smallest possible receiving catalog is defined.

The value in the CCR field determines how many free records exist in the receiving catalog. These records are available to hold new entries.

The symbols A, B, and C have no meaning except in the formulas described above, which yield the number of control intervals that contain catalog records in the low-key range. The complete format of the CCR record is included in *VSAM Logic*.

Copy Catalog Procedure for a VSAM Catalog

To copy a catalog, you need to:

1. Use the DEFINE command to define a catalog into which the source catalog is to be copied. The preceding section describes a method of determining in records the smallest possible receiving catalog.
2. Use the REPRO command to copy the source catalog to the receiving catalog. You cannot specify any of the REPRO command's delimiters (FROMKEY, TOKEY, SKIP, or COUNT) when you use REPRO to copy a catalog. A concatenated JOBCAT or STEPCAT DD statement is required to describe and allocate both the source and receiving catalogs. When copying the master (source) catalog, use a single JOBCAT or STEPCAT DD statement to describe and allocate the receiving catalog. A DD statement is not required to describe and allocate the master (source) catalog. If a DD statement is included, an error message results.
3. Use the EXPORT command to disconnect the source user catalog from the master catalog. This step is not necessary if the source catalog is the master catalog.

   If the source catalog was a user catalog, list the aliases of the user catalog in the master catalog before you issue the EXPORT command. Use the LISTCAT command to list the aliases:

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

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

   If you use the IMPORT command to make the catalog available to another system, issue the DEFINE ALIAS command to reestablish each of the catalog's aliases.
4. If the source catalog was a user catalog, use the DEFINE ALIAS command to establish the required aliases of the receiving catalog.
5. Use the DELETE command to remove the source catalog from the receiving volume. The source catalog appears in the receiving catalog as a cluster as a result of the copy operation.

The DELETE CLUSTER, a special form of DELETE, removes the source catalog records from the receiving catalog. It results in the following:

- The source catalog records are deleted from the receiving catalog.
- The source catalog's Format-1 DSCB is changed to a suballocatable data space. It no longer indicates that the volume contains a catalog.
- The receiving catalog now owns the volume that contained the source catalog.

After this delete, the source catalog no longer exists. If you do not want the new catalog to own the source catalog volume, use DELETE SPACE for the source catalog to remove volume ownership from the new catalog.

The copy catalog procedure should be performed with caution. Until the source catalog is disconnected from the master catalog, two catalogs are available for use.

If you are copying the master catalog, special care should be taken that the job is executed when the system is otherwise quiesced. Otherwise, it is possible that updates could be made to the master catalog during the copy. To use the receiving catalog as a master catalog, the following steps should be taken following step 2 described above:

1. Replace the SYSCATLG member in SYS1.NUCLEUS (now cataloged in the target master catalog) so that the SYSCATLG member points to the new master catalog.
2. Re-IPL the system. With the SYSCATLG member changed, the IPL will reference the new master catalog.
3. Proceed with step 5.

Backing Up a VSAM Catalog for Recovery

You can use the REPRO command to unload (make a backup copy of) a catalog. If the catalog becomes inaccessible, you can use REPRO to reload the copy. See "Unloading a VSAM Catalog" on page 196 and "Reloading a VSAM Catalog" on page 196, which describe catalog unload/reload. The backup copy may not reflect current information for some of the cataloged objects.

Backing Up the VSAM Master Catalog

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

Because the system requires a master catalog, the reload procedure described in "Reloading a VSAM Catalog" on page 196 requires modification for reloading the master catalog. If you have a master catalog to reload and it was not defined as a recoverable catalog, you may reload the master catalog with REPRO. For catalogs defined as recoverable, see "Restoring the Catalog Entry Obtained Using the EXPORTRA Command" on page 221. When only some of the entries in the master catalog are inaccessible, and the master catalog itself is still operational, you may be able to reload the backup catalog into it. Otherwise, you can get a master catalog by:

- Using Device Support Facility (on another system) to restore the volume that contains the master catalog (from a tape onto which you have previously dumped the volume). You can do an IPL (initial program load) to bring your system up with the restored volume and then use REPRO to reload the backup into the restored version of the catalog. If the backup is no more recent than the restored catalog, you can use the restored version without reloading.
- Using DUMP/RESTORE, design the volume that contains the master catalog to contain information that changes very little, so that you will not lose changes when you restore the volume. To avoid needing another system to restore the

volume from a tape, you can have two direct access volumes (with the same volume serial number) that contain a copy of the master catalog.

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

Dumping a VSAM Catalog and Its Data Sets

This procedure calls for periodically dumping the volume(s) upon which a catalog and its data sets are stored, with Device Support Facility (or with some other utility that achieves the same effect). Then, if the catalog is lost or somehow becomes inaccessible, you can restore the volumes (or alternative volumes with the same volume serial numbers). The volumes will contain what they contained when the backup copies were made. See "Updating a Backup Catalog" on page 204 for information on how to bring the restored catalog up to date.

When you use Device Support Facility to dump a volume containing a VSAM catalog, ensure that all update activity (for example, DEFINE, DELETE, ALTER, data set extension, etc.) is quiesced. If any catalog update activity occurs during the dump operation, the chain of free records in the catalog's low key range would be damaged in the backup copy. If the backup copy is restored, the damaged chain will be introduced into the restored copy.

If some of the space on a volume does not belong to a VSAM data space, the system may allocate that space to non-VSAM data sets. With DUMP/RESTORE, you will have to fall back to a previous copy of both the VSAM data sets and catalog, and the non-VSAM data sets.

When you use Device Support Facility to dump or restore a volume that contains VSAM objects (clusters, catalogs, data spaces, and alternate indexes), you must include a JOBCAT or STEPCAT DD statement with the Device Support Facility job or jobstep. This statement describes the VSAM catalog that owns the volume. This volume's table of contents (VTOC) indicates that part of the volume is owned by a VSAM catalog. The passwords for the password-protected objects are part of the object's catalog information. However, the volume's VTOC does not identify the catalog that contains the passwords. Before a volume is dumped, the system asks the operator for the correct password of each password-protected object on the volume. However, if the volume is RACF protected, RACF read authorization to the volume overrides VSAM password protection.

Use the REPRO command to unload a VSAM catalog into a key-sequenced, entry-sequenced, or sequential (SAM) data set. If the catalog becomes accessible, redefine the catalog and use REPRO to reload the backup. If the catalog is accessible, you can use REPRO to reload the backup and reestablish catalog entries. Because it is difficult to recover VSAM data spaces, page spaces, and data sets that have extended after a backup copy of the catalog is made, reloading a catalog should be done carefully.

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

Use a STEPCAT DD statement to identify the catalog that you are loading or reloading.

To use the REPRO command to unload or reload a catalog, access method services must be authorized. See "Authorized Program Facility (APF)" in *System Macros and Facilities* for more information. Using REPRO to unload or reload a password-protected catalog requires the catalog's master password.

Unloading a VSAM Catalog

A sequential, key-sequenced, or entry-sequenced backup copy of a catalog is inaccessible as a catalog. Because there is little advantage in having the backup on a direct access volume, it is convenient to use magnetic tape to copy a catalog in a sequential data set.

To unload a catalog into a key-sequenced or entry-sequenced data set, first define the data set in another catalog for protection.

To continually have a current backup copy available, unload each catalog periodically. With tape, you can easily alternate two or more volumes for several levels of backup for each catalog.

Use LISTCAT before unloading a catalog. You can compare the listing with the one you obtain after reloading. If you are unloading a recoverable catalog, use LISTCRA with the COMPARE option to ensure that the catalog and its volumes are synchronized at the time of the unload operation.

An example in "REPRO Examples" in *Access Method Services Reference*, describes the DCB parameters you must specify on your DD statement if you unload your catalog to a non-VSAM sequential (SAM) data set.

Reloading a VSAM Catalog

Use REPRO to reload the backup copy into a target catalog with the same name, volume serial number, and device type as the source catalog. Reloading the master catalog has special requirements. (For details, see "Backing Up the VSAM Master Catalog" on page 194.)

The target catalog can be either a version of the source catalog, or a newly defined user catalog (after using EXPORT DISCONNECT to remove the user catalog entry in the master catalog). The primary allocation of the newly defined catalog must be able to hold at least as many records as the source catalog held at the time the backup copy was made. Catalog extension will not take place during the reload operation. If the newly defined catalog is not large enough, your reload job will fail. The same method as that described in "Copy Catalog Preparation for a VSAM Catalog" on page 191 can be used to determine the smallest possible size of the newly defined catalog. However, you must make the calculations based on the original catalog at the time the backup was made.

Reloading a version of the source catalog results in a catalog equivalent to the original one at the time the backup was made. Reloading replaces entries in the target catalog with entries of the same name in the backup. It inserts entries into the target that exist only in the target. During reload, access method services issues a maximum of 100 messages to indicate entries that exist only in the target or only in the backup.

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

After you reload a catalog, use LISTCAT to list its contents. Run LISTCAT in a separate job step so that the catalog will be closed after it is reloaded (to update its self-defining information). Compare the listing with the one you obtained before unloading the original catalog to ensure that you have used the right backup.

Note: LISTCAT cannot run in a job step where the catalog is empty when opened. To ensure that the LISTCAT correctly reflects the contents of the catalog, it should be run as a separate job step.

Reloading or restoring a recoverable catalog does not cause the catalog recovery area to be updated. Therefore, the LISTCRA command with COMPARE option should be run to identify mismatches between the catalog and the catalog recovery area (CRA). (This should be done in a separate job step, immediately after the reload.) These mismatches should be resolved before the catalog can be used. No other jobs should be run between the reload and the LISTCRA if the jobs access any of the data sets cataloged in the reloaded catalog.

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

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

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

The data in any extents that have been acquired by the data set since the catalog was backed up is unrecoverable. For an entry-sequenced data set, the data in any new extents should consist only of records that have been added to the end of the data set. Therefore, it is possible to recover all the data in the old extents by accessing the data set sequentially up to the end of the old physical space allocation. For a key-sequenced data set, the new extents may be any portion of the data set because of control area splits. An attempt to read the data in logical sequence will fail with an invalid RBA indication when the data in the new extents is reached. You could access the key-sequenced data set by means of address sequence, but you then have the problem of identifying the missing records. Individual data set recovery will be necessary for those data sets affected.
See "Updating a Backup Catalog" on page 204 for a discussion of making the contents of the backup catalog agree with the contents of the original catalog at the time it became inaccessible.

Optimizing the Performance of Unload/Reload

You can specify additional I/O buffers for unloading and reloading by using:

- The AMP parameter in the STEPCAT DD statement that identifies the catalog—AMP='BUFND=x,BUFNI=2', where x equals 2 times the number of 512-byte control intervals per track of the device used for the catalog.
- The AMP parameter in the DD statement that identifies a key-sequenced or entry-sequenced backup copy—AMP='BUFND=x,BUFNI=2', where x equals 2 times the number of 512-byte control intervals per track of the device used for the backup.
- 2 times the number of physical records per track of the device used for the backup (when the backup is on a direct access volume).

Block the records in a sequential backup data set. Some catalog records are 47 bytes long; the rest are 505 bytes long. Use DCB=RECFM=VB.

Displaying the VSAM Catalog's Contents

The LISTCAT command is used to list entries from a catalog. The entries listed can be selected by name or entry type, and the fields to be listed for each entry can additionally be selected. For an explanation of the output produced as a result of the LISTCAT commands, see *Access Method Services Reference*.

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

To specify an entry by generic name, supply all but one qualifier of the name. The qualifier omitted is indicated by an asterisk (*). The first qualifier cannot be indicated by an asterisk.

If you specify ENTRIES(A.*), all two-qualifier entry names that have an A for the first qualifier are selected to be listed.

If you specify ENTRIES(A.*.B), all three-qualifier entry names that have A as the first qualifier and B as the last qualifier are selected to be listed. You can further limit the printing by identifying certain entry types to be printed (non-VSAM, cluster, etc.).

However, if you specify LEVEL(A.*.B), all entry names that have an A as the first qualifier and B as the third qualifier are selected to be listed; some of the selected

entry names might have four or more qualifiers (each must have at least three qualifiers and satisfy the A.*.B selection criterion).

Also, if you specify LEVEL(A), all entry names that have an A as the first qualifier (regardless of the number of qualifies) are selected to be listed.

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

Restriction for use of REPRO and LISTCAT in the same job step: LISTCAT cannot run in a job step where the catalog is empty when it is opened. To ensure that the LISTCAT correctly reflects the contents of the catalog, it is recommended that LISTCAT be run as a separate job step.

Changing the Volume Serial Number

When you change the volume serial number of a volume that contains VSAM objects, use the following procedure so the VSAM objects can be located when the object's catalog information is referenced:

1. Issue the EXPORT PERMANENT command for each VSAM object (alternate index and cluster) on the volume. If the volume also contains a VSAM catalog, issue an EXPORT DISCONNECT command to disconnect the user catalog from the master catalog. The EXPORT command copies each VSAM object and its catalog entry onto a movable volume. The object's entry in the user catalog is deleted.
2. Issue the DELETE command to delete all empty data spaces on the volume, and to delete the volume entry from the catalog. Non-VSAM data sets are described in the volume's VTOC and are not affected by the DELETE command.
3. If the volume contains an empty catalog that only describes space on the volume, delete the catalog. If the empty catalog describes space on more than its own volume, you can delete the space on each volume first, then delete the catalog.
4. Execute the Device Support Facility (ICFDSF) program with the LABEL statement to change the volume's serial number. You might reorganize the non-VSAM data sets remaining on the volume, so the space available for the VSAM data space is contiguous.
5. If the catalog was deleted in step 3, it must be redefined when beginning this step. Issue the DEFINE SPACE command to build a data space on the volume, to establish the VSAM catalog's ownership of the volume, and to build a volume entry that points to the volume with its new serial number. The data space should be large enough to contain all the suballocated VSAM objects removed from the volume during step 1.
6. Issue the IMPORT command for each VSAM object removed from the volume during step 1. Specify the new volume information for the object.

Altering Attributes of Entries in a VSAM Catalog

The ALTER command is used to alter attributes in catalog entries and to rename members of non-VSAM partitioned data sets. To alter an entry, you must supply its name and the attributes to be altered.

Altering an entry does not normally require that the entry's volume be mounted, because the entry's use of space and the availability of space in the volume's data spaces can be determined by examining the catalog. The entry's volume must be mounted whenever the volume's VTOC must be consulted or modified, such as when a unique component or a non-VSAM data set is renamed. The entry's volume must also be mounted when the entry is cataloged in a recoverable catalog. A JCL DD statement can be used to cause the data set or volume to be allocated. A data set or volume can be dynamically allocated by specifying the data set name or volume serial number if no DD statement is supplied. The volume must be mounted as permanently resident or reserved.

Most attributes of the VSAM data set are associated with its data or index components. Certain attributes such as retention period, owner ID, and cluster protection are associated with the cluster entry. If you use the cluster name in the ALTER command, only the cluster name attributes are changed. If you use the cluster name in the ALTER command to alter any attribute not associated with the cluster entry, access method services terminates the ALTER request and issues an error message. The reverse is also true: If you specify a data or index component name and attempt to alter an attribute associated with a cluster entry, access method services terminates the ALTER command and issues an error message. You must specify the explicit data or index component name to alter any of the remaining attributes such as share options, buffer space, write check, the data, or index protection attributes. You should specify the data and index component names at DEFINE time to facilitate the use of the ALTER command. Otherwise, you have to use the long system-generated names. To determine which values or attributes you may alter for a particular entry, see "ALTER" in *Access Method Services Reference.*

You cannot ALTER any attributes of a user catalog except those associated with the cluster name. Data and index component names specified for a user catalog at DEFINE time are meaningless, because the system always generates standard names. If you specify the system-generated names for the data or index components in the ALTER command, VSAM returns a return code of '8', claiming the entry is not there. If you alter the expiration date to an invalid date, such as '999999', access method services returns a nonzero condition code; the expiration date remains unchanged.

Generic Names and ALTER

To alter entries with qualified names (for example, PAYROLL.74.MAY), you can identify the entry with its full name (all qualifiers) or with all qualifiers but one. The unspecified qualifier is indicated with an asterisk (for example, PAYROLL.*.MAY). You must always specify the first qualifier of a qualified entry name. This kind of shortened name, PAYROLL.*.MAY, is called a generic name. If you specify full entry name, only that entry is modified. If you specify all qualifiers of the entry name but one, the entries whose entry names match the supplied qualifiers are altered.

For example, when you specify PAYROLL.*, the entries that might be altered are the entry names that contain two qualifiers; the first qualifier is PAYROLL. When

you specify PAYROLL.*.MAY, the entries that might be altered are the entry names that contain three qualifiers; the first is PAYROLL and the third and last are MAY.

When you identify a catalog (by specifying the ALTER command's CATALOG parameter), access method services searches only the specified catalog for generically named entries. If no catalog is specified, the catalog is selected as indicated in "Order of Catalog Use: ALTER" in *Access Method Services Reference* and is searched for generic name entries.

You must identify all qualifiers in the qualified name. For example, PAYROLL.* cannot be used to identify a qualified name that contains three or more qualifiers, even though its first qualifier is PAYROLL. PAYROLL.81.MAY.* cannot be used to identify a qualified name that has more or fewer than four qualifiers.

Renaming Generically Named Entries

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

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

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

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

VSAM Catalog Performance

Sharing Services with User Catalogs

A large number of concurrent requests for information (that is, for catalog entries) from a VSAM catalog might result in some of the requests being answered more slowly than they would be if the entries were distributed among several user catalogs. You might have the VSAM master catalog primarily contain pointers to user catalogs, which would contain entries for most data sets, indexes, and volumes. By decentralizing data set entries, you also reduce the time required to search a given catalog and minimize the effect of a catalog's being inoperative or unavailable.

Improving Catalog Performance

To improve catalog performance, you can:

- Mount the catalog volume on an unsharable direct access device.
- Define entries into a catalog that is not protected with an update-level password. This results in greatly improved performance when you are using the CNVTCAT command to convert OS catalog entries to VSAM catalog entries.
- Specify a large buffer space value when defining your catalog. This is an important factor in catalog performance, especially if there are many concurrent users. The value specified via BUFFERSPACE helps determine the number of catalog RPLs to be set up at catalog OPEN time. An insufficient number of request parameter lists (RPLs) can cause subsequent users to have to wait until an RPL is available. Also, a large buffer space can result in catalog index control intervals staying in main storage, thus avoiding I/O to the index.

Note: Any change that causes the index control interval size to increase, may also increase the value specified for BUFFERSIZE to maintain the number of RPLs needed for optimum performance. Regardless of the BUFFERSIZE specified, the number of RPLs is within a minimum limit of 2 and a maximum limit of 7.

For more details about catalog performance and operation, see *Access Method Services Reference*.

Performance Measurement

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

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

- Control interval size
- Percentage of free control intervals per control area
- Number of bytes of available space (includes distributed free control intervals and allocated space beyond the last used control interval)

- Length and displacement of the key
- Maximum record length
- Number of levels in the index
- Number of extents
- Number of records retrieved, added, deleted, and updated
- Number of control interval splits in the data and in the sequence set of the index
- Number of EXCPs that VSAM has issued for access to a data set

Note: When a cluster or component is exported, that is, is named in an EXPORT command, the statistics are exported with the catalog record. When the cluster is imported (with the IMPORT command), it is reorganized. Its old statistics aren't lost—they merely don't apply to the reorganized data. When the cluster is loaded (as a result of IMPORT), its statistics are revised to reflect the newly loaded cluster.

VSAM Catalog Backup and Recovery

Because of the importance of the VSAM catalog, you should consider backup for the catalog as well as for the individual data sets. In theory, if all the data sets in the catalog are backed up individually, as they should be, it is possible to recover from destruction of the catalog by carrying out recovery procedures for each of the data sets. In practice, this may be reasonable. The probability of losing an entire catalog is very low.

However, to speed recovery or minimize exposure in the event of catalog damage or destruction, three tools are available: catalog unload and reload using the REPRO command; catalog recovery (from catalog recovery areas on volumes owned by recoverable catalogs) using the RESETCAT command; and recovery of data and its associated catalog information (from catalog recovery areas on volumes owned by recoverable catalogs) using the EXPORTRA/IMPORTRA commands.

Catalog Unload and Reload

You can schedule catalog backups to minimize the exposure for critical data. Some events that may have changed the catalog since the last backup are the execution of a DELETE, permanent EXPORT, DEFINE, or IMPORT command, or the extension of a data set owned by the catalog.

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

If VSAM data sets or data spaces have been defined or imported since the last catalog backup and the catalog is reloaded or restored, then the defined data sets or data spaces will not be defined in the reloaded or restored catalog. Processing these data sets or data spaces by means of the restored catalog is not possible because they cannot be accessed. The space formerly occupied by these VSAM data sets or data spaces will not be usable, but may be recovered by scratching the Format-1 DSCBs in the VTOC for the data spaces. If any volumes were added to the catalog

(between the backup and the recovery), they will also be unusable until you use the DELETE space command with the FORCE option.

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

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

You can monitor the growth of a data set with the LISTCAT command or the SHOWCB macro. (The use of the SHOWCB macro to display statistics is described in *VSAM Administration Guide*.)

Updating a Backup Catalog

Changes made in the original catalog between the time it is loaded to make the backup and the time it becomes inaccessible are not present in a reloaded or restored backup catalog. Three types of changes may have occurred:

- Entries may have been deleted by way of DELETE or permanently exported by way of EXPORT PERMANENT. You can remove these entries from the backup catalog with DELETE NOSCRATCH, described in "VSAM Catalog Cleanup" on page 206.
- Entries may have been defined by way of DEFINE, imported by way of IMPORT, or defined (non-VSAM entries only) by the scheduler. You can use DEFINE NON-VSAM to bring the backup catalog up to date on non-VSAM entries. But a VSAM object defined by an entry missing from the backup catalog is no longer accessible. To regain the use of the space on a volume or in a VSAM data space for entries missing from the backup catalog, use ALTER REMOVEVOLUMES (described in "VSAM Volume Cleanup" on page 205).

  To recover objects that are lost, redefine them and rerun the jobs that loaded and updated them.
- Entries may have been updated to describe extensions. For an extension using space allocated to an object before the original catalog was unloaded to make the backup, use the VERIFY command to update the object's end-of-file indicator. For an extension using newly allocated space, you cannot update the backup catalog to recover the data in the extension.

VSAM Volume Cleanup

The VSAM volume recovery function of the ALTER command removes all VSAM data spaces and VSAM data sets from specified volume(s).

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

The ALTER command with the REMOVEVOLUMES parameter enables you to remove all the VSAM data spaces on a volume without gaining access to the catalog that owns the volume. ALTER REMOVEVOLUMES overwrites the data spaces with binary zeros, rewrites the VTOC to remove the data spaces' Format-1 DSCBs, and turns off the VSAM ownership bit in the Format-4 DSCB.

When the user catalog is on the volume, ALTER REMOVEVOLUMES overwrites it. You can use ALTER REMOVEVOLUMES to clean up a volume that contains an inaccessibleuser catalog. Use EXPORT DISCONNECT to remove the catalog's entry in the master catalog before using ALTER REMOVEVOLUMES.

ALTER REMOVEVOLUMES does not remove non-VSAM data sets from a volume, nor does it remove VSAM objects from a volume owned by the master catalog. See "VSAM Volume Recovery Function" on page 215 for details of how to specify the ALTER REMOVEVOLUMES function.

Notes:

- ALTER REMOVEVOLUMES is also used to remove from a catalog ownership of a candidate volume that does not yet contain a VSAM data space and is not referenced by any VSAM objects.
- Do not use ALTER REMOVEVOLUMES to remove VSAM objects from a volume owned by a catalog that you can still use. Use the DELETE command.
- If you use the ALTER REMOVEVOLUMES function for a volume containing a user catalog, ensure that the user catalog is not open. Include a DD statement that identifies the user catalog and specifies DISP = OLD in the job step that includes the ALTER REMOVEVOLUMES command. The user catalog should not be identified in:
  - A JOBCAT DD statement in the ALTER REMOVEVOLUMES job step.
  - A STEPCAT DD statement in the ALTER REMOVEVOLUMES job step.
  - Any other DD statement(s) or access method services command parameter(s) that would cause the catalog to be allocated and opened before the ALTER REMOVEVOLUMES command is executed. (For example, use of a data set name with a high-level qualifier that is the alias of the user catalog.)
- Message IDC0526I with return code 160 will be issued if you try to use:
  - The ALTER REMOVEVOLUMES function on a volume owned by a VSAM master catalog
  - The ALTER REMOVEVOLUMES function on a volume whose VVDS data set is defined in an integrated catalog facility master catalog.

