Archive Backup System for OpenVMS

Guide to Operations

Order Number: AA-QHD1N-TE

Software Version	Archive Backup System for OpenVMS Version 3.2A
Required Operating System	OpenVMS VAX Version 6.2, 7.1, 7.2 or 7.3 & OpenVMS Alpha Version 6.2, 7.1-2, 7.2-1 or 7.3
Required Software	Media and Device Management Services for OpenVMS Version 3.2A DECnet (Phase IV) or DECnet-Plus(Phase V) TCP/IP Services for OpenVMS

Compaq Computer Corporation
Houston, Texas

August 2001

Printed in the U.S.A.

Preface xix

1 What Is Archive Backup System for OpenVMS?

1.1 ABS Operational Environment 1-2
1.2 ABS Policy Objects 1-4
1.2.1 Save Request 1-4
1.2.2 Restore Request 1-4
1.2.3 Storage Policy 1-5
1.2.4 Environment Policy 1-6
1.3 ABS Catalogs 1-6
1.4 Archive File System 1-7
1.5 Backup Agent 1-7
1.6 Hierarchical System Management for OpenVMS Support 1-8
1.7 ABS Supports Stacker Configured Devices 1-8
1.8 ABS Provides Fast Tape Positioning 1-8
1.9 Scheduler Interface Options 1-9
1.10 ABS Interfaces 1-10

2 ABS Client-Server Technology

2.1 ABS OpenVMS Client-Server Configuration 2-1
2.2 ABS UNIX or NT Client Configuration 2-2

3 Customizing Your ABS Policy

3.1 Deciding What Data to Save 3-2
3.2 Deciding When to Save Data 3-3
3.3 Deciding Where to Save Data 3-4
3.3.1 MDMS Archive Type 3-4
3.3.2 Files–11 Archive Type 3-5
3.3.3 Customizing the Storage Policies Provided by ABS 3-5
3.4 Deciding How to Move Data 3-6
3.4.1 Customizing the Environment Policies Provided By ABS 3-7
3.4.2 Changing the Policy Engine Location 3-8

4 Data Safety

4.1 Central Security Domain 4-1
4.2 Backup Management Domains 4-3
4.2.1 Centralized Backup Management Domain 4-4
4.2.2 Distributed Backup Management Domain 4-5
4.2.3 Combined Backup Management Domain 4-6

5 Backup Strategies

5.1 How ABS Implements Its System Backup Strategy 5-1
5.1.1 System Backup Process 5-1
5.2 How ABS Implements Its User Backup Strategy 5-2
5.2.1 User Process Context 5-2
5.2.2 User Profile Process 5-2
5.2.3 User Backup Process 5-3
5.3 Differences Between System and User Backup Operations 5-4
5.4 Configuring ABS for OpenVMS Client Backup Operations 5-5

5.4.1 Creating ABS Policy Objects For OpenVMS Client System Backup Operations 5-5
5.4.2 Creating Save Requests for OpenVMS Client System Backup Operations 5-6
5.4.3 Creating ABS Policy Objects for OpenVMS Client User Backup Operations 5-7
5.4.4 Creating Save Requests for OpenVMS Client User Backup Operations 5-9
5.5 Configuring ABS for NT and UNIX Client Backup Operations 5-9
5.5.1 Creating ABS Policy Objects For NT and UNIX Client System Backup Operations 5-10
5.5.2 Creating Save Requests for NT and UNIX Client System Backup Operations 5-11
5.6 Oracle Rdb Databases and Storage Areas Backup Operations 5-11
5.6.1 Saving Individual Storage Areas 5-12
5.6.2 Catalog Entries 5-13
5.6.2.1 Oracle Rdb Database Catalog Entries: 5-13
5.6.2.2 Oracle Rdb Storage Area Catalog Entries: 5-13
5.6.3 Searching for Storage Areas in the Catalog 5-13
5.6.4 Restoring Storage Areas and Databases 5-14
5.7 Cataloging Copied Backup Savesets 5-14

6 Displaying ABS Graphical User Interface

6.1 Displaying ABS GUI On an OpenVMS System 6-1
6.2 Displaying ABS GUI on an NT System 6-2
6.3 Standard X Window for Motif Buttons 6-3

7 Creating Storage Policies

7.1 Using ABS Policy Worksheets 7-1
7.2 Requirements 7-2
7.3 Creating an ABS Storage Policy 7-2
7.4 Storage Policy Name 7-2
7.5 Save Data To 7-3
7.5.1 Tape Options 7-3
7.5.1.1 Media Type 7-3
7.5.1.2 Pool 7-3
7.5.1.3 Drives 7-4
7.5.1.4 Location 7-4
7.5.1.5 Criteria Under Which ABS Creates Volume Sets 7-4
7.5.1.6 Clear Volume Set List From Storage Policy 7-5
7.6 Retain Data For 7-5
7.7 Catalog and Execution Node 7-6
7.7.1 Selecting ABS Catalog 7-6
7.7.2 Selecting the Node of Execution 7-6
7.8 Number of Streams 7-6
7.9 Storage Policy Access Control 7-7
7.10 Submitting the Storage Policy 7-7