If the user catalog that owns the volume is available, use the DELETE SPACE command with the FORCE option to get rid of all VSAM data spaces on the volume whether or not they contain VSAM data sets. DELETE SPACE FORCE deletes all VSAM data spaces, marks any VSAM data sets contained therein as unusable in the catalog, and releases ownership of the volume. If your volume contains VSAM data sets you want to save and VSAM data space(s) you want to eliminate, use the EXPORT command to export the data sets. Then use DELETE SPACE FORCE to eliminate all VSAM data spaces on the volume. After redefining the data spaces, reestablish your data sets by using the IMPORT command.

Do not use DELETE SPACE FORCE for a volume that contains a user catalog. Use DELETE USERCATALOG FORCE to DELETE a user catalog without first deleting all its entries. All data spaces, including the catalog volume itself, are deleted from each volume owned by the catalog, and volume ownership is released by VSAM.

VSAM Catalog Cleanup

When a volume becomes inaccessible, use the DELETE command with the NOSCRATCH parameter to remove the entries for the VSAM objects from the catalog that owns the volume. Entries for VSAM objects that are deleted from the original catalog after it is unloaded to make the backup can also be removed from a backup catalog with DELETE NOSCRATCH. DELETE NOSCRATCH removes an entry without gaining access to the volume indicated in the entry. If the cluster, alternate index, or pagespace is defined with the UNIQUE attribute, a Format-1 DSCB is written on the object's volume. If you specify NOSCRATCH and the Format-1 DSCB still exists, it will not be erased from the VTOC. (DELETE without NOSCRATCH specified allows access method services to gain access to a volume to change the VTOC or overwrite an object with binary zeros.)

VSAM cleans up the catalog:

- When DELETE CLUSTER, DELETE ALTERNATEINDEX, or DELETE PAGESPACE is specified, VSAM deletes the cluster, alternate index, or page space entry. (**Note:** NOSCRATCH cannot be specified for these entry types if they are defined in a recoverable catalog.)
- When DELETE SPACE is specified, VSAM deletes the volume entry. The volume entry must be empty. (First delete the entries of all VSAM objects indicated in the entry.)

Automatic Catalog Backup

All VSAM catalogs can be defined with the RECOVERABLE attribute that makes it possible to recover VSAM data sets and their catalog entries if the catalog is damaged or destroyed. Recovery done on a volume and a catalog produces a copy of all catalog information. Recovery information is recorded on each volume owned by the catalog. Space for this information is automatically set aside when you acquire volume ownership on a new volume and when you define the catalog. There is no separate catalog entry for the recovery space: VSAM records its physical track address in the volume's Format-4 DSCB.

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

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

RESETCAT is another catalog recovery tool. If inconsistencies develop between your catalog and CRAs of its owned volumes, consider RESETCAT as a recovery vehicle. RESETCAT synchronizes a recoverable catalog and its associated CRAs. If your catalog or any of its owned volumes become unusable, you can restore a backup volume. Inconsistencies may exist between the catalog and the CRAs of the volume it owns. Either the catalog or the CRAs will be down level. RESETCAT provides the necessary synchronization facility to ensure consistency between the catalog and its volumes. RESETCAT confines its processing to the catalog and CRAs. VSAM data sets are unaffected by this operation.

You can use the LISTCRA command to list the contents of the CRA before selective recovery or to list the entries in the recovery area that are different from those in its associated catalog.

Regaining Access to Data

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

- Data set not properly closed
- Inaccessible data set
- Unusable catalog
- Inaccessible volume

Use the LISTCRA command with the COMPARE option to identify a mismatch between the CRA and the catalog. The records in the CRA are used as a basis for comparison. When a mismatch is detected, LISTCRA prints the CRA record, then identifies the mismatch by printing asterisks below the mismatched area(s) of the records. LISTCRA, EXPORTRA, and IMPORTRA, are usable only with recoverable catalogs. The mismatches detected by LISTCRA vary in their degree of seriousness. Figure 34 and Figure 35 on page 208 list (ordered by severity) the type, cause, and severity of mismatches. Only the most serious mismatch is identified.

Other sections in this chapter tell how to use corrective measures to recover. Appendix C in *Access Method Services Reference* has an example of the LISTCRA output using the DUMP COMPARE options.

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

Figure 34. Catalog Volume Records

Message	Type	Cause	Severity
CATALOG ENTRY HAS DIFFERENT NAME	Mismatched name	The catalog or the volume containing the data set was restored. As a result, the CRA record points to a record in the catalog which no longer contains the same object.	Requires data set recovery.
VOLUME OR KEYRANGE	Mismatched volume or key range.	The catalog or the volume containing the data set was restored. As a result, the object's volume locations or key ranges in the CRA do not match those in the catalog.	Requires data set recovery.

Figure 35 (Part 1 of 2). VSAM Object Records

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

Figure 35 (Part 2 of 2). VSAM Object Records

In general, there are two types of data recovery: repair and reset.

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

The reset operation restores addressability and access to a version of the data other than the most recent. Reset operations are generally used to correct logical problems such as programming errors and faulty transactions. Reset is the most common form of recovery, because of the types of problems encountered and the level of data available for recovery; for example, a reset operation dumping (copying) and restoring a volume.

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

EXPORT/IMPORT	Reset
REPRO	Reset/repair
EXPORTRA/IMPORTRA	Repair (recoverable catalogs only)
COPY AND RESTORE	Reset
LISTCRA (COMPARE)	Analysis (recoverable catalogs only)
DELETE UCAT(FORCE)	Reset/repair
ALTER REMOVEVOLUMES DELETE SPACE(FORCE)	Reset/repair

NOSCRATCH	Reset/repair
DELETE CLUSTER NOSCRATCH	Reset/repair (MVS only)
VTOC UTILITY (SUPERZAP)	Reset/repair
DELETE SPACE (FORCE)	Reset/repair
VERIFY	Repair
RESETCAT	Reset/repair (recoverable catalogs only)

REPRO can be used to create a backup copy of the catalog. The catalog can be reloaded (using REPRO) from this backup copy and can be used to minimize the amount of work required to accomplish recovery. The usefulness of a backup copy depends on what modifications are made to the catalog after the backup is created. The following actions that may have occurred before taking a backup copy, affect the usefulness of the backup catalog:

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

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

Suballocating space to a VSAM data set: The volume space map is invalidated, which, in itself, is not serious. However, the data set entry is also invalidated, which is a serious problem.

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

- The volume being restored doesn't contain multivolume data sets.
- If the volume being restored contains a portion of a multivolume data set, the entire set is restored as a single unit all volumes containing portions of the multivolume data set.

Data Set Not Properly Closed

VSAM data sets are not properly closed if they were opened for output and a system failure occurred, or if a program that is open for output terminated abnormally. This condition is reflected in the catalog and is communicated to the next program that does an OPEN of the data set. It acts as a warning in that, although the data set may actually have been properly closed, an error condition such as an incorrect high RBA in the catalog, an incomplete write to a direct access device, or duplicate data may exist.

If an error exists, it is probably an incorrect high RBA in the catalog. The VERIFY command, which is used to correct this condition, scans a given data set starting from the catalog-specified high RBA to the end of the data set. The resultant high RBA is then used to update the catalog.

You can avoid an incomplete write to a direct access device and duplicate data either by doing synchronous direct inserts or by using abnormal termination exits in which you issue a CLOSE or TCLOSE to close the data set properly.

If you suspect that a write operation is incomplete, you can issue either an IMPORT or a REPRO command to get an old copy of the data; intermediate updates or inserts are lost. The use of IMPORT or REPRO requires that you have a previously exported version of the data set available.

Duplicate data in a key-sequenced data set, the least likely error condition to occur, can result from a failure during a control interval or control area split. If the failure occurred before the index was updated, the insert is lost, no duplicate data exists, and the data set is stable and usable.

If the failure occurred between updating the index and writing the updated control interval into secondary storage, some data is duplicated. However, both versions of the data are accessible by using addressed processing. The condition can be corrected by issuing a REPRO or an IMPORT command. If you want the current version of the data, you can use the REPRO command to copy the current version to a temporary data set and again to copy it back into a newly created key-sequenced data set. If you have a backup copy of a previous version of the data, you can use the IMPORT command to obtain a reorganized data set without duplicate data.

If the index is replicated and the error occurred between the write operations for the index control intervals but the output was not affected, both versions of the data can be retrieved. The condition is similar to that described in the preceding paragraph and the same recovery measures can be taken.

The sequence of operations for a control area split is similar to that for a control interval split; the possible error conditions and corrective actions are the same.

Although the likelihood of having duplicate data is small, you can further reduce it by specifying free space for both control intervals and control areas to reduce the problem of splits. The only warning indication that VSAM sets for this condition is "data set not properly closed." If a more positive indication is desired, you can obtain it by using the journal exit (JRNAD) to determine control interval and control area splits and the RBA range affected.

To summarize, the warning "data set not properly closed" may indicate an error in a VSAM data set. This condition can generally be corrected by using the VERIFY command. If other errors are encountered or suspected, they can generally be corrected by using either the IMPORT or the REPRO command.

Inaccessible Data Set

A VSAM data set may become inaccessible because of damage to the data set itself, to related information in the catalog, or to both. Depending on the extent of damage and prior actions, it may be possible to get access to either the current or a previous version of the data. A data set is inaccessible when it cannot be opened or is either partially or completely unreadable.

If the data set cannot be opened, there is probably damage to the catalog. To determine the extent of this damage, you can use either the LISTCAT or the LISTCRA (with the COMPARE option) command, the latter only if the catalog is recoverable. If, as a result of processing one of these commands, you find only local damage to a small number of data sets and no serious damage to volume information (that is, either there is no mismatch or a general mismatch has occurred), you can use one of the following procedures:

- If an exported copy of the data set is available, you can import it to gain access to the level of data at the time the backup copy was made. If the catalog is recoverable and you want to gain access to the current level of data, you can use the EXPORTRA command to extract the data, and you can then reestablish it by using the IMPORTRA command. It is not necessary to do any volume

cleanup prior to reestablishing the data, because the volume information was not seriously damaged.

- If the data set can be opened but none of the data can be retrieved, either the data set has been destroyed or the catalog and volume are not synchronized. To verify the condition of the catalog, you can use either the LISTCAT or LISTCRA (with the COMPARE option) command. If the results indicate catalog damage, you can use the above procedure to gain access to the data. If no catalog damage is indicated, you can import a previously exported version of the data. REPRO can also be used to effect a recovery if a copy of the data set is available.

  If the data set can be opened and partially read, the problem is either confined to the data set itself or an entire physical extent of the data set is not readable. If the latter occurs (where the catalog indicates one or more extents than there are on the volume), it may have been caused by a restore of a volume independent of the catalog. You can use the LISTCRA (with the COMPARE option) to verify the mismatch in the number of extents. If this type of mismatch is not verified, you can then issue either an IMPORT or a REPRO command to correct the problem, using a backup copy of the data set. If the problem turns out to have been in the catalog, you can use either IMPORT or REPRO, or EXPORTRA followed by an IMPORTRA.

To summarize, the inaccessibility of a VSAM data set can either be a local problem restricted to a small number of data sets, or it can be a more serious problem. This can be determined by the use of the LISTCRA command. If the problem is local, the EXPORTRA, IMPORTRA, IMPORT, and REPRO commands can be used to correct the problem.

Unusable Catalog

A catalog may become unusable because of physical damage to the catalog volume. Depending on the extent of the damage and prior actions, it may be possible to either repair or reset the catalog and the data it controls. A catalog is unusable when many VSAM data sets cannot be opened, the catalog itself cannot be opened, or the catalog volume is not usable.

If the catalog can be opened, but many VSAM data sets controlled by this catalog cannot be accessed, there is probably a problem with the catalog. To determine whether it is a catalog problem, either a LISTCAT or a LISTCRA (with the COMPARE option) can be used. If I/O errors are encountered or mismatches are detected, some form of catalog recovery is required. If not, the problem is confined to the data sets themselves and the procedures given for unusable data sets can be used.

If the problem is with the catalog, recovery depends on the availability of backup copies of the catalogs, volumes, and data sets, and whether the catalog has been defined with the RECOVERABLE attribute. This section first discusses recovery of catalogs without associated catalog recovery areas (CRAs), then catalogs with associated CRAs.

Catalogs without Associated CRAs: Recovery by way of an image copy of the data set must reestablish usable catalog entries. One way to reestablish catalog entries is to save a backup copy of the catalog along with the data. The major drawback to this approach is that other data sets defined in the catalog may become unusable as a result of the catalog reload. For example, assume that a backup copy of a data set and its corresponding catalog have been restored, but other data sets

defined in the same catalog have not. To avoid mismatch problems, you can use the reloaded catalog to unload the reloaded data set, using the REPRO or EXPORT command. You can then use the current catalog to reload the data set, using REPRO or IMPORT, while other data remains unaffected.

This procedure is useful where non-VSAM data sets on the same volumes as VSAM data sets are routinely backed up. It requires only one backup operation for multiple VSAM data sets, without a recovery operation for undamaged data sets. Recovery should be needed infrequently, if at all, so the extra time required for this procedure would be more than offset by the saving in routine backup time.

Catalogs with Associated CRAs: If an unloaded copy of the catalog built by REPRO or a backup copy of the catalog volume is available, you can do the following:

- Either reestablish the backup copy of the catalog or restore the backup copy of the catalog volume.
- Use LISTCRA with the COMPARE option to identify mismatched volumes and data sets.

If the volume entries indicate mismatched volumes (data set directory or data space group mismatched), you can use RESETCAT to reset the restored catalog. RESETCAT will provide the necessary consistency between the restored catalog and its owned volumes.

- If a volume entry other than the catalog entry is included as a mismatched volume (that is, a data set directory or a data space group mismatch), you can recover all data sets on the mismatched volumes, using the EXPORTRA command.
- If there are mismatched data sets that are not on volumes that were mismatched, you can use the VERIFY command for those data sets that have only mismatched RBAs and EXPORTRA for those with more serious mismatches.
- For mismatched volumes that require the use of EXPORTRA, use DELETE with the FORCE option to clean up the volumes and then use a DEFINE SPACE on the volumes.
- Use IMPORTRA to reestablish the data sets recovered by means of the EXPORTRA command.

Note: If no backup copy of the catalog is available, you can do the following to restore your catalog:

1. Use EXPORT DISCONNECT to restore the catalog connector.
2. Define a recoverable catalog of the same name on a different volume. Do not include volumes owned by the original catalog as owned by the new catalog.
3. Issue the RESETCAT command for the new catalog, specifying all volumes owned by the original catalog, including the original catalog's volume.

To summarize, an unusable catalog can be reestablished, provided that certain backup procedures made possible by the system copy utility and the REPRO command are followed. The amount of work required to recover is based on the currency of the backup data. Factors that affect the currency of the backup data are activities such as altering the amount of space controlled by the catalog, defining and deleting data sets, and suballocating space to a VSAM data set.

Inaccessible Volume

A given volume may become wholly or partially unusable because of physical damage to the volume or because the catalog that owns the volume was restored to a state that is not synchronized with the volume. If the problem is caused by a catalog restore operation, the procedure outlined under "Unusable Catalog" on page 212 can be used to correct the condition. If the problem is caused by physical damage to the volume, recovery depends on the availability of backup copies of the catalogs, volumes, and data sets, and whether the catalog to which the volume belongs was defined with the RECOVERABLE attribute. If the catalog was recoverable, then a catalog recovery area (CRA) on the volume contains duplicate catalog information for each data set on it. Within this context, this section first discusses recovery of volumes without CRAs, then volumes with CRAs.

Volumes without CRAs

- If a dump of the volume is available and you require a reset of the entire volume, you can restore the damaged volume.
- If a dump of the volume is available and you require reset of non-VSAM data sets, and the VSAM data sets are accessible (but no reset is desired), you can do the following:
  - Recover the accessible VSAM data sets on the volume by using an EXPORT command.
  - Restore the volume.
  - Use a DELETE command with the FORCE option to clean up the volume and then use a DEFINE SPACE on the volume.
  - Reestablish the recovered data sets using the IMPORT command.
- If no dump of the volume is available, the volume is damaged only in the non-VSAM area, and VSAM data sets are accessible, you can do the following:
  - Recover the VSAM data sets on the volume by using an EXPORT command.
  - Initialize the volume and reestablish non-VSAM data sets.
  - Use DELETE with the FORCE option to remove the volume from the catalog and then use a DEFINE SPACE on the initialized volume.
  - Reestablish the recovered data sets using the IMPORT command.
  - If reestablished non-VSAM data sets were cataloged, delete and redefine the non-VSAM entries.
- If no dump of the volume is available, VSAM data sets are not accessible, and backup copies of the data sets on the volume exist, you can do the following:
  - Initialize the volume and restore non-VSAM data sets.
  - Use DELETE with the FORCE option to remove the volume from the catalog and then use a DEFINE SPACE on the volume.
  - If exported copies of VSAM data sets are available, use the IMPORT command to reestablish them.
  - If backup copies of the VSAM data sets are available (not, however, any data sets created by IMPORT), define the data sets using the DEFINE command, and then use the REPRO command to load the backup copies onto the volume.

Volumes with CRAs

- If a dump of the volume is available and you require a reset of the entire volume, you can do the following:
  - Restore the damaged volume.
  - Use LISTCRA with the COMPARE option to see if the volume entry is mismatched, or if there are data set mismatches.
  - If there are data set mismatches only, use the VERIFY command for those data sets with only mismatched RBAs and EXPORTRA for those with more serious mismatches.
  - If there is a volume mismatch other than a general mismatch, use RESETCAT to reset the catalog from the CRA(s) of the restored volume(s).
  - If there was a volume mismatch that required the use of EXPORTRA, use DELETE with the FORCE option to clean up the volume and then use a DEFINE SPACE on the volume. Keep in mind that a DELETE FORCE results in the loss of all VSAM data on that volume.
  - Use IMPORTRA to reestablish the data sets recovered by means of the EXPORTRA command.
- If a tape dump of the volume is available and reset of damaged data sets is desired, you can do the following:
  - Recover the accessible VSAM data sets on the volume by using the EXPORTRA command.
  - Restore the volume from tape.
  - Use an EXPORTRA command to recover the previously inaccessible VSAM data sets that were restored.
  - Use a DELETE command with the FORCE option to clean up the volume and then use a DEFINE SPACE on the volume.
  - Reestablish the recovered data sets using the IMPORTRA command.

VSAM Volume Recovery Function

The VSAM volume recovery function removes all VSAM data spaces and resets volume ownership for a volume that cannot be located with its catalog entry (a system failure or I/O error might have damaged the volume entry). The volume's Format-4 DSCB (in the VTOC) is reset to remove its ownership from the VSAM catalog. Non-VSAM data sets on the volume are not affected. The VSAM catalog that owns the volume is not accessed or modified—the damaged volume entry is unchanged.

The VSAM volume recovery function can be used to remove a damaged user catalog from a volume without first deleting each of the catalog's objects (VSAM objects and non-VSAM data sets described with the catalog's entries).

Use the recovery function only when you cannot access the catalog that owns the volume. Use the function carefully to prevent unwanted loss of data. This function can contribute to system integrity exposures when used improperly. (See "VSAM Volume Cleanup" on page 205 for further information about the volume recovery function.)

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

```
ALTER (entry nameffl/password")
FILE(dname)
REMOVEVOLUMES(volserffl volser...")
```

where:

entry name[/password]
: names the master catalog. If the master catalog is password protected, then its master password must also be supplied.

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

REMOVEVOLUMES(volserffl volser...")
: identifies volume(s) on which all VSAM data spaces are removed and VSAM ownership of the volume is relinquished. Volumes owned by the master catalog identified in entry name cannot be specified.

Recovering a VSAM Recoverable Catalog

When creating a VSAM user catalog or a VSAM master catalog, you can define it as recoverable. A recoverable catalog has an area containing a copy of some of the catalog entries on each volume it owns. This area is called the catalog recovery area or CRA. When a system or device failure damages the catalog or some of its entries, each volume's catalog recovery area contains the information needed to restore the damaged entries. The contents of a volume's catalog recovery area depend on the types of objects the volume contains. Figure 36 identifies a volume whose catalog recovery area contains a copy of the cataloged object.

Type of Entry	Volume Whose Catalog Recovery Area Contains a Copy of:
Volume entry	Its own volume
Key-sequenced cluster entry and its data and index entries	The first part of the cluster's index component
Alternate index entry and its data and index entries, when the	The first part of the alternate index's base cluster's index component
Path entry, when the path is related to a key-sequenced cluster	Thefirst part of the path's base cluster's index component
Entry-sequenced cluster's entry and its data entry	The first part of the cluster's data conponent
Alternate index entry and its data and index entries, when the alternate index is for an entry-sequenced cluster	The first part of the alternate index's base cluster's data component

Figure 36 (Part 1 of 2). Catalog Recovery Area Contents

Type of Entry	Volume Whose Catalog Recovery Area Contains a Copy of:
Path entry, when the path is retated to an entry-sequenced cluster	The first part of the path's base cluster's data component
Relative record cluster entry and its data entry	The first part of the cluster's data component
Non-VSAM data set	The non-VSAM entry's catalog
Generation data group	The entry's catalog
Alias entry	The catalog where the non-VSAM association for the alias is defined
User catalog connector entry in the master catalog	The master catalog

Figure 36 (Part 2 of 2). Catalog Recovery Area Contents

If the recoverable catalog is damaged so its entries are inaccessible, are down level, or contain erroneous information, choose one of these two methods to restore your catalog to a usable condition.

- The EXPORTRA/IMPORTRA Method: This method is used mainly to selectively repair specific catalog entries. Reorganization of your catalog and your data is a by-product of this approach because it involves the movement of data. The following procedure will restore the usability of your catalog.

  1. Issue the LISTCRA command to list and, optionally, compare the catalog recovery area entries. To use the LISTCRA command, access method services must be authorized. See "Authorized Program Facility (APF)" in *System Macros and Facilities* for information about program authorization.
  2. Issue the EXPORTRA command to obtain a copy of the damaged entries from the catalog recovery area or a copy of the data set's contents if a VSAM data set entry is moved.

     To use the EXPORTRA command, access method services must be authorized. See "Authorized Program Facility (APF)" in *System Macros and Facilities* for information about program authorization.
  3. Do one or more of the following to clear the damaged volume or reset it so it is usable:
     - Issue the DELETE SPACE command with the FORCE parameter to remove VSAM data spaces from the volume.
     - Issue the ALTER REMOVEVOLUMES command to remove the VSAM catalog's ownership of the volume.
  4. Issue the IMPORTRA command to reload the data copied and moved during step 2 above.

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

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

Listing the Catalog Recovery Area's Contents

A recoverable catalog maintains a copy of each catalog entry in a separate part of the volume, called the catalog recovery area (CRA). If your catalog was created with the RECOVERABLE option, you can determine the damage to your catalog when a system or hardware failure occurred. You can also determine if a volume or volumes owned by your catalog are out of synchronization with the catalog itself (for example, by restoring a volume from a down-level backup).

When you issue the LISTCRA command with the COMPARE parameter, access method services compares each entry in the CRA to its corresponding entry in the catalog. The comparison is made on a record-by-record, byte-by-byte basis. When a mismatch is encountered within a record, access method services determines the field in which the mismatch exists and prints a message. For certain fields, the message identifies the field(s) that mismatched.

When a mismatch is detected, access method services prints both the catalog record and the corresponding CRA record. Asterisks are printed below the specific area that shows a mismatch. All records associated with the mismatched record are also printed (for example, a volume record and its extensions). Use the LISTCRA output listing to determine which catalog entries are no longer accurate, and to help you code the ENTRIES subparameter of the EXPORTRA command.

The types of output listing that the LISTCRA command can produce, are:

- A list of the name and volser of each entry and each related entry in the catalog recovery area. The entries are listed in alphameric order by group type (NOCOMPARE and NAME).
- A full dump (hexadecimal and character listing) of each entry and its related entries in the catalog recovery area. The entries are listed in alphameric order by group type (NOCOMPARE and DUMP).
- A list of the name and volser of each catalog entry whose data does not compare equally with the entry's copy in the catalog recovery area, and an indication of the type of information that mismatches. The mismatched entries are listed in alphameric order by group type. The entries that compare equally are not listed (COMPARE and NAME).
- A full dump (hexadecimal and character listing) of each catalog record whose contents do not compare equally with the record's copy in the catalog recovery area. Asterisks are placed below each byte that mismatches. The mismatched entries are listed in alphameric order by group type. The entries that compare equally are not listed (COMPARE and DUMP).
- A full dump (hexadecimal and character listing) of all entries in the catalog recovery area in sequential order as they occur in the CRA (SEQUENTIALDUMP).

Copying a Catalog Entry from the Catalog Recovery Area

If you discover that your catalog is partially or completely damaged because of a system failure or hardware problem, reconstruct the damaged catalog entries so that you can access the cataloged object's data. If you discover that some of the entries in your catalog are not in synchronization with volumes the catalog owns, resynchronize the incorrect entries so they properly reflect the actual status of the volumes. You can rebuild a catalog entry by issuing the IMPORT command if you have recently made a copy of the object with the EXPORT command. The IMPORT command replaces the damaged catalog entry with its copy in the exported file.

An exported copy of your cluster might not exist, or might not be current. The volume that contains your data might not be damaged, but if its catalog entry is damaged or out-of-synchronization, you cannot use the catalog entry to locate and access your data. You must replace the damaged catalog entry with its undamaged copy in the catalog recovery area. If the entry is for a VSAM data set, move the contents of the data set to a volume owned by an undamaged catalog. For a recoverable catalog, use the EXPORTRA command to obtain the catalog entry's copy from the catalog recovery area, then use the IMPORTRA command to replace the damaged entry with its copy.

If an entire VSAM volume becomes unusable, and a backup copy of the volume exists, use RESETCAT rather than EXPORTRA to reset your catalog so that it will correctly access the VSAM data sets on the restored volume. (See "Resetting Catalog Entries (RESETCAT)" on page 221.)

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

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

Capabilities of the EXPORTRA command that are not available as functions of the EXPORT command are:

- Copying VSAM catalog entries and the contents of nonempty VSAM data sets to a movable storage device (a magnetic tape or demountable disk pack).
- Copying all entries in a catalog recovery area. This includes VSAM clusters and alternate index entries, VSAM user catalog connectors and their aliases (from a master catalog CRA only), GDG base entries, and non-VSAM entries and their aliases. The connection between a non-VSAM entry and its GDG base entry is described in the copy. Page spaces are not copied.
- Copying the entries for an empty cluster or alternate index.
- Obtaining a copy of each entry in the catalog recovery area by issuing the EXPORTRA command once. Subsequently, you can replace all entries exported with the EXPORTRA command by issuing the IMPORTRA command once.
- Obtaining the copy of one or more entries, as you specify, without obtaining the rest of the catalog recovery area.

- Obtaining the copy of each entry from one volume's catalog recovery area without obtaining entries from the catalog recovery areas of other volumes when the catalog owns more than one volume,

Use of the EXPORTRA command is limited to a single execution per system at one time. Multiple executions cannot be processed simultaneously.

The EXPORTRA command uses the variable-blocked spanned sequential format (SAM RECFM = VBS) for its output data set. Each record contains an 8-byte header. Each data record of a VSAM relative record data set contains an additional 4-byte header. EXPORTRA sets the maximum logical record length of the output data set based on the largest maximum record size of all the VSAM data sets being copied. When each of the data sets to be exported is defined, a value is specified via the maximum subparameter of the RECORDSIZE parameter of the DEFINE CLUSTER or ALTERNATEINDEX command. EXPORTRA uses this value to determine the largest maximum record size:

- If a VSAM relative record data set has the largest maximum record size, the resulting maximum logical record length of the output data set is the greater of 280 or the largest VSAM record size + 12.
- If a VSAM key-sequenced or entry-sequenced data set has the largest maximum record size, the resulting maximum logical record length of the output data set is the greater of 280 or the largest VSAM maximum record size + 8.

The resulting maximum logical record length of the output data set is limited to 32760 bytes (access method services does not support the DCB parameter LRECL=X). Therefore, the largest VSAM record that can be handled by EXPORTRA is:

- 32748 for relative record data sets
- 32752 for all other types of VSAM data sets

If any of the VSAM data sets to be exported are defined with a maximum record size greater than that shown above, the EXPORTRA command terminates with an error message.

For portable data sets, you may specify a block size other than the default of 2048 bytes with the DCB parameter of the DD statement.

If the EXPORTRA command is executed for a RACF-protected VSAM entity, the RACF indicator is exported on to the portable data set. However, profiles are not deleted, nor are they moved to the portable data set. If RACF-protected VSAM entities are imported using IMPORTRA, you can reuse the old profiles or establish new ones.

The EXPORTRA command requires a DD statement for each volume to be accessed. All volumes must be mounted. EXPORTRA does not dynamically allocate the required volumes.

Restoring the Catalog Entry Obtained Using the EXPORTRA Command

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

VSAM clusters, their associated data and index components, and any paths over them, alternate indexes, their associated data and index components, and any paths over them, non-VSAM data sets, and generation data group entries are automatically defined in the catalog selected by the user. If a VSAM cluster or alternate index is not empty at the time the EXPORTRA command is issued, its data records and VSAM catalog entries are copied to the portable data set. These data records are reloaded into the space occupied by the redefined object. User catalog connector entries (which can be exported only from the system's master catalog) are connected to the master catalog. Existing user catalog connector entries with the same entry name as the imported entry are disconnected rather than deleted. The imported user catalog connector entry is then reconnected. The aliases of user catalog and non-VSAM entries are also redefined by IMPORTRA.

IMPORTRA requires a DD statement (specified through the OUTFILE parameter) that identifies a data set name and the serial number of each volume that is to contain the imported VSAM clusters or alternate indexes. Use concatenated DD statements if the data sets are on different device types. The data set name you specify is used by access method services for internal processing during the execution of the IMPORTRA command. The sequence of processing steps is given below:

1. The cluster or alternate index catalog information is obtained from the portable data set (created by EXPORTRA).
2. The cluster or alternate index is defined in the catalog.
3. If an existing catalog entry with the same entry name is found, the existing entry is deleted. The cluster or alternate index is then redefined in the catalog.
4. The cluster or alternate index entry in the catalog is renamed using the data set name you provided in the DD statement identified in your OUTFILE parameter.
5. The cluster or alternate index is opened, loaded with its data records, and closed.
6. The cluster or alternate index entry is renamed to the original name contained in the portable data set.

If a system failure occurs after step 4 and before the successful completion of step 6, the data set may exist in the VSAM catalog under the name provided on the DD statement. To correct the situation, delete the cluster or alternate index, using the data set name you provided in the DD statement, prior to rerunning the IMPORTRA command.

Resetting Catalog Entries (RESETCAT)

When you define a VSAM 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. Use the RESETCAT command when a recoverable catalog or one or more of its owned volumes becomes inaccessible. You can restore the inaccessible volume(s) from a backup copy by executing the RESETCAT command. The CRAs contain enough infor-

mation to reset the catalog entries so that VSAM data sets owned by that catalog can again be accessed correctly.

Unlike the EXPORTRA/IMPORTRA command, the RESETCAT command is a one-step operation that allows you to recover a catalog without movement of data. The RESETCAT command does not check or process the data. It compares catalog entries with CRA entries and resets the catalog as necessary so that you can regain access to the data. You must ensure that the data is at the correct level for your use.

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

If a recoverable catalog becomes unusable, use the LISTCRA command to help analyze the problem. (See "Recovering a VSAM Recoverable Catalog" on page 216.) If you are unable to access your data, restore the catalog volume. Then run the RESETCAT command to synchronize the catalog with its owned volumes. If volumes have been added since the catalog backup was made, RESETCAT can build these entries in the catalog from the volume's CRA. If volumes have been deleted since the last backup, use the DELETE SPACE (FORCE) command to delete the volume's space entries in the catalog and the data sets that resided on those volumes now marked unusable in the catalog.

RESETCAT uses the entries in the catalog recovery area to synchronize the catalog with each volume identifier. The actions taken by RESETCAT to accomplish this synchronization are summarized below:

- Entries that exist in the catalog but not in the CRA are deleted from the catalog.
- Entries that exist in the CRA but not in the catalog are inserted into the catalog.
- If a duplicate name for a catalog is encountered, the entry to be added is renamed. If the data or index component of a unique cluster or alternate index is renamed, the corresponding Format-1 DSCB is also renamed. If a non-VSAM entry is renamed, the corresponding Format-1 DSCB is not renamed.
- The data space accounting in the volume entry is checked against the volume table of contents (VTOC).
  - If the VTOC contains a Format-1 DSCB for VSAM space but the CRA volume entry does not reflect this space, the Format-1 DSCB is scratched.
  - If the CRA shows space that is not reflected in a Format-1 DSCB, the space is deleted from the CRA volume entry and the VSAM data sets that were shown as being contained within the space are marked unusable.
  - If the extents in the CRA volume entry do not match those in the associated Format-1 DSCB, the CRA is adjusted.
- A space consistency check is performed to ensure that the space claimed by a VSAM data set has the correct extents. If a data or index component claims

space that is not allocated to VSAM, the component is marked unusable. If a data or index component claims space that is in conflict with another claim, the component in error is marked unusable, and all unclaimed space is returned for suballocation.

- An association check is performed to ensure completeness of the structures. For example, a cluster entry must be correctly associated with its data and index components. When a structure is found to be incorrect, it is deleted and its space is returned for suballocation. If a base cluster association is incorrect, the entire structure (base cluster, all alternate indexes, and paths) is deleted. If an index structure is incorrect, only the extra alternate index structure and its paths are deleted.

Whenever RESETCAT takes action as indicated above, you are informed with appropriate messages.

If a catalog becomes unusable and no backup copy is available, use RESETCAT to recover all catalog entries:

For a user catalog, remove the catalog connector entry from the master catalog via EXPORT DISCONNECT, and use the RECOVERABLE attribute to define a catalog with the same name on a different volume. The new catalog can be on a different device type. You may specify a different allocation from that of the original catalog. Volumes owned by the unusable catalog should *not* be included as owned by the new catalog. The DEFINE operation would flag this as an error condition.

With this new catalog, issue RESETCAT, specifying all volumes owned by the previous catalog (including the unusable catalog's resident volume) for reset. Because the new catalog name is the same as the old catalog name, all entries in the specified CRAs will be added to the new catalog (including volume entries). At the conclusion of RESETCAT processing, the old catalog will have been deleted and the space freed for suballocation.

Although the catalog is always updated during RESETCAT processing, the CRA can also be updated under certain circumstances. If some external event such as a power failure were to cause RESETCAT to fail, partial updates to the catalog and CRA(s) may have taken place. Therefore, the catalog and any CRA volumes being reset should be restored before RESETCAT is rerun. It is advisable to have backup volumes of your catalog and CRA(s) before using RESETCAT.

Prior to running RESETCAT, you can execute a LISTVTOC for the volumes to be used in the reset operation. (The VTOC may be changed by RESETCAT.) Also, you can execute a LISTCAT command to reset the catalog.

After RESETCAT processing has completed, examine the messages it has issued. These messages describe specific actions taken by RESETCAT, such as marking a data set unusable or deleting an incorrect VSAM structure. You may also execute a LISTCRA command with the COMPARE option to verify that no further mismatches exist.

Any further action depends on the messages issued by RESETCAT. If a data set has been marked unusable, the data that is accessible can be copied using the REPRO command or deleted using the DELETE command. If renaming has taken place, you can alter the name selected by RESETCAT to a more meaningful name, using the ALTER command with the NEWNAME option.

RESETCAT Requirements

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

- Access method services must be authorized. See "Authorized Program Facility (APF)" in *System Macros and Facilities* for information about program authorization.
- The catalog being reset must be capable of being opened. Any errors terminating OPEN must be resolved before using RESETCAT. The catalog must also have the RECOVERABLE attribute. It may or may not have valid entries.
- CRAs must be capable of being opened. Any errors terminating OPEN must be resolved before using RESETCAT. Entries not related to the CRA itself may be inaccessible.
- CRAs must have been created by a recoverable catalog with the same name as the catalog being reset.
- No VSAM data sets cataloged in the catalog being reset can be open.
- The catalog must be extendable if it becomes enlarged as a result of the reset operation.
- If the master catalog is password protected, the master password is required.
- The master catalog may not be reset while it is in use as a master catalog.
- Before issuing RESETCAT, compatible levels of volumes containing multivolume files should be restored.

WORKFILE Space Requirements

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

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

Considerations for Multivolume Data Sets

The contents of a volume's CRA depend on the types of objects the volume contains. It is important to know which CRA has the catalog information for a particular object.

The primary CRA contains all the catalog records necessary to describe an object. For an entry-sequenced data set on two volumes, the volume that contains the first part of the data set contains all the records that describe the data set, including its allocation on the second volume. The second volume, a secondary CRA, contains information that shows that the entry-sequenced data set is allocated on the second

volume. If an I/O error makes the second volume useless and a previous version of that volume is restored, the present catalog information may be erroneous. The catalog may reflect the data set's extent on the second volume, which no longer exists. Before issuing a RESETCAT command, restore compatible levels of volumes containing multivolume data sets. RESETCAT will then reset the catalog to reflect the restored level of *all* data sets on all reset volumes.

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

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

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

Although the above checks guarantee that the catalog physically describes a data set correctly, these checks cannot guarantee that the data in the data set is at a consistent level. For instance, if a multivolume keyed-sequential data set is defined with the data on one volume and the index on another, the same define-time is associated with both. If, over some time, several additions, deletions, and updates are made without causing an extension of the data set, RESETCAT is unable to distinguish between different combinations of volumes taken from this time period. Because 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 part of the data set on the reset volume resides on the same physical place as described in the current catalog. Only verification (no reset) occurs for these partial entries. RESETCAT cannot guarantee that the level of data in the data set is at a consistent level in different volumes.

JCL Requirements

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

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

1. To define the work file in a catalog other than the master catalog, specify that catalog first in the JOBCAT or STEPCAT DD statement concatenation.

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

   The above example specifies the work file catalog and reset catalog, respectively, in the JOBCAT or STEPCAT DD statement.

2. To define the work file in the master catalog, specify the master catalog by name via the WORKCAT parameter. If the master catalog is specified in the STEPCAT or JOBCAT DD statement, it must appear last in the concatenation sequence.

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

   The above example specifies the reset catalog and master catalog, respectively, in the JOBCAT or STEPCAT DD statement.

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

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

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

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

```
//ddCRA1  DD  UNIT=3330,VOL=SER=XYZ,DISP=SHR
//ddCRA2  DD  UNIT=3330,VOL=SER=ABC,DISP=SHR
```

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

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

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

Restarting Programs after a Failure

In general, the checkpoint/restart program for VSAM data sets is similar to that provided for other data sets.

Recording Checkpoint Information

To restart after a failure that terminated processing, first determine the status of processing programs at the time of the failure. A processing program defines a checkpoint by issuing a CHKPT macro instruction. The checkpoint program issues a VSAM temporary CLOSE macro to update the catalog. It then records information about VSAM data sets in a checkpoint data set. If a failure occurs, the latest checkpoint record can be used to reconstruct what was happening when the checkpoint was taken.

Restarting the Processing Program

Restart consists of processing the checkpoint record and giving control back to the processing program interrupted by the failure. Different types of restart are distinguished for VSAM, for:

- Entry-sequenced output data sets. An entry-sequenced output data set is restored by eliminating all the records that have been added at the end since the checkpoint.
- Input data sets or key-sequenced data sets. A data set that was open for input at the checkpoint or a key-sequenced data set is prepared for restart by restoring any statistical information (such as number of records inserted) to its checkpoint status.

Restrictions and Options for Restarting a Program

The VSAM DD parameter, AMP, has a subparameter for specifying checkpoint/restart options that handle two special situations in restarting a processing program:

- Modifications other than records added sequentially to the end of an entry-sequenced data set. The restart program cannot restore a data set to its checkpoint status if there have been internal modifications to it since the checkpoint, and the restart program will normally not attempt restart processing.
- Addition of records to the end of a data set by way of a job step other than the job step that issued the checkpoint. Any records added to the end of an entry-sequenced data set will normally be erased in restoring the data set to its checkpoint status.

The AMP options for checkpoint/restart are: to let restart take its normal action for either situation, to override either one or the other of the two actions, or to override both. If you override the check for internal modification, your processing program is restarted, even though the data set it was processing cannot be restored. If you override the erasure of data at the end of a data set and the catalog has been updated, your processing program is not restarted. Your processing program is restarted only if you also override the check for modification.

For more information about checkpoint/restart, see *Checkpoint/Restart*.

Diagnostic Aids

Exits to Your Error Analysis Routines

VSAM provides optional exits to routines you supply to handle error situations. If you provide the exit routines for analyzing errors, your processing program can investigate many errors and decide what to do in an orderly manner. Not only physical errors, but also logical errors that may arise out of unlikely combinations of events in a complex application, can be handled by exits.

VSAM Messages

The messages put out by VSAM for the operator and programmer are designed to help them understand both the nature of the problem and the exact steps to take to correct it. Other messages that originate with VSAM are the diagnostic messages that are made available to your physical error analysis routines and the open/close/end-of-volume messages that are printed in a special message area provided by your processing program.

Generalized Trace Facility (GTF)

GTF is an optional program of OS/VS that continually records, as they occur, events of selected classes that are necessary to trace a processing program. You must weigh the relative values of this diagnostic ability and the added processing time required. It is a debugging tool and a maintenance aid: It produces unformatted output. To format and print this output, use the edit function of the HMDPRDMP or AMDPRDMP service aid. For information about GTF or the edit function, see *Service Aids*.

VSAM Debug Switches

The VSAM debug catalog aid allows you to exercise certain options when VSAM catalog management requests terminate. The CVT (communications vector table) contains a field that, when set, allows you to run a VSAM program that contains an error and, when the error occurs, to issue a problem determination message and to save certain work areas that would otherwise be freed.

VSAM SNAP Dump Facility

The VSAM SNAP dump facility provides a dump of VSAM-owned control blocks in CSA (common service area). Control blocks that are built for GSR (global shared resources) data sets reside in CSA subpools.

Changing a User Catalog to a VSAM Master Catalog

At some point in an existing operating system, you may want to change some of the attributes of your VSAM master catalog. For example, you may want to convert the VSAM master catalog from nonrecoverable to recoverable, or take advantage of a new VSAM enhancement. This is an example of the conversion process. A description of the process precedes an actual example of each step. When you understand the example's objectives, you can modify it so that it is appropriate for your system.

1. Install on your system the level (or higher level) of VSAM and access method services that supports the features you desire.
2. Export (with the PERMANENT attribute) all VSAM data sets from the old master catalog. Ignore for the moment the storage index and page data sets in the old master catalog.
3. Mount a scratch pack to hold a temporary catalog, a storage index, and the page data sets.
4. Define a temporary master catalog on the scratch pack. You can specify either DEFINE MASTERCATALOG or DEFINE USERCATALOG. In either case, VSAM creates a user catalog that can then be referenced in step 6 and used for the IPL in step 7.
5. Catalog the non-VSAM data sets needed for an IPL on the temporary master catalog. Define a new storage index and page data sets on the scratch volume in the temporary catalog. The page data sets should have the same name as the original page data sets so that the IEASYS00 member in SYS1.PARMLIB need not be replaced.
6. Alter the SYSCATLG member of the SYS1.NUCLEUS data set so that it points to the temporary catalog.

7. IPL the system (cold start procedure). The system now is using the temporary catalog as the master catalog, and refers to the storage index data set and page data sets cataloged in the temporary catalog.
8. Use the ALTER REMOVEVOLUMES command to remove VSAM from the volumes containing the old master catalog, the old page data sets, and the old storage index, and to remove VSAM from any volumes owned by the old master catalog.
9. Define the new catalog in place of the original. This catalog will eventually become the VSAM master catalog. As in step 4, you can specify either DEFINE MASTERCATALOG or DEFINE USERCATALOG.
10. Redefine the storage index and page data sets in their original locations on the new master catalog. (If the new catalog has been defined with the recoverable attribute, allow room for the catalog recovery area.) Catalog all non-VSAM data sets needed for IPL in the new master catalog.
11. Replace the SYSCATLG member in SYS1.NUCLEUS (now cataloged in the new master catalog) so that the SYSCATLG member points to the new master catalog.
12. Re-IPL the system (a cold start). The system should now be using the new VSAM master catalog, the redefined storage index, and the newly defined page data sets.
13. Clean up the scratch pack by using the ALTER command with the REMOVEVOLUMES parameter.
14. Redefine all the remaining non-VSAM data sets into the new master catalog.
15. Connect (and define aliases for) OS CVOLs that were connected to the old master catalog.
16. Redefine the data spaces that were deleted from the old master catalog.
17. Import the previously exported VSAM data sets into the new master catalog.
18. Import (with the CONNECT parameter) and define aliases for user catalogs that were connected to the old master catalog.

Your new master catalog is now reestablished. In this example, the new master catalog is recoverable (that is, its volume includes a catalog recovery area and can be processed with the LISTCRA, EXPORTRA, IMPORTRA, and RESETCAT commands).

If your old master catalog is already recoverable, you can use the EXPORTRA and IMPORTRA commands rather than the EXPORT and IMPORT commands as used in this example. You can use EXPORTRA for all volumes owned by the master catalog, including the master catalog volume, in place of step 2 above. After redefining any VSAM data spaces (step 16), you can use IMPORTRA in place of steps 14, 15, 17, and 18.

```
//JOB1     JOB   ...
//*  STEP 1 - INSTALL VSAM ON YOUR SYSTEM
//STEP2    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//*        DD  statements that describe the output data sets
//SYSIN    DD   *
       EXPORT commands required to export the VSAM data sets and user
       catalog connector entries from the old master catalog
/*
//*  STEP 3 - MOUNT A SCRATCH PACK
//STEP4    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//CATPAK   DD    UNIT=3330,VOL=SER=CATPAK,DISP=OLD
//SYSIN    DD    *
  DEFINE  MASTERCATALOG -
          (NAME(TEMPCAT)  -
          FILE(CATPAK)  -
          VOLUME(CATPAK)  -
          CYLINDERS (12 1))
/*
//STEP5    EXEC  PGM=IDCAMS,COND=(0,LT)
//STEPCAT  DD    DSN=TEMPCAT,DISP=OLD
//SYSPRINT DD    SYSOUT=A
//CATPAK   DD    UNIT=3330,VOL=SER=CATPAK,DISP=OLD
//SYSIN    DD    *
         See "Using JCL and the Access Method Services to
         Define System Data Sets" in System Generation
         for a list of the VSAM and non-VSAM system data
         sets required.
/*
//STEP6A   EXEC  PGM=IEHPROGM,COND=(0,LT)
//SG2001   DD    DISP=OLD,VOL=SER=SG2001,UNIT=3330
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
  SCRATCH DSNAME=SYS1.NUCLEUS,VOL=3330=SG2001,MEMBER=SYSCATLG
/*
//STEP6B   EXEC  PGM=IEBGENER,COND=(0,LT)
//SYSIN    DD    DUMMY
//SYSPRINT DD    SYSOUT=A
//SYSUT2   DD    DSN=SYS1.NUCLEUS(SYSCATLG),DISP=(OLD,KEEP),
//              DCB=BLKSIZE=13030,VOL=SER=SG2001,UNIT=3330
//SYSUT1   DD    *
  CAPAK   (See Note)
/*
//*  STEP 7 - IPL THE SYSTEM
//
//JOB2     JOB   ...
//STEP8    EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//DD3330   DD    UNIT=3330,DISP=OLD,VOL=SER=(SG2001,SPOOL1)
//SYSIN    DD    *
  ALTER TEMPCAT -
        REMOVEVOLUMES  (SG2001 SPOOL1)  -
        FILE (DD3330)
/*
```

Figure 37 (Part 1 of 2). Sample Job Stream for a Recoverable VSAM Master Catalog

```
//STEP9    EXEC  PGM=IDCAMS,COND=(0,LT)
//SYSPRINT DD    SYSOUT=A
//SG2001   DD    UNIT=3330,VOL=SER=SG2001,DISP=OLD
//SYSIN    DD    *
  DEFINE MASTERCATALOG -
         (NAME(AMASTCAT)  -
          FILE(SG2001)  -
          VOLUME(SG2001)  -
          CYLINDERS(12 1)  -
          RECOVERABLE)
/*
//STEP10   EXEC  PGM=IDCAMS,COND=(0,LT)
//STEPCAT  DD    DSN=AMASTCAT,DISP=OLD
//SYSPRINT DD    SYSOUT=A
//SPOOL1   DD    UNIT=3330,VOL=SER=SPOOL1,DISP=OLD
//SG2001   DD    UNIT=3330,VOL=SER=SG2001,DISP=OLD
//SYSIN    DD    *
         See "Using JCL and the Access Method Services to Define
         System Data Sets" in System Generation for a list
         of the VSAM and non-VSAM system data sets required.
/*
//STEP11A  EXEC  PGM=IEHPROGM,COND=(0,LT)
//SG2001   DD    DISP=OLD,VOL=SER=SG2001,UNIT=3330
//SYSPRINT DD    SYSOUT=A
//SYSIN    DD    *
  SCRATCH  DSNAME=SYS1.NUCLEUS,VOL=3330=SG2001,MEMBER=SYSCATLG
/*
//STEP11B  EXEC  PGM=IEBGENER,COND=(0,LT)
//SYSIN    DD    DUMMY
//SYSPRINT DD    SYSOUT=A
//SYSUT2   DD    DSN=SYS1.NUCLEUS(SYSCATLG),DISP=(OLD,KEEP),
//              DCB=BLKSIZE=13030,VOL=SER=SG2001,UNIT=3330
//SYSUT1   DD    *
   SG2001  (See Note)
/*
//
//JOB3     JOB   ...
//* STEP 12 - REIPL THE SYSTEM
//STEP13   EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=A
//CATPAK   DD    UNIT=3330,VOL=SER=CATPAK,DISP=OLD
//SYSIN    DD    *
  ALTER AMASTCAT -
        FILE(CATPAK) -
        REMOVEVOLUMES(CATPAK)
/*
//
```

Figure 37 (Part 2 of 2). Sample Job Stream for a Recoverable VSAM Master Catalog

Note to Figure 37:

This data record has the volume serial number of the volume containing the alternate catalog in columns 1 to 6.