8 Creating Environment Policies

8.1 Using ABS Policy Worksheets 8-1
8.2 Requirements 8-1
8.3 Creating an ABS Environment Policy 8-2
8.4 Environment Policy Name 8-2
8.5 Save and Restore Environment Options 8-2

8.5.1 Who to Notify 8-2
8.5.1.1 How to Notify and Who to Notify 8-2
8.5.1.2 When to Notify 8-3
8.5.1.3 Type of Notification 8-3
8.5.2 Data Verification 8-3
8.5.3 Listing 8-4
8.5.4 Pre- and Post- Processing Commands 8-4
8.5.5 Original File 8-5
8.5.6 Retry Options 8-5
8.5.7 User Profile 8-5
8.5.8 Open Files 8-6
8.5.9 Tape Drives 8-6
8.5.10 Compression 8-6
8.5.11 Links Option 8-7
8.5.12 Span Filesystems 8-7
8.6 Environment Policy Access Control 8-7
8.7 Submitting the Environment Policy 8-8

9 Creating Save Requests

9.1 Save Request Name 9-1
9.2 What Data To Save 9-1
9.2.1 Save Request Restrictions 9-3
9.2.2 Pre- and Post- Processing Commands 9-4
9.2.3 Selection Criteria 9-6
9.2.4 Agent Qualifier 9-6
9.3 When to Save Data 9-6
9.3.1 Immediately Executing the Save Request 9-6
9.3.2 Repetitive Scheduling of Save Request 9-6
9.4 Where and How 9-9
9.5 Save Request Access Control 9-10
9.6 Submitting the Save Request 9-10

10 Creating Restore Requests

10.1 Restore Request Name 10-1
10.2 What Data To Restore 10-1
10.2.1 Restore Request Restrictions 10-3
10.2.2 Pre and Post- Processing Commands 10-5
10.2.3 Selection Criteria 10-6
10.2.4 Agent Qualifiers 10-6
10.3 When 10-7
10.4 Where and How 10-7
10.5 Restore To 10-7
10.6 Restore Request Access Controls 10-8
10.7 Submitting the Restore Request 10-9

11 Scheduling Requests

11.1 Setting the Scheduler Interface Option 11-1
11.2 Changing between Scheduler Interface Option 11-1
11.3 Scheduler Interface Option INT_QUEUE_MANAGER 11-2
11.4 Scheduler Interface Option EXT_QUEUE_MANAGER 11-2

11.5 Scheduler Interface Option EXT_SCHEDULER 11-3
11.6 Scheduler Interface Option DECSCHEDULER 11-3
11.7 Scheduler Interface Option NONE 11-4
11.8 Scheduler Interface Internals 11-4

12 Modifying and Deleting ABS Policies and Requests

12.1 Select Request or Policy 12-2

13 Looking Up Saved Data

13.1 Data to Lookup 13-1
13.1.1 File Type 13-1
13.1.2 Entering the Correct Lookup Syntax 13-1
13.1.3 Node of Original Data 13-2
13.1.4 Storage or Catalog Name 13-3
13.1.5 Archived Dates to Search 13-3
13.2 Submitting the Lookup Operation 13-4

14 Monitoring Job Status

15 Creating ABS Catalogs

15.1 Creating An ABS Catalog 15-1
15.1.1 Creating a BRIEF type catalog 15-2
15.1.2 Creating a FULL_RESTORE type catalog 15-3
15.1.3 Creating An SLS Type Catalog 15-3
15.1.4 Creating a VOLUME_SET type catalog 15-4
15.1.5 Creating a Catalog using Staging Operation 15-4
15.2 Showing a Catalog 15-5
15.3 Modifying a Catalog 15-5
15.4 Deleting a Catalog 15-5
15.5 Improving Catalog Performance 15-5
15.5.1 Converting ABS Catalogs 15-5
15.5.2 Moving Target Catalogs to a Different Disk 15-6
15.5.3 Moving Staging Catalog Entries 15-6
15.6 Sizes of Catalog Files 15-6
15.6.1 Technical Details 15-7
15.7 What size is the ABS catalog? 15-7