Appendix F. CVOL Processor

The CVOL processor supports OS CVOLs (control volumes—volumes that contain one or more indexes of the catalog) within the single master catalog environment of MVS or MVS/XA. This appendix describes (1) the purpose of and functions supported by the CVOL processor, (2) how to use the CVOL processor, and (3) restrictions and limitations of the CVOL processor.

Purpose of the CVOL Processor

In MVS or MVS/XA, there is one master catalog—either an integrated catalog facility master catalog or a VSAM master catalog. However, the CVOL processor lets you process CVOLs if the CVOLs are cataloged in the master catalog as non-VSAM data sets.

Using the CVOL processor, you can use your CVOLs on multiple processors running OS, OS/VS1, or any release of MVS/XA or MVS without converting back and forth between the types of catalog structures supported by each operating system.

Functions Performed

In general, the CVOL processor maps the request to an OS request and performs the OS request. If a catalog function is requested for a data set in a CVOL, the CVOL processor translates or maps the request, if required, into a request that can be handled. After the request has been mapped, it is performed by the CVOL catalog management routines.

Figure 38 shows the original request and the requests they are mapped into, and gives additional information in the form of notes. Figure 39 on page 235 shows how TSO commands are mapped to integrated catalog facility or VSAM catalog requests, which are, in turn, mapped to OS CVOL requests.

Original Request	Mapped Request	Notes
OS CATLG	OS CATBX	
OS CATBX	OS CATBX	
OS UNCATLG	OS UCATDX	
OS UCATDX	OS UCATDX	

Figure 38 (Part 1 of 3). OS and VSAM Functions Supported

Original Request	Mapped Request	Notes
OS LOCATE by name	OS LOCATE by name	The maximum number of volumes in a volume list returned is 20. The TTR pointer to the next catalog block in the returned catalog block is returned in the caller's work area. Therefore, you cannot do a LOCATE by TTR to obtain additional volume lists. In the original catalog block returned, you get a number of total volumes on which your data set resides.
OS RECATLG	OS RECATLG	
VSAM SUPERLOCATE	OS LOCATE by name followed by an OS LOCATE by TTR	The number of volumes returned is dependent on the size of the output area for the volume list. If an alias is provided in the original request, the real name of the data set is returned. The information returned includes the volume serial numbers, device type, file sequence number, TTR of the DSCB for a single volume data set, and the total number of volumes that the data set resides on. The output is in the VSAM format.
VSAM DELETE	OS UCATDX	If an alias named was provided in the original request, the alias name of the uncataloged data set is returned. All index levels, except the first level index that is no longer required, are deleted. An alias name can never be deleted. If the scratch bit is on, you must ensure that the volume is mounted for the scratch to be effective. The output is in the VSAM format.
VSAM LOCATE	OS LOCATE by name followed by an OS LOCATE by TTR	If an alias is provided in the original request, the real name of the data set is returned. The output is in the VSAM format.
VSAM LOCATE (LISTCAT)	OS LOCATE by name followed by an OS LOCATE by TTR	If an alias name is provided as the first-level index of the name in the original request, then the alias name of a the data set is returned. The output is in the VSAM format.
VSAM GENERIC LOCATE	OS LOCATE by name followed by an OS LOCATE by TTR	If an alias name is provided as the first-level index of the name in the original request, then the alias name of the data set is returned. The only types returned are non-VSAM types; alias or generation data group index types are never returned. The output is in the VSAM format.
OS BLDA	OSBLDA[1]	See Note, following.
OS BLDG	OS BLDG[1]	See Note, following.
OS BLDX	OS BLDX[1]	See Note, following.

Figure 38 (Part 2 of 3). OS and VSAM Functions Supported

Original Request	Mapped Request	Notes
OS DLTA	OS DLTA[1]	See Note, following.
OS DLTX	OS DLTX[1]	See Note, following.
OS LNKX	OSLNKX[1]	LNKX serves only to establish a connection between two CVOLs. The pointer to a CVOL from the master catalog must be created using access method services commands.
OS DRPX	OS DRPX[1]	DRPX serves only to break a connection between two CVOLs. The pointer to a CVOL from the master catalog must be deleted using access method services commands.

Figure 38 (Part 3 of 3). OS and VSAM Functions Supported

Note to Figure 38:

1 Simple (unqualified) data names will be supported for these operations against CVOLs, only if the CVOL volume serial is supplied in the CAMLST parameter list. If the CVOL volume serial is not supplied in the CAMLST parameter list, the data name must be defined as a CVOL connector name in the master catalog using access method services commands. See "Using the CVOL Processor" on page 236 for more information.

TSO Command	VSAM Request	OS Request
RENAME	None	UCATDX, CATLG
ALLOCATE	None	CATLG
DELETE	GENERIC LOCATE, VSAM LOCATE, DELETE	UCATDX
LISTDS	GENERIC LOCATE, VSAM LOCATE	LOCATE by name, LOCATE by TTR
LISTCAT(FG)	GENERIC LOCATE, VSAM LOCATE	LOCATE by name, LOCATE by TTR
LISTCAT(BG)	LISTCAT (GET NEXT), VSAM LOCATE	LOCATE by name, LOCATE by TTR
LISTALC	VSAM LOCATE	LOCATE by name, LOCATE by TTR
DEFINE	VSAM LOCATE, DEFINE	LOCATE by name, None for DEFINE

Figure 39 (Part 1 of 2). TSO Command Mapping

TSO Command	VSAM Request	OS Request
ALTER	VSAM LOCATE, GENERIC LOCATE, ALTER	LOCATE by name, LOCATE by TTR None for ALTER

Figure 39 (Part 2 of 2). TSO Command Mapping

Functions Not Performed

Figure 40 lists the catalog functions that are not supported by the CVOL processor. These functions are not supported either because comparable functions are supported or because the function would extend the support of OS CVOL management. Attempting these functions results in a return code of 48.

VSAM Functions
ALTER
DEFINE
LISTCAT (GET NEXT)

Figure 40. Catalog Functions Not Supported

For more information on limitations and restrictions, see "Restrictions and Limitations" on page 238.

Using the CVOL Processor

The CVOL processor is automatically part of an MVS or MVS/XA system. You do not have to do anything at system generation time in order to use it. There are, however, several steps that must be performed after the system is generated before you can use the processor, and there are several restrictions and limitations that you must consider. This chapter describes the necessary steps and the restrictions and limitations.

How CVOLs Are Accessed

Catalog management requests may be directed to a particular CVOL in one of the following ways:

1. By the first qualifier of a data set name being defined as a CVOL connector in the master catalog (see "CVOL Connector Names in the Master Catalog" on page 237).
2. By specifying a CVOL volume serial in the CAMLST parameter list. This will cause the request to be directed to that CVOL, and will bypass integrated catalog facility or VSAM catalog processing.
3. By being linked to from another CVOL. These two CVOLs must have been previously connected with a LNKX operation.

Note that access type 3 may follow original access type 1 or 2. Also, as many as 255 CVOLs may be linked together with one linking name.

CVOL Pointers in the Master Catalog

Every OS CVOL that is to be referenced in an MVS or MVS/XA system must be defined as a non-VSAM data set in the master catalog. Because all OS CVOLs have the name SYSCTLG, a naming convention has been established to permit multiple CVOL entries in the master catalog. This involves appending some unique qualifier to the name SYSCTLG, yielding a name of the form SYSCTLG.*Vyyyyyy*. Each such entry in the master catalog is considered a master catalog CVOL pointer.

If access type 2 or 3, as described above, is to be used with a particular CVOL, you must define a master catalog CVOL pointer with a name of the form SYSCTLG.V*yyyyyy*, where *yyyyyy* is the volume serial of the CVOL. For the greatest flexibility in accessing CVOLs, an installation should define SYSCTLG.V*volser* CVOL pointers for every CVOL.

The following example uses the DEFINE command of access method services to define a CVOL pointer in the master catalog. For more information on this command, refer to *Access Method Services Reference*.

Example

```
DEFINE NONVSAM(NAME(SYSCTLG.V111111)-
    VOL(111111)-
    DEVT(3330))-
       CATALOG(AMASTCAT/MASTERPW)
```

CVOL Connector Names in the Master Catalog

To use access type 1 described above, every high-level index name that you want to use as a connector name must be defined in the master catalog as an alias of a master catalog CVOL pointer.

This function is similar to connecting a CVOL to an OS master catalog in pre-MVS systems using LNKX (often performed with the CONNECT function of IEHPROGM). LNKX now serves only to connect two CVOLs. A CVOL must be connected to the master catalog, using the DEFINE ALIAS command of access method services.

In the following example, the high-level index name "PAYROLL" is established as a connector name for the CVOL pointed to by the CVOL pointer named SYSCTLG.V111111. This causes requests for data set names such as PAYROLL.MASTER or PAYROLL.RATE.TABLE to be routed to the specified CVOL. For example,

```
DEFINE  ALIAS(NAME(PAYROLL)-
        RELATE(SYSCTLG.V111111))
```

Resource Access Control Facility (RACF)

If an installation utilizes a catalog to access RACF-protected data sets, to preserve security, the catalog entry pointing to the data set should be protected against unauthorized changes. Otherwise, the entry could be changed to point to a bogus version of the data set.

To provide this protection, certain requests in a RACF-protected CVOL require RACF authorization in order to be honored. A RACF-protected CVOL means one that has been defined to RACF as a non-VSAM data set using the name SYSCTLG.V*xxxxxx*, where *xxxxxx* is the volume serial of the CVOL.

The catalog management requests that require authorization in a RACF-protected CVOL are:

- Uncataloging a data set
- Recataloging a data set
- Disconnecting one CVOL from another (DRPX)
- Cataloging a new generation data group (GDG) generation (because this may cause implicit uncataloging of older generations)

Figure 41 describes the level of RACF authorization required by RACF, along with the name and volume serial used to define the resource being checked. If the resource being checked has not been defined to RACF, the user is implicitly authorized to perform the operation.

Note that, in the GDG case, the name checked is the GDG index name. For example, if you are cataloging data set A.B.G0012V00, the index name is A.B. This name may be defined to RACF by allocating and protecting, via RACF, a dummy data set of this name on the CVOL volume. Note that the DCB attributes in the data set label for this data set can then also serve as a model for the entire generation data group.

CAUTION: There is no protection against deletion of a CVOL alias index. Installations requiring catalog alias names for RACF-protected data sets should catalog these data sets in integrated catalog facility or VSAM catalogs.

Operation	Auth Level	Resource Name	Volume Serial
UNCATLG	Alter	Name of data set	Volser of 1st volume of data set
RECATLG	Update	Name of data set	Volser of 1st volume of data set
DRPX	Alter	SYSCTLG.V*volser*	Volser of RELEASEd CVOL
CATLG GDG	Update	GDG index name	Volser of containing CVOL

Figure 41. RACF Authorization Checking

Restrictions and Limitations

There are several restrictions and limitations that you must consider when using the CVOL processor. The following topics discuss those restrictions and limitations.

Unqualified Data Set Names

In MVT and SVS, an installation could use JCL to locate unqualified data set names cataloged in an OS CVOL (by using the unqualified data set name in the system catalog as a CVOL pointer). This same method cannot be used in MVS or MVS/XA. This incompatibility occurs because the OS master catalog search algorithms for MVT and SVS are different from those used in MVS or MVS/XA for the integrated catalog facility or VSAM master catalog.

Generation Data Groups

A CVOL generation data group index cannot be defined and referenced via a CVOL alias entry. For example, if A.B.C.G0001V00 is an entry in the CVOL, that entry can be referenced. However, if X is an alias for A, X.B.C.G0001V00 can neither be defined nor referenced. The request will be terminated with a return code indicating that the data set was not found.

When a new generation data group is cataloged, and the oldest generation is to be scratched and uncataloged, the oldest generation will be uncataloged regardless of whether scratch is successful or not.

Password-Protected Data Sets

In a CVOL, there is no provision to prevent the update of cataloged entries for password-protected data sets. Update protection is provided by the RACF function discussed in Chapter 3, "Defining, Altering, and Deleting an Integrated Catalog Facility Catalog."

Application Program Considerations

There are also several restrictions and limitations that you must consider in relation to your application programs.

General Considerations: The data set name SYSCTLG is now a restricted name. It is the name required for all CVOLs.

CVOL VOLSERs are honored in CAMLSTs, and cause a catalog management request to be routed directly to the specified CVOL. Note that, in this case, a master catalog CVOL pointer of the form SYSCTLG.V*volser* must be defined as described earlier in this publication.

CVOLs that you want to access or allocate cannot be defined via a JOBCAT or a STEPCAT DD statement.

A CVOL (SYSCTLG data set) cannot be scratched while any job, or TSO session, that has opened that CVOL is still active.

The CAMLST and its associated fields must not be located in READ-ONLY storage.

Return Code Considerations: If the request is a VSAM request, register 15 contains a return code defined by integrated catalog facility or VSAM catalog management. These return codes are explained in *System Messages* under "IDC Access Method Services."

If the request is an OS request, register 15 contains one of the return codes described in Figure 42. Register 0 contains the integrated catalog facility or VSAM catalog

management return code if the OS request was satisfied in an integrated catalog facility or VSAM catalog and if register 15 does not contain a 0.

Figure 42 describes all return codes from the CVOL processor. For your convenience, return codes from CVOL catalog management are also included.

Type of Return Code	Decimal Value	Meaning
LOCATE	0	Successful.
	4	Either the required catalog does not exist, it cannot be opened, or there is a closed chain of CVOL pointers.
LOCATE	8	One of the following happened: • The entry was not found. Register 0 contains the number of valid index levels if in a CVOL. Register 0 contains the integrated catalog facility or VSAM catalog return code if in an integrated catalog facility or VSAM catalog. • The user is not authorized to perform this operation. Register 0 contains decimal 56. • A GDG alias was found. Register 0 contains the number of valid index levels.
	12	Either an index or an alias was found when the list of qualified names was exhausted. If an index pointer entry was found, the work area contains the first block of the specified index.
	16	A data set exists at other than the lowest index level specified. Register 0 contains the number of the index level where the data set was encountered.
	20	A syntax error exists in the name.

Figure 42 (Part 1 of 5). Return Codes and Their Meanings

Type of Return Code	Decimal Value	Meaning
	24	One of the following happened: • Permanent I/O error occurred. Register 0 contains the VSAM return code, or 0, if in a CVOL. • Unrecoverable error occurred. Register 0 contains the VSAM return code, or 0, if in a CVOL. • Nonzero ESTAE or GETMAIN return code. • Error in parameter list.
	28	Relative track address supplied to LOCATE routine is outside of the SYSCTLG data set extents.
LOCATE	32	Reserved.
INDEX	0	Successful.
	4	The CVOL does not exist or cannot be opened.
	8	One of the following happened: • The existing catalog structure is inconsistent with the operation requested. If the error was detected while processing in a CVOL, register 0 has the number of valid index levels and register 1 has the return code that would have resulted if a LOCATE macro had been issued on the same entry name. If the error was detected during master catalog processing, register 0 contains the integrated catalog facility VSAM catalog return code and register 1 contains 0. • The user is not authorized to perform the operation. Register 0 contains decimal 56; register 1 contains 0.
	12	An attempt was made to delete an index that has an alias, or has indexes or data sets cataloged under it. The index is unchanged.

Figure 42 (Part 2 of 5). Return Codes and Their Meanings

Type of Return Code	Decimal Value	Meaning
	16	The qualified name specified when building an index or generation index implies an index structure that does not exist; the high-level index, specified when connecting control volumes, does not exist.
	20	Space is not available on the specified control volume.
	24	Not used with the index macro instruction.
INDEX	28	A permanent I/O error was found when processing the catalog, or a nonzero return code from ESTAE or GETMAIN was encountered.
CATALOG	0	Successful.
	4	Either the required catalog does not exist, it is not open, or the "do not allocate" bit is on.
	8	One of the following happened: • The existing catalog structure is inconsistent with the operation requested. If the error was detected while processing in a CVOL, register 0 has the number of valid index levels and register 1 has the return code that would have resulted if a LOCATE macro had been issued for the same entry name. If the error was detected in an integrated catalog facility or VSAM catalog, register 0 contains the VSAM return code and register 1 contains 0. • The user is not authorized to perform the operation. Register 0 contains decimal 56, and register 1 contains 0.
	12	Not used with the CATALOG macro instruction.
	16	The index structure necessary to catalog the data set does not exist.
	20	There is insufficient space on the catalog data set.

Figure 42 (Part 3 of 5). Return Codes and Their Meanings

Type of Return Code	Decimal Value	Meaning
	24	An attempt was made to catalog an improperly named generation data set, or the generation index is full and the name data set is older than any currently in the index.
CATALOG	28	One of the following happened: • A permanent I/O or unrecoverable error was encountered. • An error was found in a parameter list. • An I/O error occurred in a CVOL. • There was a nonzero return code from ESTAE or GETMAIN.
SUPERLOCATE	0	Successful.
	4	Unable to allocate a CVOL.
	8	Either the data set was not found, there is an inconsistent structure of a CVOL, SYSCTLG does not exist on the specified volume, or the CVOL could not be opened.
	24	Either an I/O or an unrecoverable error was encountered.
	40	Insufficient space was supplied. The size of the calculated space is returned in the CTGFDBK area of the integrated catalog facility or VSAM catalog parameter list.
	68	No allocation occurred. The "do not allocate" bit is on.
	164	ESTAE or GETMAIN return code was nonzero
Other VSAM return codes when accessing CVOLs	0	Successful.
	4	Unable to allocate a CVOL.
	8	Either the data set was not found, there is an inconsistent structure of a CVOL, SYSCTLG does not exist on the specified volume, or the CVOL could not be opened.

Figure 42 (Part 4 of 5). Return Codes and Their Meanings

Type of Return Code	Decimal Value	Meaning
	24	There was an I/O or unrecoverable error on locate.
Other VSAM return codes when accessing CVOLs	28	There was an I/O or unrecoverable error on nonlocate.
	40	Insufficient space was supplied. The size of the calculated space is returned in the CTGFDBK area of the integrated catalog facility or VSAM catalog parameter list.
	44	Insufficient space was supplied. Size required not known.
	48	The function was invalid. It is not consistent with a CVOL.
	56	Security or password verification failure.
	84	DELETE failed because of unexpired purge date causing a SCRATCH failure.
	164	ESTAE or GETMAIN return code was nonzero.
	168	DELETE failed because of unmatched device type causing a SCRATCH failure.
	184	DELETE failed because of data set being currently open, causing a SCRATCH failure.

Figure 42 (Part 5 of 5). Return Codes and Their Meanings

Appendix G. Using Catalog Management Macro Instructions for OS CVOLs

Using catalog management macro instructions, you can do the following things:

- Retrieve information from an integrated catalog facility catalog, a VSAM catalog, or an OS CVOL
- Catalog non-VSAM data sets in an integrated catalog facility catalog, a VSAM catalog, or an OS CVOL
- Uncatalog non-VSAM data sets from an integrated catalog facility catalog, a VSAM catalog, or an OS CVOL
- Recatalog non-VSAM data sets in an integrated catalog facility catalog, a VSAM catalog, or an OS CVOL
- Read a block from an OS CVOL
- Build an index in an OS CVOL
- Build a generation index in an OS CVOL
- Delete an index from an OS CVOL
- Assign an alias to a high-level index in an OS CVOL
- Delete an index alias from an OS CVOL
- Connect two OS CVOLs
- Disconnect two OS CVOLs

Note: Access methods run in 24-bit addressing mode. Users running in 31-bit addressing mode must interface to the access methods using a user-written routine that is resident below 16 megabytes virtual. This is because the access methods will be able to return control only to a 24-bit addressable location. All addresses, buffers, parameters, control blocks, save areas, and exit addresses must be below 16 megabytes virtual. All access methods (except VSAM), for example, GET or PUT, must be called in 24-bit addressing mode.

Catalog Order of Search

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

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

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

   If no STEPCAT catalog is specified for the job step, and a user catalog is specified for the current job with a JOBCAT DD statement, the JOBCAT catalog(s) is searched. If more than one catalog is specified for the job, the catalogs are

searched in order of concatenation. If the entry is found, no other catalog is searched. Otherwise,

3. If the entry is identified with a qualified entryname and its first qualifier is the same as:
   - The name of a user catalog, or
   - The alias of a user catalog, or
   - The alias of an OS CVOL,

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

Return Code Considerations

The interpretation of catalog management return codes depends on whether the request is initiated using a CAMLST macro or a catalog parameter list (CPL), and whether the request is satisfied in an integrated catalog facility catalog, a VSAM catalog, or an OS CVOL.

If CAMLST is used and the request is satisfied in an OS CVOL, register 15 contains the OS CVOL return code and registers 0 and 1 may further describe the return code meaning. If CAMLST is used and the request is satisfied in an integrated catalog facility or a VSAM catalog, register 15 contains the OS CVOL return code, register 0 contains the VSAM return code, and register 1 is zero.

If a CPL is used and the request is satisfied in an OS CVOL, register 15 contains the VSAM return code, register 0 is not meaningful, and register 1 is nonzero. If a CPL is used and the request is satisfied in an integrated catalog facility or a VSAM catalog, register 15 contains the VSAM return code. The return code, reason code, and module identification can also be found in the CPL. These codes are explained in *System Messages* under message IDC3009I.

Note that, regardless of which parameter list is used, if the request is satisfied in an integrated catalog facility or a VSAM catalog, register 1 is zero, and, if the request is satisfied in an OS CVOL, register 1 contains X'08' in the high-order byte and may contain return information in the low-order byte.

Retrieving Information from a Catalog

To read an entry from a catalog, use the LOCATE and CAMLST macro instructions. You may specify the entry you want to read into your work area by using either (1) the fully or partially qualified name of a data set, or (2) the relative block address (TTR) of the block within an OS CVOL containing the entry. If you specify a fully qualified data set name, a list of volumes on which the data set resides will be read into your work area. This volume list always begins with a 2-byte entry that is the number of volumes in the list. If the data set resides on more than 20 volumes and is cataloged in an OS CVOL, the address of a volume control block will follow the volume list entries. (See Figure 47 on page 274 for an explanation of the control block.)

Note: There is a restriction when CAMLST is used to locate a data set that is over 20 volumes in length and on a VSAM catalog: Only the information from the first 20 volumes is returned.

If you specify a partially qualified data set name, the first block in the OS CVOL pointed to by the lowest-level index specified will be read into your work area. This is true if you specify two or more qualifiers, or if you specify the CVOL-RELEXP parameter in the CAMLST macro. Register 15 will contain return code 12. If you specify a single qualifier and do not include the CVOL-RELEXP parameter, the OS CVOL identifier 'SYSCTLG.Vyyyyyy' is read into your work area (the area previously occupied by the data set name). You may then insert 'yyyyyy' as the CVOL-RELEXP parameter in the CAMLST and reissue the LOCATE.

If you specify a relative track address (TTR), the block at that relative address in the CVOL catalog will be read into your work area.

You must add a step when specifying either an unqualified name or the highest level of a partially qualified name to retrieve information from an OS CVOL. You receive, instead, the volume information for the OS CVOL that is found in the master catalog. In addition, the single qualifier name that you specified is replaced by the SYSCTLG.Vyyyyyy name. You may then use that information to specify the OS CVOL volume serial number in CAMLST so that the search starts in the OS CVOL and gives you the information that you expected.

See Figure 43 on page 270 through Figure 50 on page 277 for descriptions of the contents of the volume control block and the other catalog data areas.

Retrieving Information by Data Set Name (LOCATE and CAMLST NAME)

When you specify a data set name, a volume list is built in your work area. A volume list consists of an entry for each volume on which part of the data set resides; it is preceded by a 2-byte field that contains a count of the number of volumes in the list. The count field is followed by a variable number of 12-byte entries. Each 12-byte entry consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. As many as 20 of these 12-byte entries can be built in your work area. (Device codes are presented in the UCBTYP data area description of *Debugging Handbook*.)

If the named data set is stored on only one volume, bytes 252 through 254 of your area may contain the relative track address of the DSCB for that data set; otherwise, these bytes are zero. Byte 255 contains zeros.

If the data set is cataloged in an OS CVOL and resides on more than five volumes, the volume list in your work area is really a volume control block (VCB) that has been read into your work area. In a VCB, the count field contains the number of volume entries in this VCB and any following VCBs. Thus a count of 41 indicates two following VCBs with counts of 21 and one, respectively. The relative track address (TTR) of the next VCB is in bytes 252 through 254 of your work area. The last VCB for a data set has binary zeros in bytes 252 through 254.

The macro format is:

[*symbol*] *listname-addrx*	**LOCATE** **CAMLST**	*listname* **NAME** *,dsname-relexp* *,[cvol-relexp]* *,area-relexp*

listname-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

NAME
this operand must be coded as shown, to retrieve information from a catalog by name.

dsname-relexp
specifies the virtual storage location of a fully qualified data set name. The area that contains the name must be 44 bytes long. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of the 6-byte volume serial number of the OS CVOL to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

area-relexp
specifies the virtual storage location of your 265-byte work area, which you must define. The work area must begin on a doubleword boundary.

Example: In the following example, the catalog entry containing a list of the volumes on which data set A.B resides is read into virtual storage.

```
         LOCATE    INDAB               READ CATALOG ENTRY
*                                      FOR DATA SET A.B
*                                      INTO VIRTUAL STORAGE
*                                      AREA NAMED LOCAREA.
*                                      LOCAREA MAY ALSO
*                                      CONTAIN A 3-BYTE
*                                      TTR AND THE 6-BYTE
*                                      OS CVOL SERIAL NUMBER
         Check Return Codes

INDAB    CAMLST    NAME,AB,,LOCAREA
AB       DC        CL44'A.B'
LOCAREA  DS        0D
         DS        265C
```

The LOCATE macro instruction points to the CAMLST macro instruction. NAME, the first operand of CAMLST, specifies that the system is to search for a catalog entry using the name of a data set. AB, the second operand, specifies the virtual storage location of a 44-byte area into which you have placed the fully qualified name of a data set. LOCAREA, the fourth operand, specifies a 265-byte area you have reserved in virtual storage.

After execution of these macro instructions, the 265-byte area contains a volume list or a volume control block for the data set A.B.

Return Codes from LOCATE

Control will be returned to your program at the next executable instruction after the LOCATE macro instruction. If the block has been successfully read from the catalog, register 15 will contain zeros. Otherwise, register 15 will contain one of the following return codes. The return codes are shown in decimal, with hexadecimal shown in parentheses.

Code	Meaning
4(04)	Either the required catalog does not exist or it cannot be opened or there is a closed chain of OS CVOL pointers.
8(08)	One of the following happened: • The entry was not found. If in an OS CVOL, register 0 contains the number of valid index levels. If in an integrated catalog facility or a VSAM catalog, register 0 contains the catalog return code. • The user is not authorized to perform this operation. Register 0 contains hexadecimal 38. • A generation data group (GDG) alias was found. Register 0 contains the number of valid index levels. The alias name was replaced by the true name.
12(0C)	One of the following happened: • An index or generation data group base entry was found when the list of qualified names was exhausted. Register 0 contains the number of valid index levels. The work area contains the first block of the specified index. • An alias entry was found. The alias name was replaced in the user parameter list by the true name. • An invalid low-level GDG name was found.
16(10)	A data set exists at other than the lowest index level specified. Register 0 contains the number of the index level where the data set was encountered.
20(14)	A syntax error exists in the name.
24(18)	One of the following happened: • Permanent I/O error occurred. Register 0 contains the VSAM or integrated catalog facility return code, or 0 if in an OS CVOL. • Nonzero ESTAE return code. • Error in parameter list.
28(1C)	Relative track address supplied to LOCATE routine is outside of the SYSCTLG data set extents.
32(20)	Reserved.

Note: See *System Messages*, Message IDC3009I, for documentation of integrated catalog facility catalog and VSAM catalog return codes.

Retrieving Information by Generation Data Set Name (LOCATE and CAMLST NAME)

You specify the name of a generation data set by using the fully qualified generation index name and the relative generation number of the data set. The value of a relative generation number reflects the position of a data set in a generation data group. The following values can be used:

- Zero—specifies the latest data set (highest generation number) cataloged in a generation data group.
- Negative number—specifies a data set cataloged before the latest data set.

  Note: When DISP (disposition) is DELETE to make room for other data sets and no generation data group exists, the job will complete indicating a deleted generation name (G0000V00). However, if a generation data group exists but is not in the range specified for deletion, then the step will fail.
- Positive number—specifies a data set not yet cataloged in the generation data group.

When you use zero or a negative number as the relative generation number, a volume list (or a volume control block) is placed in your work area, and the relative generation number is replaced by the absolute generation name.

When you use a positive number as the relative generation number, an absolute generation name is created and replaces the relative generation number. Zeros are read into the first 256 bytes of your work area, because there are no entries in the catalog.

The format is:

[*symbol*] *listname*	**LOCATE** **CAMLST**	*list-addrx* **NAME** *,dsname-relexp* *,[cvol-relexp]* *,area-relexp*

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

NAME
this operand must be coded as shown, in order to read a block from the catalog by generation data set name.

dsname-relexp
specifies the virtual storage location of the name of the generation index and the relative generation number. The area that contains these must be 44 bytes long. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of the 6-byte volume serial number of the OS CVOL to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

area-relexp
specifies the virtual storage location of your 265-byte work area, which you must define. The work area must begin on a doubleword boundary. The first 256 bytes of the work area will contain a volume list that is built from the catalog. If the data set resides on one volume, bytes 252 through 254 may contain the relative track address of the DSCB. This address is relative to the beginning of the volume.

Example: In the following example, the list of volumes that contain generation data set A.PAY(-3) is read into virtual storage.

```
          LOCATE    INDGX                 READ CATALOG ENTRY
*                                         FOR DATA SET A.PAY(-3)
*                                         INTO YOUR STORAGE
*                                         AREA NAMED LOCAREA
         Check Return Codes

INDGX     CAMLST    NAME,APAY,,LOCAREA
APAY      DC        CL44'A.PAY(-3)'
LOCAREA   DS        0D
          DS        265C
```

The LOCATE macro instruction points to the CAMLST macro instruction. NAME, the first operand of CAMLST, specifies that the system is to search the catalog for a catalog entry by using the name of a data set. APAY, the second operand, specifies the virtual storage location of a 44-byte area into which you have placed the name of the generation index and the relative generation number of a data set in the generation data group. LOCAREA, the fourth operand, specifies a 265-byte area you have reserved to receive the catalog information.

After execution of these macro instructions, the system will have replaced the relative generation number that you specified in your 44-byte area with the data set's absolute generation name. Control will be returned to your program at the next executable instruction after the LOCATE macro instruction. If the entry has been located and read successfully, register 15 will contain zeros. Otherwise, register 15 will contain a return code. For a description of the contents of the work area or the meaning of the exception return codes, see "Retrieving Information by Data Set Name (LOCATE and CAMLST NAME)" on page 247.

Retrieving Information by Alias (LOCATE and CAMLST NAME)

For each of the preceding functions, you can specify an alias as the name of a data set. Each function is performed exactly as previously described, with one exception: The alias name specified is replaced by the true name.

Note: Aliases are not allowed for generation data sets cataloged in OS CVOLs.

The format is:

[*symbol*] *listname*	**LOCATE** **CAMLST**	*list-addrx* **NAME** *,dsname-relexp* *,*[*cvol-relexp*] *,area-relexp*

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

NAME
this operand must be coded as shown, to retrieve information from a catalog.

dsname-relexp
specifies the virtual storage location of a fully qualified data set name, the first or only name of which is the alias. The area that contains the name must be 44 bytes long. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of the 6-byte volume serial number of the OS CVOL to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

area-relexp
specifies the virtual storage location of your 265-byte work area, which you must define. The work area must begin on a doubleword boundary. The first 256 bytes of the work area will contain a volume list that is read from a catalog. If the data set resides on one volume, bytes 252 through 254 may contain the relative track address of the DSCB. This address is relative to the beginning of the volume.

Example: In the following example, the catalog entry containing a list of the volumes on which data set A.B.C resides is read into virtual storage (data set A.B.C, however, is addressed by an alias name, X.B.C).

The LOCATE macro instruction points to the CAMLST macro instruction. NAME, the first operand of CAMLST, specifies that the system is to search the catalog for an entry using the name of a data set. ABC, the second operand, specifies the virtual storage location of a 44-byte area into which you have placed the fully qualified name of a data set (in this case, data set A.B.C is addressed by its alias X.B.C). LOCAREA, the fourth operand, specifies a 265-byte area you have reserved in virtual storage.

For information on return codes and the contents of your work area after execution, see "Retrieving Information by Data Set Name (LOCATE and CAMLST NAME)" on page 247.

```
         LOCATE    INDAB                READ CATALOG ENTRY
*                                       FOR DATA SET X.B.C
*                                       INTO VIRTUAL STORAGE
*                                       AREA NAMED LOCAREA.
```

Check Return Codes

```
INDAB    CAMLST    NAME,ABC,,LOCAREA
ABC      DC        CL44'X.B.C'
LOCAREA  DS        0D
         DS        265C
```

Reading a Block by Relative Block Address (LOCATE and CAMLST BLOCK)

You can read any block in an OS CVOL by specifying, in the form TTR, the identification of the block and its location relative to the beginning of the catalog. TT is the number of tracks from the beginning of the catalog; R is the record number of the desired block on the track.

The format is:

[*symbol*] *listname*	**LOCATE** **CAMLST**	*list-addrx* **BLOCK** *,ttr-relexp* *,cvol-relexp* *,area-relexp*

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

BLOCK
you must code this operand as shown.

ttr-relexp
specifies the virtual storage location of a 3-byte relative block address (TTR). This address indicates the position relative to the beginning of the catalog data set, of the track containing the block (TT), and the block identification (R) on that track.

cvol-relexp
specifies the virtual storage location of a 6-byte volume serial number for the volume to be processed.

area-relexp
specifies the virtual storage location of your 265-byte work area, which you must define. The work area must begin on a doubleword boundary. The first 256 bytes of the work area will contain the block that is read from the catalog, and the last 6 bytes will contain the serial number of the volume on which the block was found. If the data set resides on one volume, bytes 252 through 254 will contain the relative track address of the DSCB.

Example: In the following example, the block at the location indicated by TTR is read into virtual storage.

```
          LOCATE   BLK

         Check Return Codes

BLK       CAMLST   BLOCK,TTR,VOLSER,LOCAREA
*                                    READ A BLOCK INTO
*                                    VIRTUAL STORAGE AREA
TTR       DC       H'5'              RELATIVE TRACK 5
          DC       X'03'             BLOCK 3 ON TRACK
VOLSER    DC       C'111111'         VOLUME SERIAL OF OS CVOL
LOCAREA   DS       0D                NAMED LOCAREA
          DS       265C              LOCAREA ALSO CONTAINS
*                                    6-BYTE SERIAL NO.
```

The LOCATE macro instruction points to the CAMLST macro instruction. BLOCK, the first operand of CAMLST, specifies that the system is to search the catalog for the block indicated by TTR, the second operand. VOLSER, the third operand, specifies the virtual storage location of a 6-byte volume serial number for the volume to be processed. LOCAREA, the fourth operand, specifies a 265-byte area you have reserved in virtual storage.

After execution of these macro instructions, the 265-byte area contains the 256-byte block and the 6-byte serial number of the volume on which the block was found (in bytes 259 through 264).

Control will be returned to your program at the next executable instruction following the LOCATE macro instruction. If the index block at the address you specified has been successfully located and read into your work area, register 15 will contain zeros. Otherwise, register 15 will contain one of the exception return codes described under "Retrieving Information by Data Set Name (LOCATE and CAMLST NAME)" on page 247.

Building and Deleting Indexes

You handle OS CVOL indexes (build them, delete them, and so forth) by using combinations of the INDEX and CAMLST macro instructions.

Building an Index (INDEX and CAMLST BLDX)

To build a new OS CVOL index structure and add it to the catalog, you may create each level of the index separately. (You can also create index levels while you are cataloging a data set onto those index levels. To create each level of the index, use the INDEX and CAMLST macro instructions.)

These two macro instructions can also be used to add index levels to existing index structures.

The format is:

[*symbol*] *listname*	**INDEX** **CAMLST**	*list-addrx* **BLDX** *,namerelexp* [*,cvol-relexp*]

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

BLDX
this operand must be coded as shown.

namerelexp
specifies the virtual storage location of the fully qualified name of a data set or index level. The name cannot exceed 44 characters. If the name is less than 44 characters, it must be followed by at least one blank. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of a 6-byte volume serial number of the OS CVOL to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

Example: In the following example, index structure A.B.C is built on the OS CVOL whose serial number is 000045.

Each INDEX macro instruction points to an associated CAMLST macro instruction. BLDX, the first operand of CAMLST, specifies that an index level be built. The second operand specifies the virtual storage location of the area into which you have placed the fully qualified name of an index level. The third operand specifies the virtual storage location of the area into which you have placed the 6-byte serial number of the volume on which the index level is to be built.

```
          INDEX    INDEXA          BUILD INDEX A

         Check Return Codes

          INDEX    INDEXB          BUILD INDEX STRUCTURE
*                                  A.B

         Check Return Codes

          INDEX    INDEXC          BUILD INDEX STRUCTURE
*                                  A.B.C

         Check Return Codes

INDEXA    CAMLST   BLDX,ALEVEL,VOLNUM
INDEXB    CAMLST   BLDX,BLEVEL,VOLNUM
INDEXC    CAMLST   BLDX,CLEVEL,VOLNUM
VOLNUM    DC       CL6'000045'     VOLUME SERIAL NUMBER
ALEVEL    DC       CL2'A'          INDEX STRUCTURE NAMES
BLEVEL    DC       CL4'A.B'        FOLLOWED BY A BLANK
CLEVEL    DC       CL6'A.B.C'      WHICH DELIMITS FIELDS
```

Return Codes from INDEX

Control will be returned to your program at the next executable instruction following the INDEX macro instruction. If the index has been built successfully, register 15 will contain zeros. Otherwise, register 15 will contain one of the following exception return codes. The return codes are shown in decimal, with hexadecimal shown in parentheses.

Code	Meaning
4(04)	The OS CVOL does not exist or cannot be opened.
8(08)	One of the following happened: • The existing catalog structure is inconsistent with the operation requested. If the error was detected while processing in an OS CVOL, register 0 has the number of valid index levels and register 1 has the return code that would have resulted if a LOCATE macro had been issued on the same entry name. If the error was detected during the master catalog search process, register 0 contains the catalog return code and register 1 contains zero. • The user is not authorized to perform the operation. Register 0 contains 56 (decimal); register 1 contains 0.
12(0C)	An attempt was made to build an index or generation index that has an alias or has indexes or data sets cataloged under it. The index is unchanged.
16(10)	The qualified name specified when building an index or generation index implies an index structure that does not exist; the high-level index, specified when connecting control volumes, does not exist.
20(14)	Space is not available on the specified OS CVOL.
24(18)	Not used with the INDEX macro instruction.
28(1C)	A permanent I/O error was found when processing the catalog, or a nonzero return code from ESTAE was encountered.

Building a Generation Index (INDEX and CAMLST BLDG)

You build a generation index in an OS CVOL by using the INDEX and CAMLST macro instructions. All higher levels of the index must exist. If the higher levels of the index are not in the catalog, you must build them. How to build an index has been explained previously.

The format is:

[*symbol*] *listname*	**INDEX** **CAMLST**	*list-addrx* **BLDG** *,namerelexp* **,[***cvol-relexp***]** **,[DELETE]** **,[EMPTY]** *,number-absexp*

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

BLDG
this operand must be coded as shown.

namerelexp
specifies the virtual storage location of the fully qualified name of a data set or index level. The name cannot exceed 44 characters. If the name is less than 44 characters, it must be followed by at least one blank. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of a 6-byte volume serial number of the OS CVOL to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

DELETE
specifies that all data sets on direct access volumes that are removed from a generation data group are to be deleted, that is, the space allocated to the data set(s) is to be made available for reallocation. A SCRATCH macro instruction will be issued by the catalog management routines to delete the data set, which will be deleted from the volume if there are no conditions preventing deletion (for example, expiration date not passed, password not verified, volume not mounted, permanent I/O error encountered while trying to delete the data set).

EMPTY
specifies that references to all data sets in a generation data group cataloged in the generation index are to be removed from the index when the number of entries specified is exceeded.

number-absexp
specifies the number of data sets to be included in a generation data group. This number must be specified, and cannot exceed 255.

Example: In this example, generation index D is built on the OS CVOL, serial number 000045. The higher-level indexes A.B.C already exist. When the number of generation data sets in the generation index D exceeds four, the oldest data set is to be uncataloged. When the DELETE operand has been specified and the data set has been successfully uncataloged, the catalog management routines issue a SCRATCH macro to delete the data set. If there are no conditions preventing the data set from being deleted (for example, the expiration date was not passed, the password could not be verified, or a permanent I/O error was encountered when trying to delete the data set), the data set will be deleted.

```
          INDEX    GENINDX         BUILD GENERATION INDEX

          Check Return Codes

GENINDX   CAMLST   BLDG,DLEVEL,VOLNUM,,DELETE,,4
DLEVEL    DC       CL8'A.B.C.D '   ONE BLANK, DELIMITER
VOLNUM    DC       CL6'000045'
```

The INDEX macro instruction points to the CAMLST macro instruction. BLDG, the first operand of CAMLST, specifies that a generation index is to be built. DLEVEL, the second operand, specifies the virtual storage location of an area into which you have placed the fully qualified name of a generation index. VOLNUM, the third operand, specifies the virtual storage location of the area into which you have placed the 6-byte serial number of the volume on which the generation index is

to be built. DELETE, the fifth operand, specifies that all data sets dropped from the generation data group are to be deleted. The final operand, 4, specifies the number of data sets that are to be maintained in the generation data group. Control will be returned to your program at the next executable instruction following the INDEX macro instruction. If the generation index was built successfully, register 15 contains zeros. Otherwise, register 15 will contain one of the exception return codes described under "Building an Index (INDEX and CAMLST BLDX)" on page 254.

Deleting an Index (INDEX and CAMLST DLTX)

You can delete any number of index levels from an existing OS CVOL index structure. Each level of the index is deleted separately. Generation indexes are also removed this way. (You can also delete index levels automatically when you uncatalog a data set.) You delete each level of the index by using the INDEX and CAMLST macro instructions.

If an index level either has an alias, or has other index levels or data sets cataloged under it, it cannot be deleted.

The format is:

[*,symbol*] *listname*	**INDEX** **CAMLST**	*list-addrx* **DLTX** *,namerelexp* [*,cvol-relexp*]

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

DLTX
this operand must be coded as shown.

namerelexp
specifies the virtual storage location of the fully qualified name of a data set or index level. The name cannot exceed 44 characters. If the name is less than 44 characters, it must be followed by at least one blank. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of a 6-byte volume serial number of the OS CVOL to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

Example: In the following example, index level C is deleted from index structure A.B.C.

```
          INDEX    DELETE           DELETE INDEX LEVEL
*                                   C FROM INDEX STRUCTURE
*                                   A.B.C

          Check Return Codes

DELETE    CAMLST   DLTX,LEVELC
LEVELC    DC       CL6'A.B.C'       ONE BLANK FOR
*                                   DELIMITER
```

The INDEX macro instruction points to the CAMLST macro instruction. DLTX, the first operand of CAMLST, specifies that an index level is to be deleted. LEVELC, the second operand, specifies the virtual storage location of the area into which you have placed the fully qualified name of the index structure whose lowest level is to be deleted. Control will be returned to your program at the next executable instruction following the INDEX macro instruction. If the index level(s) was successfully deleted, register 15 contains zeros. Otherwise, register 15 contains one of the exception return codes described under "Building an Index (INDEX and CAMLST BLDX)" on page 254.

Assigning an Alias for an Index (INDEX and CAMLST BLDA)

For OS CVOLs, you assign an alias to an index level by using the INDEX and CAMLST macro instructions. An alias can be assigned only to a high-level index; for example, index A of index structure A.B.C can have an alias, but index B cannot. Assigning an alias to a high-level index effectively provides aliases for all data sets cataloged under that index. An alias cannot be assigned to a generation index.

The format is:

[*symbol*] *listname*	**INDEX** **CAMLST**	*list-addrx* **BLDA** *,index namerelexp* *,[cvol-relexp]* *,alias namerelexp*

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

BLDA
this operand must be coded as shown.

index namerelexp
specifies the virtual storage location of the name of a high-level index. The area that contains the name must be 8 bytes long. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of a 6-byte volume serial number of the OS CVOL catalog to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

alias namerelexp
specifies the virtual storage location of the name that is to be used as an alias for a high-level index. The area that contains the name must be 8 bytes long. The name may be defined by a C-type Define Constant (DC) instruction.

Example: In the following example, high-level index A is assigned an alias of X.

```
          INDEX     ALIAS             BUILD AN ALIAS FOR
*                                     A HIGH LEVEL INDEX

          Check Return Codes

ALIAS     CAMLST    BLDA,DSNAME,,DSALIAS
DSNAME    DC        CL8'A'            MUST BE 8-BYTE FIELDS
DSALIAS   DC        CL8'X'
```

The INDEX macro instruction points to the CAMLST macro instruction. BLDA, the first operand of CAMLST, specifies that an alias is to be built. DSNAME, the second operand, specifies the virtual storage location of an 8-byte area into which you have placed the name of the high-level index to be assigned an alias. DSALIAS, the fourth operand, specifies the virtual storage location of an 8-byte area into which you have placed the alias to be assigned.

Control will be returned to your program at the next executable instruction following the INDEX macro instruction. If the alias has been successfully assigned, register 15 will contain zeros. Otherwise, register 15 will contain one of the exception return codes described under "Building an Index (INDEX and CAMLST BLDX)" on page 254.

Deleting an Alias for an Index (INDEX and CAMLST DLTA)

For OS CVOLs, you can delete an alias previously assigned to a high-level index by using the INDEX and CAMLST macro instructions.

The format is:

[*symbol*] *listname*	**INDEX** **CAMLST**	*list-addrx* **DLTA** *,alias namerelexp* [*,cvol-relexp*]

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

DLTA
this operand must be coded as shown.

alias namerelexp
specifies the virtual storage location of the name that is used as an alias for a high-level index. The area that contains the name must be 8 bytes long. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of a 6-byte volume serial number of the OS CVOL catalog to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

In the following example, alias X, previously assigned as an alias for index level A, is deleted.

```
          INDEX    DELALIAS           DELETE AN ALIAS FOR
*                                     A HIGH LEVEL INDEX

          Check Return Codes

DELALIAS  CAMLST   DLTA,ALIAS
ALIAS     DC       CL8'X'             MUST BE 8-BYTE FIELD
```

The INDEX macro instruction points to the CAMLST macro instruction. DLTA, the first operand of CAMLST, specifies that an alias is to be deleted. ALIAS, the second operand, specifies the virtual storage location of the 8-byte area into which you have placed the alias to be deleted.

Connecting and Disconnecting OS CVOLs

You connect and disconnect OS CVOLs by using combinations of the INDEX and CAMLST macro instructions.

Connecting OS CVOLs (INDEX and CAMLST LNKX)

You connect two OS CVOLs by using the INDEX and CAMLST macro instructions.

You must supply the serial number of the volume to be connected and the high-level index name that will be used to associate the two volumes. If the index name is an alias of an OS CVOL pointer entry in the master catalog, then the serial number of the "from" volume may be omitted. Otherwise, you must supply the serial numbers of both volumes and the name of a high-level index associated with the volume to be connected.

The result of connecting OS CVOLs is that the volume serial number of the OS CVOL connected and the name of a high-level index are entered into the volume index of the volume to which it was connected. This entry is called a control-volume pointer.

The format is:

[*symbol*] *listname*	**INDEX** **CAMLST**	*list-addrx* **LNKX** *,index namerelexp* *,[cvol-relexp]* *,new cvol-relexp*

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

LNKX
this operand must be coded as shown.

index namerelexp
specifies the virtual storage location of the name of a high-level index. The area that contains the name must be 8 bytes long. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of a 6-byte volume serial number of the OS CVOL catalog to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

new cvol-relexp
specifies the virtual storage location of the 4-byte device code and 6-byte volume serial number of the control volume that is to be connected to another OS CVOL.

Example: In the following example, the OS CVOL whose serial number is 001555 is connected to the OS CVOL numbered 000155. The name of the high-level index is HIGHINDX.

```
          INDEX    CONNECT          CONNECT TWO OS CVOLS
*                                   WHOSE SERIAL NUMBERS ARE
*                                   000155 and 001555.
*                                   3330 DISK DEVICE CODE
          Check Return Codes

CONNECT   CAMLST   LNKX,INDXNAME,OLDCVOL,NEWCVOL
*
INDXNAME  DC       CL8'HIGHINDX'
OLDCVOL   DC       CL6'000155'
NEWCVOL   DC       X'30C0200D'
          DC       CL6'001555'
```

The INDEX macro instruction points to the CAMLST macro instruction. LNKX, the first operand of CAMLST, specifies that control volumes are to be connected. INDXNAME, the second operand, specifies the virtual storage location of the 8-byte area into which you have placed the name of the high-level index of the volume to be connected. OLDCVOL, the third operand, specifies the virtual storage location of a 6-byte area into which you have placed the serial number of the volume to which you are connecting.

NEWCVOL, the fourth operand, specifies the virtual storage location of a 10-byte area into which you have placed the 4-byte hexadecimal device code of the volume to be connected followed by the 6-byte area to contain the volume serial number of the volume to be connected.

Control will be returned to your program at the next executable instruction following the INDEX macro instruction. If the OS CVOLs have been successfully connected, register 15 will contain zeros. Otherwise, register 15 will contain one of the exception return codes described under "Building an Index (INDEX and CAMLST BLDX)" on page 254.

Disconnecting OS CVOLs (INDEX and CAMLST DRPX)

You disconnect two OS CVOLs by using the INDEX and CAMLST macro instructions.

The result of disconnecting OS CVOLs is that the OS CVOL pointer is removed from the volume index of the volume from which you are disconnecting.

The format is:

[*symbol*] *listname*	**INDEX** **CAMLST**	*list-addrx* **DRPX** *,index namerelexp* [*,cvol-relexp*]

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

DRPX
this operand must be coded as shown.

index namerelexp
specifies the virtual storage location of the name of a high-level index. The area that contains the name must be 8 bytes long. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of a 6-byte volume serial number of the OS CVOL catalog to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

Example: In the following example, the OS CVOL that contains the high-level index HIGHINDX is disconnected from the OS CVOL pointed to by the entry 'HIGHINDX' in the master catalog.

```
          INDEX    DISCNECT              DISCONNECT TWO
*                                        OS CVOLS

          Check Return Codes

DISCNECT  CAMLST   DRPX,INDXNAME
INDXNAME  DC       CL8'HIGHINDX'         MUST BE 8-BYTE FIELD
```

The INDEX macro instruction points to the CAMLST macro instruction. DRPX, the first operand of CAMLST, specifies that OS CVOLs are to be disconnected. INDXNAME, the second operand, specifies the virtual storage location of the 8-byte area into which you have placed the name of the high-level index of the OS CVOL to be disconnected.

Control will be returned to your program at the next executable instruction following the INDEX macro instruction. If the OS CVOLs were successfully disconnected, register 15 will contain zeros. Otherwise, register 15 will contain one of the exception return codes described under "Building an Index (INDEX and CAMLST BLDX)" on page 254.

Working with Non-VSAM Data Set Catalog Entries

You can catalog, uncatalog, and recatalog non-VSAM data sets in OS CVOLs, integrated catalog facility catalogs, and VSAM catalogs by using combinations of the CATALOG and CAMLST macro instructions. CATALOG macro instructions are used to point to CAMLST macro instructions; CAMLST macro instructions are used to specify cataloging options.

To catalog non-VSAM data sets in integrated catalog facility or VSAM catalogs, the search algorithm is the same as that given under "Order of Catalog Selection for DEFINE" in *Access Method Services Reference*. To uncatalog or recatalog non-VSAM data sets in integrated catalog facility or VSAM catalogs, the search algorithm is the same as that given under "Order of Catalog Search for DELETE" in *Access Method Services Reference*.

Cataloging a Non-VSAM Data Set (CATALOG and CAMLST CAT)

The format of the CATALOG and CAMLIST macros is:

[*symbol*] *listname*	**CATALOG** **CAMLST**	*list-addrx* **CAT[BX]** *,name-relexp* *,[cvol-relexp]* *,vol list-relexp* **[,DSCBTTR** = *dscb ttr-relexp***]**

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

CAT|BX&|
this operand must be coded as shown. Either CAT or CATBX may be coded; but, in either case, missing indexes within an OS CVOL are always automatically created.

name-relexp
specifies the virtual storage location of the fully qualified name of a data set. The name cannot exceed 44 characters. If the name is less than 44 characters, it must be followed by at least one blank. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of the 6-byte volume serial number of the OS CVOL catalog to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Building an Index (INDEX and CAMLST BLDX)" on page 254.

vol list-relexp
specifies the virtual storage location of an area that contains a volume list. The list must begin on a halfword boundary and consist of an entry for each volume on which the data set is stored. The first two bytes of the list indicate the number of entries in the volume list; the number cannot be zero. Each 12-byte volume list entry consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. The sequence number is always zero for direct access volumes. (Device codes are presented in *Debugging Handbook*.)

DSCBTTR = *dscb ttr-relexp*
specifies the virtual storage location of the 3-byte relative track address (TTR) of the format-1 data set control block (DSCB) for a data set that resides on only one volume. The address is relative to the beginning of the volume.

Programming Considerations for Multiple-Step Jobs

When you are executing multiple-step jobs, it is preferable to catalog or uncatalog data sets using JCL, instead of using IEHPROGM or a user program. Because ALLOCATION/UNALLOCATION monitors data sets during job execution and is not aware of the functions performed by the user programs, conflicting functions can be performed or GDG orientation can be lost.

UNALLOCATION recatalogs existing cataloged data sets at job termination. This action occurs because the data set is opened sometime during the job and the DSCB TTR was not found in the catalog entry. Therefore, if you are using the CAMLST macro to uncatalog and then catalog data sets with new volume information, be sure to include the DSCB TTR.

Example: In the following example, the non-VSAM data set named A.B.C is cataloged. The data set is stored on two volumes.

```
         CATALOG ADDABC            CATALOG DATA SET A.B.C.

         Check Return Codes

ADDABC   CAMLST    CAT,DSNAME,,VOLUMES
DSNAME   DC        CL6'A.B.C'      ONE BLANK FOR DELIMITER
VOLUMES  DC        H'2'            DATA SET ON TWO VOLUMES
         DC        X'30C0200D'     3330 DISK DEVICE CODE
         DC        CL6'000014'     VOLUME SERIAL NUMBER
         DC        H'0'            DATA SET SEQUENCE NUMBER
         DC        X'30C0200D'     3330 DISK DEVICE CODE
         DC        CL6'000015'     VOLUME SERIAL NUMBER
         DC        H'0'            SEQUENCE NUMBER
```

The CATALOG macro instruction points to the CAMLST macro instruction. CAT, the first operand of CAMLST, specifies that a data set is to be cataloged. DSNAME, the second operand, specifies the virtual storage location of the area in which the data set name A.B.C was placed. VOLUMES, the fourth operand, specifies the virtual storage location of the volume list that was built.

Return Codes from CATALOG

Control will be returned at the instruction following the CATALOG macro instruction. If A.B.C was successfully cataloged, register 15 will contain zeros. Otherwise, register 15 will contain one of the following return codes. The return codes are shown in decimal, with hexadecimal in parentheses.

Code	Meaning
4(04)	Either the required catalog does not exist, it is not open, or the "do not allocate" bit is on.
8(08)	One of the following happened: • The existing catalog structure is inconsistent with the operation requested. If the error was detected while processing in an OS CVOL, register 0 has the number of valid index levels and register 1 has the return code that would have resulted if a LOCATE macro had been issued for the same entry name. If the error was detected in an integrated catalog facility or a VSAM catalog, register 0 contains the catalog return code and register 1 contains zero. • The user is not authorized to perform the operation. Register 0 contains decimal 56 ('36') and register 1 contains zero.
12(0C)	Not used with the CATALOG macro instruction.
16(10)	The index structure necessary to catalog the data set does not exist.
20(14)	There is insufficient space on the catalog data set.
24(18)	An attempt was made to catalog an improperly named generation data set, or the generation index is full and the named data set is older than any currently in the index.
28(1C)	One of the following happened: • A permanent I/O or unrecoverable error was encountered. • An error was found in a parameter list.

- An I/O error occurred in an OS CVOL.
- There was a nonzero return code from ESTAE.

Uncataloging a Non-VSAM Data Set (CATALOG and CAMLST UNCAT)

When the UNCAT or UCATDX operand of the CAMLST macro instruction is used, a data set reference and unneeded indexes, with the exception of the highest-level index, are removed.

The format of the CATALOG and CAMLST macros is:

[*symbol*] *listname*	**CATALOG** **CAMLST**	*list-addrx* **UNCAT** *or* **UCATDX** *,name-relexp* [*,cvol-relexp*]

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

UNCAT *or* **UCATDX**
this operand must be coded as shown. Either UNCAT or UCATDX may be coded but, in either case, unneeded indexes, with the exception of the highest-level index, are always removed along with the data set reference.

name-relexp
specifies the virtual storage location of the fully qualified name of a data set or index level. The name cannot exceed 44 characters. If the name is less than 44 characters, it must be followed by at least one blank. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of the 6-byte volume serial number of the OS CVOL catalog to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

In the following example, the catalog entry for data set A.B.C is removed from a catalog. In an OS CVOL, index B is removed unless it contains references to other data sets. Index A remains because it is the highest-level index.

```
         CATALOG REMOVE                REMOVE REFERENCES TO
*                                      DATA SET A.B.C FROM
*                                      CATALOG

        Check Return Codes

REMOVE   CAMLST     UNCAT,DSNAME
DSNAME   DC         CL6'A.B.C'        ONE BLANK FOR DELIMITER
```

The CATALOG macro instruction points to the CAMLST macro instruction. UNCAT, the first operand of CAMLST, specifies that references to a data set are to be removed from the catalog. DSNAME, the second operand, specifies the virtual

storage location of an area into which you have placed the fully qualified name of the data set whose references are to be removed.

Control will be returned to your program at the instruction following the CATALOG macro instruction. If your data set has been successfully uncataloged, register 15 will contain zeros. Otherwise, register 15 will contain one of the return codes described under "Cataloging a Non-VSAM Data Set (CATALOG and CAMLST CAT)" on page 264.

Recataloging a Non-VSAM Data Set (CATALOG and CAMLST RECAT)

You can recatalog a cataloged non-VSAM data set by using the CATALOG and CAMLST macro instructions. Recataloging is usually necessary if a data set is extended to a new volume.

As in the original cataloging procedure, you must build a complete volume list in virtual storage. This volume list consists of an entry for each volume on which the data set resides. The first 2 bytes of the list indicate the number of entries in the list; the number may not be zero. Each 12-byte volume pointer consists of a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. The sequence number is always zero for direct access volumes. (Device codes are presented in *Debugging Handbook*.)

The format of the CATALOG and CAMLST macros is:

[*symbol*] *listname*	**CATALOG** **CAMLST**	*list-addrx* **RECAT** *,name-relexp* [,*cvol-relexp*] *,vol list-relexp* [,**DSCBTTR** = *dscb ttr-relexp*]

list-addrx
points to the parameter list (labeled listname) set up by the CAMLST macro instruction.

RECAT
this operand must be coded as shown.

name-relexp
specifies the virtual storage location of the fully qualified name of a data set. The name cannot exceed 44 characters. If the name is less then 44 characters, it must be followed by at least one blank. The name may be defined by a C-type Define Constant (DC) instruction.

cvol-relexp
specifies the virtual storage location of the 6-byte volume serial number of the OS CVOL catalog to which this catalog request is directed. For a discussion of the effect of specifying or omitting this operand, see "Catalog Order of Search" on page 245.

vol list-relexp
specifies the virtual storage location of an area that contains a volume list. The area must begin on a halfword boundary.

DSCBTTR = *dscb ttr-relexp*
specifies the virtual storage location of the 3-byte relative track address (TTR) of the identifier (format-1) DSCB for a data set that resides on only one volume. The address is relative to the beginning of the volume.

Example: In the following example, the two-volume data set named A.B.C is recataloged to add a third volume. An entry is added to the volume list, which previously contained only two entries.

```
          CATALOG RECATLG            RECATALOG DATA SET
*                                    A.B.C ADDING A NEW
*                                    VOLUME

          Check Return Codes

RECATLG   CAMLST    RECAT,DSNAME,,VOLUMES
DSNAME    DC        CL6'A.B.C '      FOR DELIMITER ONE BLANK
VOLUMES   DC        H'3'             THREE VOLUMES
          DC        X'30C0200D'      3330 DISK DEVICE CODE
          DC        CL6'000014'      VOLUME SERIAL NUMBER
          DC        H'0'             SEQUENCE NUMBER
          DC        X'30C0200D'      3330 DISK DEVICE CODE
          DC        CL6'000015'      VOLUME SERIAL NUMBER
          DC        H'0'             SEQUENCE NUMBER
          DC        X'30C0200D'      3330 DISK DEVICE CODE
          DC        CL6'000016'      VOLUME SERIAL NUMBER
          DC        H'0'             SEQUENCE NUMBER
```

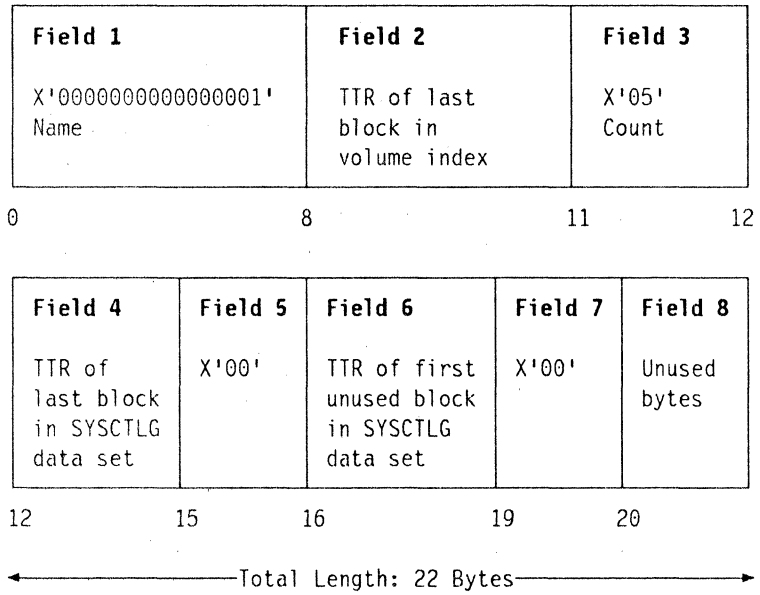
The CATALOG macro instruction points to the CAMLST macro instruction. RECAT, the first operand of CAMLST, specifies that a data set is to be recataloged. DSNAME, the second operand, specifies the virtual storage location of an area into which you have placed the fully qualified name of the data set to be recataloged. VOLUMES, the fourth operand, specifies the virtual storage location of the volume list you have built.

Control will be returned to your program at the instruction following the CATALOG macro instruction. If the data set has been successfully recataloged, register 15 will contain zeros. Otherwise, register 15 will contain one of the return codes described under "Cataloging a Non-VSAM Data Set (CATALOG and CAMLST CAT)" on page 264.

OS CVOL Entry Formats

This section describes the format and contents of each of the entries that may appear in the OS CVOL.

OS CVOL Volume Index Control Entry



Field 1: Name (8 bytes)—contains only a hexadecimal 1 to ensure that this entry is the first entry in the first block of the index.

Field 2: Last-block address (3 bytes)—contains the relative track address (TTR) of the last block in the volume index.

Field 3: Halfword count (1 byte)—contains a hexadecimal 5 to indicate that 5 halfwords follow.

Field 4: Catalog upper limit (3 bytes)—contains the relative track address (TTR) of the last block in the catalog data set.

Field 5: Zero field (1 byte)—contains binary zeros.

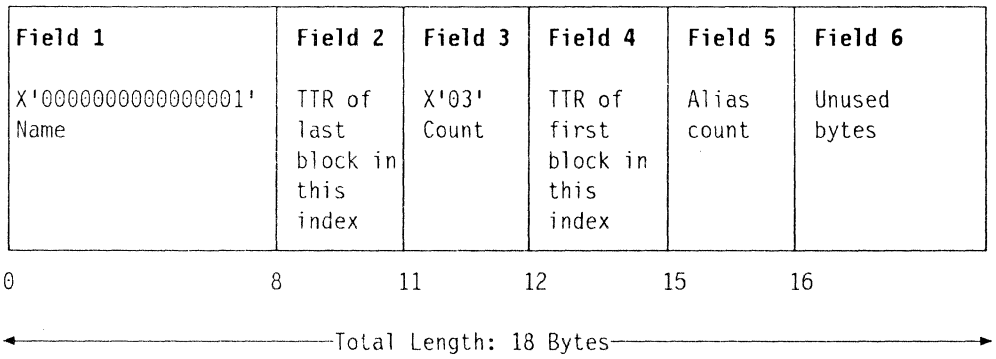
Field 6: First-available-block address (3 bytes)—contains the relative track address (TTR) of the unused block in the catalog that is closest to the beginning of the catalog data set.

Field 7: Zero field (1 byte)—contains binary zeros.

Field 8: Unused (2 bytes)

Figure 43. OS CVOL Volume Index Control Entry

OS CVOL Index Control Entry



This index control entry is similar to a volume index control entry, but it only contains information about the index, which it begins. It is 18 bytes long and contains six fields.

Field 1: Name (8 bytes)—contains only a hexadecimal 1 to ensure that this entry, because it has the lowest binary name value, is the first entry in the first block of the index.

Field 2: Last block address (3 bytes)—contains the relative track address (TTR) of the last block assigned to this index.

Field 3: Halfword count (1 byte)—contains a hexadecimal 3 to indicate that 3 halfwords follow.

Field 4: Index lower limit (3 bytes)—contains the relative track address (TTR) of the block in which this entry appears.

Field 5: Number of aliases (1 byte)—contains the binary count of the number of aliases assigned to the high-level index. If the index is not a high-level index, this field is zero.

Field 6: Unused (2 bytes)

Figure 44. OS CVOL Index Control Entry

OS CVOL Index Link Entry and Index Pointer Entry

Index Link Entry

Field 1	Field 2	Field 3
X'FFFFFFFFFFFFFFFF' Name	TTR of next block in index (or zero if no next block)	X'00' Count

0 8 11

←Total Length: 12 Bytes→

Index Pointer Entry

Field 1	Field 2	Field 3
Index name (padded to right with blanks if necessary)	TTR of index	X'00' Count

0 8 11

←Total Length: 12 Bytes→

The index link and index pointer entries are similar. An index link entry is used to chain several blocks of an index together, and an index pointer entry is used to chain an index to the next lower-level index. An index link entry is always the last entry in any index block. Each block contains three fields and is 12 bytes long.

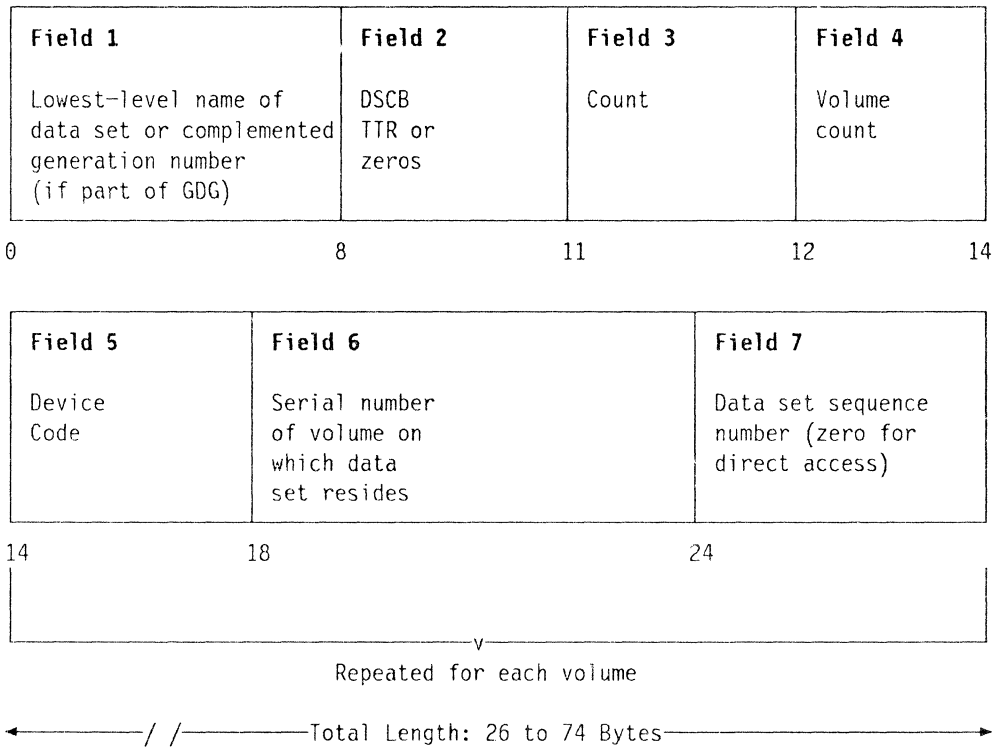
Field 1: Name (8 bytes)—contains the name of the index to which this entry points. If the entry is an index link entry, the name field contains X'FF FF FF FF FF FF FF FF'.

Field 2: Address (3 bytes)—contains either the relative block address (TTR) of the first block of the next level index if it is an index pointer entry, or the relative block address (TTR) of the next block of the same level index if it is an index link entry.

Field 3: Halfword count (1 byte)—contains 1 byte of binary zeros to indicate that the entry ends here.

Figure 45. OS CVOL Index Link and Index Pointer Entries

OS CVOL Data Set Pointer Entry



The data set pointer entry can appear in any index. It contains the simple name of a data set and from one to five 12-byte fields, each of which identifies a volume on which the named data set resides. If the data set resides on more than five volumes, a volume control block pointer entry is substituted for the data set pointer entry. A volume control block pointer entry points to a volume control block or chain of volume control blocks that point to the volumes that contain the data set.

The data set pointer entry varies in length. The length is determined by the formula 14 + 12m, where m is the number of volumes containing the data set. The variable m can be from one to five. The data set pointer entry can appear in any index, and it contains seven fields.

Field 1: Name (8 bytes)—contains the simple name of the data set whose volumes are identified in field 5. If part of a GDG, these names have the format GxxxxV00, where xxxx is the complement of the GDG number.

Field 2: DSCB TTR (3 bytes)—contains the track address (TTR) of the data set control block if the data set resides on one volume. If the data set resides on more than one volume, this field contains binary zeros.

Field 3: Halfword count (1 byte)—contains the binary count of the number of halfwords that follow. The number is found by the formula 6m + 1, where m is the number of volumes on which the data set resides. The variable m can be from one to five.

Figure 46 (Part 1 of 2). OS CVOL Data Set Pointer Entry

Field 4: Volume count (2 bytes)—contains the binary count of the number of volumes identified in field 5 of this entry.

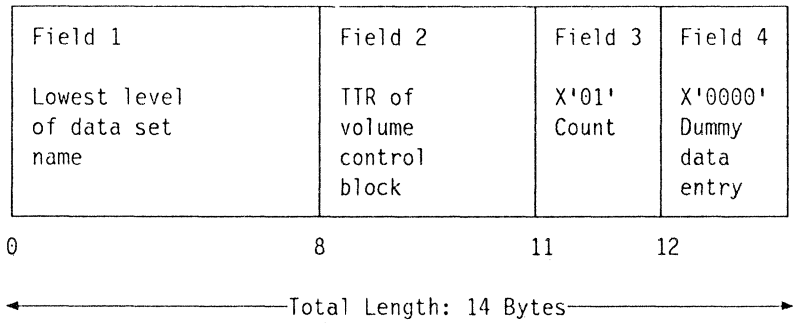
Field 5: Device code (4 bytes)—contains the device code of the device on which the volume with the volume serial number in field 6 can be mounted.

Field 6: Volume serial number (6 bytes)—contains the volume serial number of one of the volumes of the data set.

Field 7: Data set sequence number (2 bytes)—contains the sequence number of the data set on a magnetic tape volume. It is zero for any other device class.

Figure 46 (Part 2 of 2). OS CVOL Data Set Pointer Entry

OS CVOL Volume Control Block Pointer Entry



The volume control block pointer entry is used instead of a data set pointer entry when the data set resides on more than five volumes. This entry points to a volume control block, which, in turn, describes the data set. The entry is 14 bytes long.

Field 1: Name (8 bytes)—contains the last name of the qualified name of the data set identified by this entry.

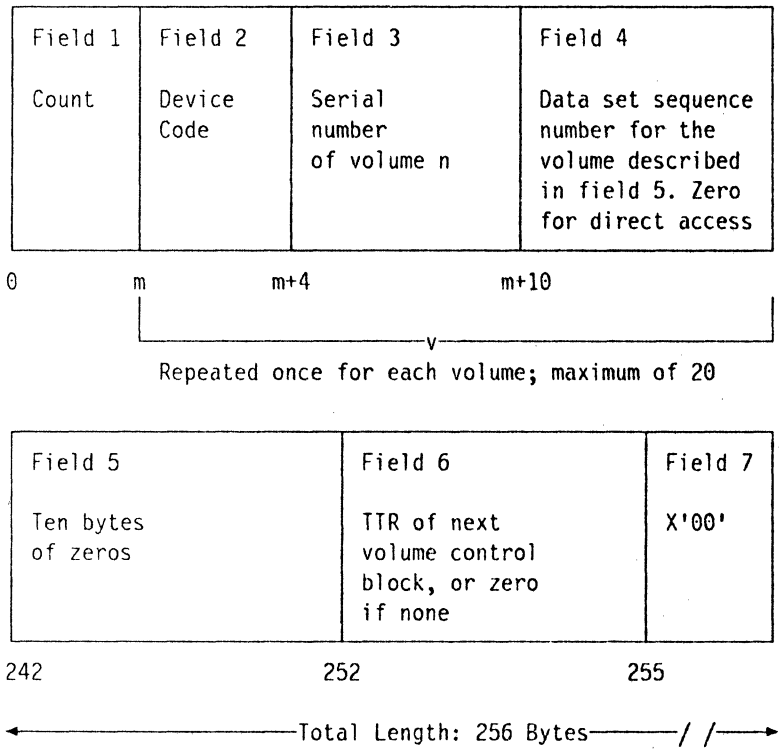
Field 2: Address (3 bytes)—contains the relative block address (TTR) of the volume control block identifying the volumes containing the data set named in field 1.

Field 3: Halfword count (1 byte)—contains a hexadecimal 1 to indicate that 1 halfword follows.

Field 4: Zero field (2 bytes)—contains hexadecimal zeros.

Figure 47. OS CVOL Volume Control Block Pointer Entry

Volume Control Block



A volume control block contains the description of all the volumes of a data set that resides on more than five volumes. If a data set resides on less than six volumes, a volume control block is not built and the volumes are described in a data set pointer entry. One volume control block can describe as many as 20 volumes. Volume control blocks may be chained together to catalog a data set residing on more than 20 volumes.

The volume control block is always 256 bytes long, regardless of the number of volumes described.

Field 1: Volume count (2 bytes)—the first volume control block contains the binary count of the total number of volumes on which the data set resides. The value of this field is reduced by 20 for each subsequent volume control block. If, for example, the data set resides on 61 volumes, there will be four volume control blocks for the data set. The volume count field of each will contain 61, 41, 21, or 1, respectively.

Fields 2, 3, 4: Volume identification (12 to 240 bytes)—contains from 1 to 20 entries, each of which identifies a volume on which the data set resides. Each entry contains a 4-byte device code, a 6-byte volume serial number, and a 2-byte data set sequence number. The data set sequence number is zero for data sets on direct access volumes.

Field 5: Zero field (10 bytes)—contains binary zeros.

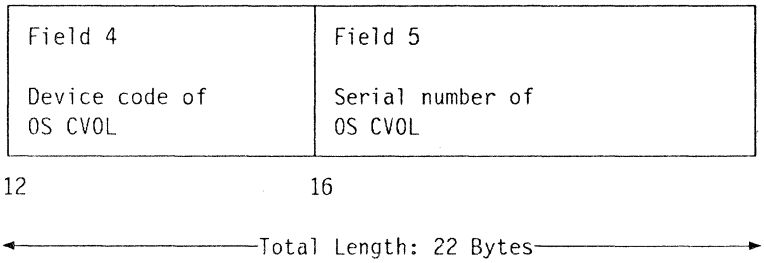
Figure 48 (Part 1 of 2). OS CVOL Volume Control Block

Field 6: Chain address (3 bytes)—contains the relative block address (TTR) of the next volume control block, if additional blocks are needed to describe the data set. If this is the last volume control block for the data set, this field will be set to binary zeros.

Field 7: Zero field (1 byte)—contains binary zeros.

Figure 48 (Part 2 of 2). OS CVOL Volume Control Block

OS CVOL Pointer Entry



The OS CVOL pointer entry is used to indicate that a particular index resides on a volume other than the system residence volume. OS CVOL pointer entries can exist only in the volume index. Each is 22 bytes long.

Field 1: Name (8 bytes)—contains a high-level index name that appears in the volume index of the OS CVOL identified in fields 4 and 5.

Field 2: Address (3 bytes)—contains zeros, because this entry references no other entry in the catalog.

Field 3: Halfword count (1 byte)—contains the hexadecimal value 5 to indicate that 5 halfwords follow.

Field 4: OS CVOL device code (4 bytes)—contains the device code of the specified control volume.

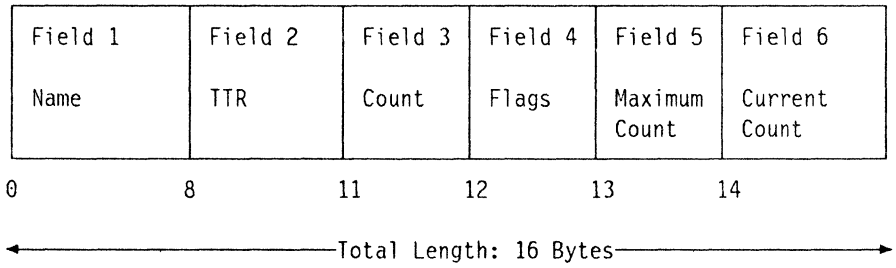
Field 5: OS CVOL serial number (6 bytes)—contains the volume serial number of the OS CVOL which has an entry in its volume index of the same name as this entry.

Figure 49. OS CVOL Pointer Entry

OS CVOL Pointer Entry (OLD)

Until Release 17 of OS MFT/MVT, the OS CVOL pointer entry was the same as the present OS CVOL pointer, except that there was no field 4 (device code); the OS CVOL pointer entry was 18 bytes long. After Release 17, the OS CVOL pointer entry is 22 bytes long. This is mentioned because some OS CVOLs may still contain entries in the old format and the catalog management routines may still check for them.

OS CVOL Generation Index Pointer Entry



A generation index pointer entry is the entry that identifies a generation data group (GDG). It represents the next to the lowest-level of a group of generation data set names. It is created by using the BLDG macro.

Field 1: Name (8 bytes)—this name represents the GDG level that is next to the lowest level of GDG data set names.

Field 2: Address (3 bytes)—contains the relative track address (TTR) of the first block of the level containing the lowest-level GDG names. These names have the format GxxxxV00, where xxxx is a complement of the GDG number.

Field 3: Count (1 byte)—X'02' identifies this entry and indicates the number of halfwords that follow this field.

Field 4: Flags (1 byte)—indicates the options specified by the creator of the GDG.

X'02' = DELETE option.

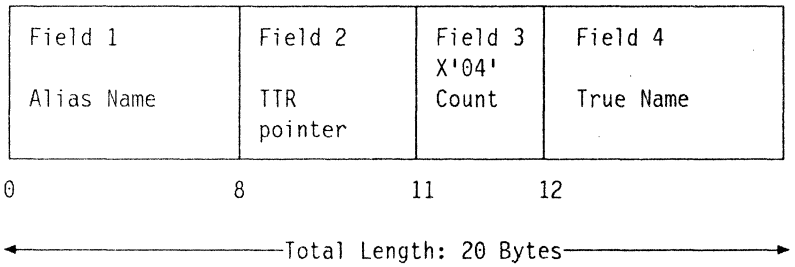
X'01' = EMPTY option.

Field 5: Maximum Count (1 byte)—a binary number that specifies the maximum number of generations allowed in the generation index at one time.

Field 6: Current Count (2 bytes)—the binary count of the number of generations currently cataloged in the generation data group (GDG).

Figure 50. OS CVOL Generation Index Pointer Entry

OS CVOL Alias Name



An alias entry defines an alternative name for the high-level qualifier of a data set name.

Field 1: Name (8 bytes)—contains the alias of the high-level index whose relative track address is found at field 2.

Field 2: Address (3 bytes)—contains the relative track address (TTR) of the first block of the index named in field 4.

Field 3: Count (1 byte)—identifies this entry and contains the binary count of the number of halfwords that follow. The number is X'04'.

Field 4: True name (8 bytes)—contains the name of the index whose alias appears in field 1.

Figure 51. OS CVOL Alias Name

Appendix H. Region Requirements for Access Method Services Jobs

Virtual storage is required in the user's address space for virtual storage access method (VSAM) control blocks, buffers, and if used, the indexed sequential access method (ISAM) interface routines.

Virtual storage information for VSAM control blocks in the user's address space is given below.

Control Blocks	*User Address Space*
Access Method Services	
Any function except BLDINDEX, EXPORTRA, RESETCAT	220.0K
BLDINDEX (without internal sort area)	170.0K
EXPORTRA	445.0K
RESETCAT	270.0K
Catalog Management	
For catalog access	4.0K
Record Management	
VSAM ESDS (1 string is the default)	3.8K
VSAM RRDS (1 string is the default)	3.8K
VSAM LDS (1 string is the default)	3.8K
VSAM KSDS (1 string is the default)	6.1K
Overhead per string (above the default)	1.2K

Besides VSAM, other system components share the user's address space for control blocks and buffers. Roughly, 100K bytes are taken from the user's address space. Users need to count these 100K bytes toward their virtual storage requirement for VSAM jobs.

If there is more than one access method services command in one job step, the virtual storage used to process one access method services command is always released before the next access method services command begins. Thus, if more than one access method services command is used, the minimum region size requirement for that job step is the largest virtual storage used for each access method services function.

The following example demonstrates how to calculate region requirements for an access method services job.

```
//VSAMCAT JOB, REGION=334K
//STEP01  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=A
//VOL       DD VOL=SER=338001,UNIT=3380,DISP=OLD
//SYSIN     DD *

  DEFINE USERCATALOG        -
         (NAME(ICFUCAT1)    -
         CYLINDERS(1 1)     -
         VOLUMES(338001)    -
         STRNO(4)           -
         ICFCATALOG         -
         FILE(VOL)
  DEFINE USERCATALOG        -
         (NAME(ICFUCAT2)    -
         CYLINDERS(1 1)     -
         VOLUMES(338001)    -
         ICFCATALOG         -
         FILE(VOL)
  DEFINE CLUSTER            -
         (NAME(IBMUSER.KSDS)-
         TRACKS(10 1)       -
         VOLUMES(338001))
  DEFINE AIX                -
         (NAME(IBMUSER.KSDS.AIX)  -
         TRACKS(1 1)        -
         VOLUMES(338001)    -
         RELATE(IBMUSER.KSDS))
/*
//
```

Virtual memory requirements calculations for the above four access method services commands proceed as follows. The number given in parentheses is the total virtual memory requirement rounded to the nearest even number.

DEFINE USERCATALOG (ICFUCAT1)

Access method services	220.0K	
Catalog management	4.0K	
Record management	9.7K	(7.3K + (1.2K*2))
Other system components	100.0K	

	333.7K	(334K)

DEFINE USERCATALOG (ICFUCAT2)

Access method services	220.0K	
Catalog management	4.0K	
Record management	7.3K	(VSAM KSDS)
Other system components	100.0K	

	331.3K	(332K)

DEFINE CLUSTER

Access method services	220.0K	
Catalog management	4.0K	
Record management	6.1K	(VSAM KSDS)
Other system components	100.0K	

	330.1K	(331K)

DEFINE AIX

Access method services	220.0K	
Catalog management	4.0K	
Record management	6.1K	(VSAM KSDS)
Other system components	100.0K	

	330.1K	(331K)

Thus, from the above four calculations, 334K (DEFINE USERCATALOG (ICFUCAT1)) represents the minimum region size we need to use to run this job.

Glossary of Terms and Abbreviations

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 Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699.

access method services. A multifunction service program that defines VSAM data sets and allocates space for them, converts indexed-sequential data sets to key-sequenced data sets with indexes, modifies data set attributes in the catalog, reorganizes data sets, facilitates data portability between operating systems, creates backup copies of data sets and indexes, helps make inaccessible data sets accessible, and lists the records of data sets and catalogs.

AIX. (*See* alternate index.)

alias. An alternative name for an entry.

alias entry. An entry that relates an alias (alternate entryname) to the real entryname of a user catalog or non-VSAM data set.

alternate index. In systems with VSAM, a collection of index entries related to a given base cluster and organized by an alternate key, that is, a key other than the prime key of the associated base cluster data records. Its function is to provide an alternate directory for locating records in the data component of a base cluster.

alternate index entry. A catalog entry that contains information about an alternate index. An alternate index is conceptually a key-sequenced cluster, and is cataloged in the same way. An alternate index entry points to a data entry and an index entry to describe the alternate index's components, and to a cluster entry to identify the alternate index's base cluster. (*See also* cluster entry.)

alternate index record. A collection of items used to sequence and locate one or more data records in a base cluster. Each alternate index record contains an alternate key value and one or more pointers. When the alternate index supports a key-sequenced data set, each data record's prime key value is the pointer. When the alternate index supports an entry-sequenced data set, the data record's RBA value is the pointer. (*See also* alternate index, alternate key, base cluster, and key.)

alternate key. One or more characters within a data record, used to identify the data record or control its use. Unlike the prime key, the alternate key can identify more than one data record. (*See also* key and key field.)

application. As used in this publication, the use to which an access method is put or the end result that it serves; contrasted to the internal operation of the access method.

authorized program facility. A facility that permits the identification of programs that are authorized to use restricted functions.

backup data set. A copy that can be used to replace or reconstruct a damaged data set.

base cluster. The VSAM cluster whose data records are to be accessed through a path. Usually, a base cluster is the key-sequenced or entry-sequenced data set which an alternate index supports (that is, an alternate index is used by VSAM to sequence and locate the data records of a base cluster). (*See also* alternate index and path.)

basic catalog structure. Is the name of the actual catalog structure with the integrated catalog facility environment. integrated catalog facility is composed of a BCS together with its related VTOCs and VVDSs (VSAM volume data sets).

BCS. (*See* basic catalog structure.)

CA. (*See* control area.)

CAS. Catalog address space.

catalog. (*See* master catalog and user catalog.)

catalog connector. A catalog entry, called either a user catalog entry or a catalog connector entry, in the master catalog that points to a user catalog's volume (that is, it contains the volume serial number of the direct access volume that contains the user catalog).

catalog recovery area. In systems with VSAM, an entry-sequenced file that exists on each volume owned by a recoverable catalog, including the catalog itself. The CRA contains records that are duplicates of the catalog entries describing the volume and the files it contains.

cell. An occurrence of information such as passwords, volume information, or associations.

CI. (*See* control interval.)

cluster. A data component and an index component when data is key sequenced; a data component alone when data is entry sequenced.

cluster entry. A catalog entry that contains information about a key-sequenced or entry-sequenced VSAM cluster: ownership, cluster attributes, and the cluster's passwords and protection attributes. A key-sequenced cluster entry points to a data entry and an index entry. An entry-sequenced cluster entry points to a data entry.

component. The data portion or, for a key-sequenced cluster, alternate index, or VSAM catalog, the index portion or a VSAM object. In this book, the components of an object are usually referred to as the object's data component and index component.

control area. A group of control intervals used as a unit for formatting a data set before adding records to it. Also, in a key-sequenced data set, the set of control intervals pointed to by a sequence-set index record; used by VSAM for distributing free space and for placing a sequence-set index record adjacent to its data.

control interval. A fixed-length area of auxiliary-storage space in which VSAM stores records and distributes free space. It is the unit of information transmitted to or from auxiliary storage by VSAM.

control volume. A volume that contains one or more indexes of the catalog.

CRA. (*See* catalog recovery area.)

CVOL. (*See* control volume.)

DASD. direct access storage device.

data component. That part of a VSAM data set, alternate index, or catalog that contains the object's data records.

data entry. A catalog entry that describes the data component of a cluster, alternate index, page spaces, or catalog. A data entry contains the data component's attributes, allocation and extent information, and statistics. A data entry for a cluster's or catalog's data component can also contain the data component's passwords and protection attributes.

data integrity. Preservation of data or programs for their intended purpose. As used in this publication, the safety of data from inadvertent destruction or alteration.

data record. A collection of items of information from the standpoint of its use in an application, as a user supplies it to VSAM for storage.

data security. Prevention of access to or use of data or programs without authorization. As used in this publication, the safety of data from unauthorized use, theft, or purposeful destruction.

data set. The major unit of data storage and retrieval in the operating system, consisting of data in a prescribed arrangement and described by control information to which the system has access. As used in this publication, a collection of fixed- or variable-length records in auxiliary storage, arranged by VSAM in key sequence or in entry sequence. (*See also* key-sequenced data set and entry-sequenced data set.)

data set control block. A data set label for a data set in direct access storage.

DFDSS. Data Facility Data Set Services (an IBM program product).

direct access. The retrieval or storage of data by a reference to its location in a data set rather than relative to the previously retrieved or stored data.

DSCB. (*See* data set control block.)

dynamic allocation. The allocation of a data set or volume by the use of the data set name or volume serial number rather than by the use of information contained in a JCL statement.

entry. A collection of information about a cataloged object in a VSAM master or user catalog. Each entry resides in one or more 512-byte record.

entry name. A unique name for each component or object as it is identified in a catalog. The entryname is the same as the dsname in a DD statement that describes the object.

entry sequence. The order in which data records are physically arranged (according to ascending RBA) in auxiliary storage, without respect to their contents.

entry-sequenced data set. A data set whose records are loaded without respect to their contents, and whose RBAs cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

ESDS. (*See* entry-sequenced data set.)

extent. A continuous space allocated on a direct-access storage volume, reserved for a particular data space or data set. An extent of a data set contains a whole number of control areas.

field. In a record or a control block, a specified area used for a particular category of data or control information.

GDG. (*See* generation data group.)

GDS. (*See* generation data set.)

generation data group. An entry that permits non-VSAM data sets to be associated with other non-VSAM data sets as generation data sets.

generation data set. One of a collection of historically related non-VSAM data sets; the collection of these data sets is known as a generation data group.

index. As used in this publication, an ordered collection of pairs, each consisting of a key and a pointer, used by VSAM to sequence and locate the records of a key-sequenced data set; organized in levels of index records. (*See also* index level, index set, and sequence set.)

index component. That part of a key-sequenced data set, catalog, or alternate index, that establishes the sequence of the data records within the object it indexes. The index is used to locate each record in the object's data component, based on the record's key value.

index entry. A catalog entry that describes the index component of a key-sequenced cluster, alternate index, or catalog. An index entry contains the index component's attributes, passwords and protection attributes, allocation and extent information, and statistics.

index level. A set of index records that order and give the location of records in the next lower level or of control intervals in the data set that it controls.

index record. A collection of index entries that are retrieved and stored as a group. (*Contrast* with data record.)

index set. The set of index levels above the sequence set. The index set and the sequence set together comprise the index.

initial program load. (1)The initialization procedure that causes an operating system to commence operation. (2)The process by which a configuration image is loaded into storage at the beginning of a work day or after a system malfunction.

integrated catalog facility. The name of the catalog associated with the Data Facility Product program product.

integrity. (*See* data integrity.)

IPL. (*See* initial program load.)

key. One or more characters within an item of data that are used to identify it or control its use. As used in this publication, one or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records.

key-sequenced data set. A data set whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the data set in key sequence by means of distributed free space. RBAs of records can change.

KSDS. (*See* key-sequenced data set.)

LDS. (*See* linear data set.)

linear data set. A data set that has no record definition field (RDF), no control interval definition field (CIDF), and can only be accessed in control interval mode. It can only be defined in an integrated catalog facility catalog.

logical record. Contains a group of logically related cells that are physically adjacent.

master catalog. A key-sequenced data set with an index containing extensive data-set and volume information that VSAM requires to locate data sets, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and to accumulate usage statistics for data sets.

non-VSAM entry. A catalog entry that describes a non-VSAM data set. A non-VSAM entry contains the data-set's volume serial number and device type. If the data set resides on a magnetic tape volume, the entry can also identify the data set's file number. When the data set resides on a direct access device, the operating system obtains further information by examining the data set's DSCB (data set control block) in the volume's VTOC (volume table of contents).

object. A logical entity created by VSAM, such as a cluster (VSAM data set) and its components, an alternate index and its components, a VSAM catalog and its components, a path, or a VSAM data space.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

OS. (*See* operating system.)

page space. A system data set. A page space is cataloged as an entry sequenced cluster (that is, the page space entry is similar to a cluster entry, and it points to a data entry).

password. A unique string of characters stored in a catalog that a program or a computer operator at the console must supply to meet security requirements before the program gains access to a data set.

path. A data set name for the combination of an alternate index and its base cluster, or an alias for a VSAM data set.

path entry. A catalog entry that contains information about a path, and that points to the path's related objects.

primary space allocation. Initially allocated space on a direct access storage device, occupied by or reserved for a particular data set.

prime index. The index component of a key-sequenced data set. (*See also* index and alternate index.)

prime key. (*See* key.)

RACF. (*See* Resource Access Control Facility.)

RBA. (*See* relative byte address.)

record. (*See* index record, data record, logical record, spanned 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.)

relative byte address. The displacement of a data record or a control interval from the beginning of the data set to which it belongs; independent of the manner in which the data set is stored.

relative record data set. A data set whose records are loaded into fixed-length slots.

Resource Access Control Facility. A program product that provides for access control by identifying and verifying users to the system authorizing access to DASD data sets, logging detected unauthorized attempts to enter the system, and logging detected accesses to protected data sets.

RRDS. (*See* relative record data set.)

secondary space allocation. A contiguous space on a direct access device, occupied by or reserved for a particular data set, which is allocated after space in the primary extent has been exhausted. (*See also* primary space allocation.)

security. (*See* data security.)

sequence set. The lowest level of the index of a key-sequenced data set; it gives the locations of the control intervals in the data set and orders them by the key sequence of the data records they contain. The sequence set and the index set together comprise the index.

spanned record. A logical record whose length exceeds control interval length, and crosses (or spans) one or more control interval boundaries within a control area.

sphere record. A collection of logically related subrecords in one VSAM logical record.

subrecord. The user definition level of a sphere, such as an AIX, cluster, or generation data set.

terminal monitor program. In TSO, a program that accepts and interprets commands from the terminal, and causes the appropriate command processors to be scheduled and executed.

time sharing option. An optional configuration of the operating system that provides conversational time sharing from remote stations.

TMP. (*See* terminal monitor program.)

true name. Refers to the names generated for the data and index components as a result of the DEFINE command.

TSO. (*See* time sharing option.)

user catalog. A catalog used in the same way as the master catalog, but optional and pointed to by the master catalog, and also used to lessen the contention for the master catalog and to facilitate volume portability.

user catalog connector. (*See* catalog connector.)

virtual storage access method. An access method for direct or sequential processing of fixed- and variable-length records on direct access devices. The records in a VSAM data set or file can be organized in logical sequence by a key field (key sequence), in the physical sequence in which they are written on the data set or file (entry-sequence), or by relative record number.

volume table of contents. A table on a direct access volume that describes each data set on the volume.

VSAM. (*See* virtual storage access method.)

VSAM volume control record. The first logical record in the VVDS that contains information to manage DASD space and the BCS back pointers

VSAM volume data set. The VSAM volume data set is used to describe data set characteristics of VSAM data sets residing on a given volume. There is one, and only one, VVDS for each volume containing VSAM data sets cataloged in an integrated catalog facility catalog.

VSAM volume record. The VSAM volume record is a VSAM logical record within a VVDS.

VTOC. (*See* volume table of contents.)

VVCR. (*See* VSAM volume control record.)

VVDS. (*See* VSAM volume data set.)

VVR. (*See* VSAM volume record.)

Index

A

access method services
  ALTER 21
  BLDINDEX 21
  CHKLIST 21
  CNVTCAT 21
  commands 180
  data protection 32
  DEFINE 21
  DELETE 21
  DIAGNOSE 21
  EXAMINE 21
  EXPORT 21
  IMPORT 21
  LISTCAT 21
  PRINT 21
  printing 115
  REPRO 21
  VERIFY 21
alias
  identifying a user catalog 27
alias name 233
  entry 278
  use in retrieving catalog information 251–252
ALLOCATE command 235
allocating the volume 182
  DD statement 182
  dynamic allocation 182
  VOLUMES parameter 182
allocation amount
  secondary 187
allocation units
  VSAM catalog 184
allocation, space
  BCS 25, 27
  catalog recovery area (CRA)
    VSAM catalog only 188
  VSAM catalog 178, 183
  VSAM objects 183
    primary allocation 183
    secondary allocation 183
ALTER command 21, 236
  generic names 45, 200
  REMOVEVOLUMES 86–88, 96
ALTER REMOVEVOLUMES
  VSAM volume cleanup 205
altering attributes of catalog entries 44, 200
alternate index
  DEFINE RECATALOG 89
alternate master catalog 43–44
  name of 44
  specifying 44
AMASPZAP 96
analysis of DIAGNOSE sample output 136
APF (authorized program facility)
  access method services processing 31
  authorization 31
  establishing authorization 31
  terminal monitor program 31
  TSO 31
application programs
  converting 64
assigning space
  VSAM catalog 183
attributes, catalog entry
  altering 44, 200
authorized program facility
  See APF
automatic catalog backup 206

B

backup and recovery 73–113
  data set 96
  examples 97
  EXPORT IMPORT 94
  solutions for problems 85
  VVDS 96
backup catalog
  automatic 206
  updating 204
BCS (basic catalog structure)
  cells 149
  components 6
  DEFINE command 25
  DELETE FORCE 92
  DELETE RECOVERY 92
  delete-in-progress bit 91
  description 5
  estimating size 27
  estimating space 27
  performance 28, 43
  recovery examples 97–112
  relationship to VVDS 86
  reorganization 75, 82–84
  REPRO NOMERGECAT 75
  requests 30, 43
  setting up 27
  sharing 43
  space allocation 25, 27
  space requirements 27
BLDA statement 233
BLDG statement 233
BLDINDEX command 21
BLDX statement 233
BUFFERSPACE parameter
  DEFINE ICFCATALOG 30

C

CAMLST macro
  parameter list 236
  with BLDA operand 259
  with BLDG operand 256
  with BLDX operand 254
  with BLOCK operand 253
  with CAT(BX) operand 264
  with DLTA operand 260
  with DLTX operand 258
  with DRPX operand 263
  with LNKX operand 261
  with RECAT operand 268
  with UNCAT operand 267
CAS (catalog address space) 3, 139
CATALG statement 238
CATALOG
  return codes
    CVOL 240
catalog address space
  See CAS (catalog address space)
catalog control record (CCR) 191
catalog entry
  rebuilding 219
CATALOG macro
  with CAT(BX) operand 264
  with RECAT operand 268
  with UNCAT operand 267
catalog maintenance
  using CATALOG macro 264—269
  using LOCATE macro 247—254
catalog performance
  improving 202
catalog recovery area
  See CRA
catalog recovery procedures 133
catalog recovery tools 206
catalog space
  use 178
catalog, CVOL
  accessing 236
  functions not supported 239
catalog, integrated catalog facility
  attributes for define 25
  central information point 22
  copying 75
  defining space 18, 19
  DELETE command
    ALIAS 48
    ERASE|NOERASE 37, 47
    FORCE|NOFORCE 47
    PURGE|NOPURGE 47
    RECOVERY|NORECOVERY 48
    SCRATCH|NOSCRATCH 48
    TRUENAME parameter 48
  estimating space 27
  listing 115
  monitoring use 82
catalog, integrated catalog facility *(continued)*
  number of extents 18
  performance-related attributes 25
  printing 115
  procedures 21
  self-describing sphere record 6
  space assigned 18
  space requirements 27
  structure 5
  suballocation space 18
  unique space 18
  volume ownership 19
  VTOC entries 19
catalog, OS CVOL
  functions not supported 236
  non-VSAM data set 233
  restrictions 238
  support 233
catalog, VSAM
  cleanup 206
  copying 190
  data and space management 176
  DEFINE command 181
  information contained 177
  recoverable
    exporting 190
    importing 190
    space allocation 188
  secondary allocation amount 187
  timestamps 179
  volume ownership 178
cataloging non-VSAM data sets
  coding example 265
  macro specifications 264
  return codes 266
cataloging objects 180
catalogs
  central information point 180
  data set password protection 35
  entry format 246
  maintaining
    using CATALOG macro 264—269
    using LOCATE macro 247—254
  master 245
  order of search 245
  password protected 32
  private 246
  user 245
CATBX statement 233
CATLG statement 233
CCR (catalog control record) 191
cell formats 8—13
cells
  BCS 6
  VVR 18
changing attributes
  master catalog 228
changing the volume serial number 199

checkpoint restart 226
CHKLIST command 21
cluster
  DEFINE RECATALOG 89
CNVTCAT command 21
  failure 62
common problems 95
COMPARE parameter 116
CONNECT parameter 95
CONNECT statement
  IEHPROGM 237
considerations for multivolume data sets 224
control area
  VSAM catalog data component 175
control area size
  integrated catalog facility catalog 19
  limited by size of sequence set record 176
  used in estimating size of BCS 28
control area split
  VSAM catalog 178
control interval size
  integrated catalog facility catalog 19
  used in estimating size of BCS 28
  VSAM catalog 202
control interval split
  VSAM catalog 178, 203
conversion 55
  application programs 64
  master catalog 62, 72
  OS CVOLS 63
  OS CVOL 69
  VSAM catalog 55
    backup 60
    catalog damage 60
    EXPORT IMPORT command 56
    full VTOC 59
    multiple device types 60
    NON-VSAM data sets 58
    nonrecoverable catalog 59, 68
    nonrecoverable volume 70
    one catalog at a time 55
    one volume at a time 55
    recoverable catalog 59, 67, 70
    removable volumes 60
    removing an unavailable volume 70
    sample procedures 66
    unavailable unused volumes 59
    UNIQUE data sets 58
    verification 61
  VSAM catalog to integrated catalog facility catalog
    DASD space requirements 57
    full volumes 57
copying a VSAM catalog 190
  preparation 191
  procedures 193
copying catalog entry from CRA 219
CRA (catalog recovery area)
  catalogs with CRAs 212
  catalogs without CRAs 212
CRA (catalog recovery area) *(continued)*
  copying a catalog 219
  VSAM recoverable catalog 182, 216
creating a VSAM catalog 181
creating objects 180
CYLINDERS parameter 183

D

data
  on separate volume 41, 189
  protection 32
data component
  VSAM catalog 175
data control area size 28
data control interval size 28
data integrity 30
data management
  use of VSAM catalog 176
DATA parameter
  specifying attributes 24
data protection 30
data recovery
  utility programs 209
data security 30
data set
  backup and recovery 96
  catalog password protection 35
  DEFINE RECATALOG 88
  DELETE NOSCRATCH 91
  inaccessible 211
  multivolume 224
  not properly closed 210
  password protected 32
  pointer entry 273
data set record
  information 177
data space
  extended 178
  managing space on storage volume 178
  suballocated VSAM objects 179
  VSAM catalog 183
DD statement
  allocating the volume 182
debug switches 228
debugging
  generalized trace facility (GTF) 228
DEFINE command 21, 235, 236
  BCS STRNO 30
  data set RECATALOG—required parameters 88
  defining a BCS 25
  defining a catalog 22, 180
  defining a cluster 89
  defining a CVOL pointer 237
  defining a VVDS 26
  defining an alternate index 89
  duplicate cluster name 87
  ICFCATALOG BUFFERSPACE 30
  MODEL parameter 39

DEFINE command *(continued)*
organization of parameters 23
path RECATALOG 90
specifying attributes 24
USERCATALOG 25
VVDS NORECATALOG 26
VVDS RECATALOG 89, 93
define space
integrated catalog facility catalog 18
defining a VSAM catalog 181
defining a VSAM recoverable catalog 182
defining an integrated catalog facility catalog
example 49, 50
defining non-VSAM objects 180
defining objects
specifying attributes 23
types of objects 23
defining user catalog
TSO 181
DELETE
alternate index 47
DELETE command 21, 235
ALIAS 48
CLUSTER 46
component VVR 90
data set NOSCRATCH 91
ERASE|NOERASE 37, 47
example 112
FORCE|NOFORCE 47
objects or entries 46
parameters for 46
PURGE|NOPURGE 47
RECOVERY|NORECOVERY 48
SCRATCH|NOSCRATCH 48
TRUENAME parameter 48, 90
USERCATALOG 46
USERCATALOG FORCE 92
USERCATALOG RECOVERY 92
using 46
VSAM DELETE 233
VVDS NOSCRATCH 93
VVDS RECOVERY 93
DELETE NOSCRATCH
VSAM catalog cleanup 206
DELETE SPACE FORCE
VSAM volume cleanup 206
deleting alias catalog record 48
deleting an integrated catalog facility catalog
example 52
DFDSS (Data Facility Data Set Services)
volume dump and restore 96
DIAGNOSE command 21, 115–133
after ALTER REMOVEVOLUMES 88
analysis of sample output 136
COMPARE parameter 116
DIAGNOSE output
interpreting 119
error messages 118, 120
EXCLUDE parameter 115
DIAGNOSE command *(continued)*
execution error messages 121
INCLUDE parameter 115
messages 117
entry notation 119
execution errors 119
interpreting 119
record notation 118
summary 119
syntax errors 119
processing 116
summary messages 118
VTOC 116
DIAGNOSE error messages
DIAGNOSE command 119
DIAGNOSE output 136
analysis of 136
DIAGNOSE command 117
sample 134
storage estimate 117
diagnosing a data set 115
diagnostic aids 227
DISCONNECT parameter 95
display data set statistics
SHOWCB 202
DLTA statement 233
DLTX statement 233
DRPX statement 233, 238
dump facility
SNAP 228
dynamic allocation 49
allocating the volume 182
data set and volume 42

E

error
closing data set 210
error analysis routines
exits 227
error messages
DIAGNOSE 120
message, explanation, and recovery
procedure 121–133
error prevention 81
estimating catalog space requirements 18
estimating storage for DIAGNOSE 117
estimating VSAM catalog space 185
EXAMINE command 21
examples
defining an integrated catalog facility catalog 49, 50
EXCLUDE parameter 115
execute form
SHOWCAT macro 172
execution error messages
DIAGNOSE 121
DIAGNOSE command 119
message, explanation, and recovery
procedure 121–133

exits to error analysis routines 227
EXPORT command 21
  backup and recovery 94–96
  DISCONNECT parameter 95
EXPORT/IMPORT command
  BCS
    reorganization 82
  rebuilding a catalog 219
  recovering shared catalogs 82
exporting
  VSAM recoverable catalogs 190
EXPORTRA/IMPORTRA command
  catalog recovery 217
  recovering catalog entries 219
  VSAM recoverable catalogs 190
expressions
  for operands 173
extension records
  for catalogs 14

F

FILE parameter
  DELETE UCAT FORCE 92
format
  execute form
    SHOWCAT macro 172
  list form
    SHOWCAT macro 172
  standard form
    SHOWCAT macro 168
free chain rebuild
  VSAM catalogs 191

G

generalized trace facility (GTF) 228
generated name
  VSAM catalog 179
generation data group 233
generation data set
  name
    use in retrieving catalog information 250
generation index
  pointer entry 277
generic name 45, 200
  renaming 46, 201
glossary 283
GTF (generalized trace facility) 228

H

high-key range 175

I

IDC01360I message 131, 137
IDC01371I message 130
IDC11361I message 120, 130, 131
IDC11362I message 131
IDC11367I message 132
IDC11373I message 132
IDC11374I messages 132
IDC11375I message 132
IDC21363I message 131, 136
IDC21364I message 136
IDC21365I message 137
IDC21372I message 121
IDC3009I message 87, 95, 96
IDC31366I message 120
IDC31368I message 120
IDC31369I message 137
IDC31370I message 121, 130
IDC31376I message 130
IDC31377I message 130
IDC3351I message 95
identifying a catalog's volume 42
IEC161I message 95
IEC331I message 95
IEHLIST LISTVTOC 115
IEHPROGM program 96
  CONNECT statement 237
IMPORT command 21
  backup and recovery 94–96
  BCS
    control area size 95
    reorganization 82
  CONNECT parameter 95
  DD statement DISP parameter 95
  recovering shared catalogs 78
importing
  VSAM recoverable catalogs 190
IMPORTRA command
  restoring catalog entries 221
  VSAM recoverable catalogs 190
improving catalog performance 202
inaccessible data set 211
inaccessible volume 214
INCLUDE parameter 115
index
  control entry 271
  link entry 272
  on separate volume 41, 189
  pointer entry 272
INDEX and CAMLST macros
  with BLDG operand 256–258
  with BLDX operand 254–256
  with DLTX operand 258–259
index component
  VSAM catalog 175
index control interval size 41, 189
INDEX macro
  return codes
    CVOL 240

INDEX macro *(continued)*
with BLDA operand 259—260
with DLTA operand 260—261
with DRPX operand 263—264
with LNKX operand 261—263
index options 40, 188
summary 42, 190
INDEX parameter
specifying attributes 24
index record
replication of 41
index set
records 40, 189
installation procedures
JOBCAT/STEPCAT DD statements 64
revising 64
UNIQUE 64
installation procedures, revision of 64
integrated catalog facility
converting master catalog to 62
converting OS CVOLS to 63
reorganization 62
revising installation procedures 64
sample procedures for conversion 66
a recoverable catalog 67
master catalog 72
nonrecoverable catalog 68
OS/CVOL 69
recoverable catalog 70
integrated catalog facility catalog
attributes for define 25
copying 75
defining
example 49, 50
defining space 18, 19
DELETE command
ALIAS 48
ERASE|NOERASE 37, 47
FORCE|NOFORCE 47
PURGE|NOPURGE 47
RECOVERY|NORECOVERY 48
SCRATCH|NOSCRATCH 48
TRUENAME parameter 48
deleting
example 52
estimating space 27
listing 115
master 246
monitoring use 82
number of extents 18
order of search 246
performance-related attributes 25
printing 115
procedures 21
self-describing sphere record 6
space assigned 18
space requirements 27
structure 5
suballocation space 18
integrated catalog facility catalog *(continued)*
unique space 18
user 246
volume ownership 19
VTOC entries 19
interpreting DIAGNOSE output 119
interpreting error messages 95
invocation error messages 120

J

JCL (job control language)
data set and volume allocation 42
identifying a catalog's volume 42
RESETCAT processing 225
JOBCAT/STEPCAT DD statement
ALTER REMOVEVOLUMES 205
AMP parameters 26
catalog order of search 245
copying a VSAM catalog 191, 192
DELETE SCRATCH|NOSCRATCH 48
DELETE UNSERCATALOG RECOVERY 92
DELETE USERCATALOG FORCE 92
DELETE VVR 91
dumping a VSAM catalog 195
production job stream 64
RESETCAT 225
restrictions with CVOLs 239
when using an alias 27, 51
JOBCAT/STEPCAT user catalog
SHOWCAT macro 168

K

keyrange data set 16, 20

L

list form
SHOWCAT macro 172
LISTALC command 235
LISTCAT command 21, 115, 235, 236
use with DELETE 47
LISTCRA command
identifying mismatches 207
listing the catalog recovery area's contents 218
to determine damage to data set 211
LISTDS command 235
listing catalog information 115
listing the catalog recovery area's contents 218
listing VSAM information 202
LNKX statement 233
LOCATE macro
retrieving catalog information
by alias name 251—252
by data set name 247—249
by generation name 250—251
by relative block address 253—254
return codes
CVOL 240

LOCATE statement 233
low-key range 175

M

macros 173
  See also format and individual macros
  expressions that can be used for operands 173
macros, data management
  CATALOG 264–269
  LOCATE 246–249
mapping requests 233
  TSO to integrated catalog facility or VSAM catalog 233
Mass Storage System (MSS)
  user catalog defined 181
mass storage volume
  user catalog defined on 181
master catalog 23, 181, 246
  alternate 43–44
  order of search 247
master catalog, conversion 72
master catalog, converting 62
messages
  from DIAGNOSE 118
  IDC3009I 87, 95, 96
  IDC3351I 95
  IEC331I 95
  VSAM 227
mismatches between CRA and catalog 207
MODEL parameter
  in DEFINE command 39
modeling
  overriding attributes 39
  specifying MODEL parameter 39
modeling objects 39
modeling one entry after another 39
MODIFY system command 139
monitoring usage 82
mounted volume required 48
multivolume data set 224

N

names
  VSAM data space 179
non-VSAM data set 233, 237, 238
nonrecoverable catalog, conversion of 68
nonrecoverable catalog, removing an unavailable volume 70
notation for macros, operand 173
number of extents
  integrated catalog facility catalog 18
  VSAM data set 18

O

objects
  defining 23
  modeling 39
objects that cannot be modeled 39
operand notation
  for macros 173
  SHOWCAT 167
OS CVOL
  accessing 236
  ALLOCATE command 235
  ALTER command 235
  BLDA statement 233
  BLDG statement 233
  BLDX statement 233
  CATALG statement 238
  CATBX statement 233
  CATLG statement 233
  DEFINE command 235
  DELETE command 233, 235
  DLTA statement 233
  DLTX statement 233
  DRPX statement 233, 238
  functions not supported
    ALTER command 236
    DEFINE command 236
    LISTCAT command 236
  LISTALC command 235
  LISTCAT command 235
  LISTDS command 235
  LNKX statement 233
  LOCATE statement 233
  pointer in master catalog 237
  RECATLG statement 233, 238
  RENAME command 235
  restrictions 238
  return codes 239
  SUPERLOCATE statement 233
  support 233
  UCATDX statement 233
  UNCATLG statement 233, 238
OS CVOL processor 233
  mapping requests 233
OS CVOLS
  converting 63
OS/CVOL 69

P

password
  control access 32
  control password 32
  degrees of security 32
  full access 32
  master password 32
  prompting 35
  RACF 35
  read access 33

password *(continued)*
read password 33
update access 33
update password 33
VSAM 32
password protection
catalog 34, 35
data set 34, 35
USVR 37
VSAM data sets 32
path
DEFINE RECATALOG 90
performance 43
BCS 28
options that influence 40, 188
performance measurement
statistics kept by VSAM 202
performance-related attributes
integrated catalog facility catalog 25
PRINT command 21, 115
printing
access method services 115
examples 112
IEHLIST LISTVTOC 115
LISTCAT command 115
PRINT command 115
private catalog 246
problems
solutions 85
procedures
copying a VSAM catalog 193
procedures for catalog conversion 66
protection
RACF 35

Q

qualified name 200
qualified names 45

R

RACF (Resource Access Control Facility)
authorization checking 36
ERASE option 37
generic profiles 36
OS CVOL
facility 237
protected data sets 237
password protection 35
reading catalog information
using a data set name 247—249
using a generation name 250—251
using an alias name 251—252
recataloging a data set
coding example 269
macro specification 268
return codes 266
RECATLG statement 233, 238
record
extension 14
nonsphere 6
sphere 6
record size
for catalogs 14
RECORDS parameter 183
RECOVERABLE attribute 206
recoverable catalog, conversion of 67
recoverable catalog, removing an unavailable volume 70
RECOVERABLE parameter
DEFINE USERCATALOG COMMAND 188
VSAM catalog 182
recovering a VSAM recoverable catalog 216
recovering damaged BCS entries 133
recovering damaged VVDS entries 134
recovering shared catalogs 78
recovery and backup 76
recovery function
VSAM volume 215
recovery procedures 133, 134
damaged BCS entries 133
damaged GDG entries 133
damaged VVDS entries 134
relative generation number 250
reload
VSAM catalog 203
RENAME command 235
renaming generically named entries 46, 201
reorganization
BCS 75, 83
repair option
data recovery 209
replication of index records 189
REPRO command 21
backing up a VSAM catalog 190
reloading backup copy 190
BCS reorganization 75
copying a VSAM catalog 190
NOMERGECAT BCS 75
requests
BCS 30, 43
reset operation
data recovery 209
RESETCAT command 221
catalog recovery 217
processing
JCL requirements 225
requirements 224
workfile space requirements 224
resetting catalog entries 221
Resource Access Control Facility
See RACF
restarting programs 226
restrictions and options 227
restoring catalog entry
EXPORTRA command 221

restoring catalog entry *(continued)*
IMPORTRA command 221
return codes
CATALOG macro 266
CVOL 240
considerations 246
INDEX macro
CVOL 240
LOCATE macro 249
CVOL 240
OS CVOL 239
SUPERLOCATE macro
CVOL 240

S

sample DIAGNOSE output 134
scanning a data set 115
secondary allocation
amount
VSAM catalog 187
security 30
selecting a solution to a problem 85
selective checking
DIAGNOSE command
EXCLUDE parameter 115
INCLUDE parameter 115
sequence set record
VSAM catalog 176
shared catalogs, recovery 78
SHAREOPTIONS parameter
BCS 43
VVDS 43
sharing 43
BCS 43
VVDS 43
SHOWCAT macro
execute form 172
list form 172
operand notation 167
standard form 168
SMF (System Management Facilities)
password violation 35
type 36 records 81
solutions for problems 85
space allocation
BCS 25, 27
CRA (catalog recovery area)
VSAM catalog only 188
VSAM catalog 178, 183
VSAM objects 183
primary allocation 183
secondary allocation 183
space assigned
integrated catalog facility catalog 18
space management
use of VSAM catalog 176
space requirements
BCS 27
space requirements *(continued)*
integrated catalog facility catalog 27
VSAM catalog 185
estimating 185
workfile 224
sphere records 6
splits
control area 30
control interval 30
standard form
SHOWCAT macro 168
STEPCAT
required for dynamic allocation 49
STRNO parameter
DEFINE BCS 30
suballocated VSAM objects
data space 179
suballocation space
integrated catalog facility catalog 18
summary messages
DIAGNOSE command 119, 131—133
summary of index options 42, 190
SUPERLOCATE macro
return codes
CVOL 240
SUPERLOCATE statement 233
SYSCATnn member
alternate master catalog job stream 157
converting a master catalog 62
during IPL 44
SYSCTLG data set 237, 238, 239
SYSZRPLW.catname 30

T

terminal monitor program
APF authorization 31
test data set statistics
TESTCB 202
timestamp value 19
timestamps
VSAM catalog 179
TRACKS parameter 183
true name record 6
TRUENAME parameter
in DELETE command 90
TSO (time sharing option)
APF authorization 31
command mapping 233
defining user catalog 181
TTR
pointer 234

U

UCATDX statement 233
unavailable volume, removal from a nonrecoverable catalog 70
unavailable volume, removal from a recoverable catalog 70
uncataloging a non-VSAM data set
  coding example 267
  macro specification 267
  return codes 267
UNCATLG statement 233, 238
unique data space 179
  integrated catalog facility catalog 18
UNIQUE parameter
  defining a VSAM data set 179
unload and reload
  VSAM catalog 203
unusable catalog 212
updating a backup catalog 204
user catalog 181, 246
  identifying 27
using the alternate master catalog 43
using the DEFINE command 22
USVR (user-security-verification routine)
  password protection 37
utility programs
  data recovery 209

V

variable-length index entries 175
VCB (volume control block)
  format of 275
  use of 247
VERIFY command 21
  correct data set not properly closed 210
volume cleanup 87
volume control block pointer entry 274
volume index control entry 270
volume list
  definition 246
  use in catalog maintenance 246
volume ownership
  integrated catalog facility catalog 19
  VSAM catalogs 178
volume record
  information 177
volume recovery
  volumes with CRAs 215
  volumes without CRAs 214
volume serial number
  changing 199
volumes mounted
  BCS volumes required 24
  VVDS volumes required 24
VSAM (virtual storage access method)
  password protection 32
VSAM catalog
  cleanup 206
  copying 190
  data and space management 176
  DEFINE command 181
  information contained 177
  master 246
  order of search 246
  recoverable
    exporting 190
    importing 190
    space allocation 188
  secondary allocation amount 187
  structure 175, 178
  timestamps 179
  user 246
  volume ownership 178
VSAM conversion using CNVTCAT 56
  space requirements 57
VSAM data space
  VTOC 179
VSAM debug switches 228
VSAM messages 227
VSAM objects
  in data space 182
    VSAM catalogs only 182
  space allocation 183
VSAM SNAP dump facility 228
VSAM volume
  cleanup 205
  recovery function 215
VTOC (volume table of contents)
  BCS entry 96
  extent restrictions 19
  IEHLIST LISTVTOC 115
  integrated catalog facility catalog entries 19
  VSAM data space 179
VVCR (VSAM volume control record) 15, 16, 26
VVDS (VSAM volume data set)
  backup and recovery 96
  cells 151
  DEFINE NORECATALOG 26
  DEFINE RECATALOG 89, 93
  DELETE NOSCRATCH 93
  DELETE RECOVERY 93
  description 15
  explicit definition 26
  implicitly created 25
  recovery 101, 104, 110
  relationship to BCS 86
  reorganization 84
  self-describing volume record 15, 16, 26
  sharing 43
  space allocation 84
  space requirements 30
  volume mounted requirements 22
VVR (VSAM volume record)
  cells 18
  DELETE command 90

VVR (VSAM volume record) *(continued)*
information 16
number per data set 16

W

WORKFILE parameter
space requirements 224
worksheet
estimating VSAM catalog space 185

MVS/XA Catalog Administration Guide

GC26-4138-4

Reader's Comment Form

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should 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.

If you have applied any technical newsletters (TNLs) to this book, please list them here: _____

Chapter/Section _____

_____ Page No. _____

Comments:

Note: Staples can cause problems with automatic mail-sorting equipment. Please use pressure-sensitive or other gummed tape to seal this form.

If you want a reply, please complete the following information.

Name _____ Phone No. (_____) _____

Company _____

Address _____

_____

Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

GC26-4138-4

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Programming Publishing
P.O. Box 49023
San Jose, CA 95161-9023

Fold and tape | Please do not staple | Fold and tape

IBM®

MVS/XA Catalog Administration Guide

GC26-4138-4

Reader's Comment Form

This manual is part of a library that serves as a reference source for system analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should 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.

If you have applied any technical newsletters (TNLs) to this book, please list them here: _____

Chapter/Section _____

_____ Page No. _____

Comments:

Note: Staples can cause problems with automatic mail-sorting equipment. Please use pressure-sensitive or other gummed tape to seal this form.

If you want a reply, please complete the following information.

Name _____ Phone No. (_____) _____

Company _____

Address _____

_____

Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

GC26-4138-4

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Programming Publishing
P.O. Box 49023
San Jose, CA 95161-9023

Fold and tape | Please do not staple | Fold and tape

IBM®

MVS/Extended Architecture
Catalog Administration Guide

File Number S370-34

GC26-4138-04

Printed in U.S.A.