16 What is MDMS?

16.1 MDMS Objects 16-1
16.2 MDMS Interfaces 16-2

17 MDMS Configuration

17.1 The MDMS Management Domain 17-1
17.1.1 The MDMS Database 17-2
17.1.1.1 Database Performance 17-3
17.1.1.2 Database Safety 17-3
17.1.1.3 Moving the MDMS Database 17-4
17.1.2 The MDMS Process 17-5
17.1.2.1 Server Availability 17-5

17.1.2.2 The MDMS Account 17-6
17.1.3 The MDMS Start Up File 17-6
17.1.3.1 MDMS$DATABASE_SERVERS - Identifies Domain Database Servers 17-7
17.1.3.2 MDMS$LOGFILE_LOCATION 17-7
17.1.3.3 MDMS Shut Down and Start Up 17-8
17.1.4 Managing an MDMS Node 17-8
17.1.4.1 Defining a Node's Network Connection 17-8
17.1.4.2 Defining How the Node Functions in the Domain 17-8
17.1.4.3 Enabling Interprocess Communication 17-9
17.1.4.4 Describing the Node 17-9
17.1.4.5 Communicating with Operators 17-9
17.1.5 Managing Groups of MDMS Nodes 17-9
17.1.6 Managing the MDMS Domain 17-10
17.1.6.1 Domain Configuration Parameters 17-10
17.1.6.2 Domain Options for Controlling Rights to Use MDMS 17-11
17.1.6.3 Domain Default Volume Management Parameters 17-11
17.1.7 MDMS Domain Configuration Issues 17-12
17.1.7.1 Adding a Node to an Existing Configuration 17-12
17.1.7.2 Removing a node from an existing configuration 17-12
17.2 Configuring MDMS Drives, Jukeboxes and Locations 17-13
17.2.1 Configuring MDMS Drives 17-13
17.2.1.1 How to Describe an MDMS Drive 17-13
17.2.1.2 How to Control Access to an MDMS Drive 17-13
17.2.1.3 How to Configure an MDMS Drive for Operations 17-14
17.2.1.4 Determining Drive State 17-14
17.2.1.5 Adding and Removing Managed Drives 17-14
17.2.2 Configuring MDMS Jukeboxes 17-14
17.2.2.1 How to Describe an MDMS Jukebox 17-14
17.2.2.2 How to Control Access to an MDMS Jukebox 17-14
17.2.2.3 How to Configure an MDMS Jukebox for Operations. 17-15
17.2.2.4 Attribute for DCSC Jukeboxes 17-15
17.2.2.5 Attributes for MRD Jukeboxes 17-15
17.2.2.6 Determining Jukebox State 17-15
17.2.2.7 Magazines and Jukebox Topology 17-15
17.2.3 Summary of Drive and Jukebox Issues 17-17
17.2.3.1 Enabling MDMS to Automatically Respond to Drive and Jukebox Requests 17-17
17.2.3.2 Creating a Remote Drive and Jukebox Connection 17-17
17.2.3.3 How to Add a Drive to a Managed Jukebox 17-18
17.2.3.4 Temporarily Taking a Managed Device From Service 17-18
17.2.3.5 Changing the Names of Managed Devices 17-18
17.2.4 Locations for Volume Storage 17-19
17.3 Sample MDMS Configurations 17-20

18 Basic MDMS Operations

18.1 MDMS User Interfaces 18-1
18.1.1 Command Line Interface 18-1
18.1.1.1 Command Structure 18-1
18.1.1.2 Process Symbols and Logical Names for DCL Programming 18-1
18.1.1.3 Creating, Changing, and Deleting Object Records With the CLI 18-1
18.1.1.4 Add and Remove Attribute List Values With the CLI 18-2
18.1.1.5 Operational CLI Commands 18-2
18.1.1.6 Asynchronous Requests 18-3

18.1.2 Graphic User Interface 18-3
18.1.2.1 Object Oriented Tasks 18-3
18.1.2.2 Combined Tasks 18-4
18.2 Access Rights for MDMS Operations 18-5
18.2.1 Description of MDMS Rights 18-5
18.2.1.1 Low Level Rights 18-5
18.2.1.2 High Level Rights 18-5
18.2.2 Granting MDMS Rights 18-6
18.3 Creating, Modifying, and Deleting Object Records 18-8
18.3.1 Creating Object Records 18-8
18.3.1.1 Naming Objects 18-8
18.3.1.2 Differences Between the CLI and GUI for Naming Object Records 18-8
18.3.2 Inheritance on Creation 18-9
18.3.3 Referring to Non-Existent Objects 18-9
18.3.4 Rights for Creating Objects 18-9
18.3.5 Modifying Object Records 18-9
18.3.6 Protected Attributes 18-9
18.3.7 Rights for Modifying Objects 18-10
18.3.8 Deleting Object Records 18-10
18.3.9 Reviewing Managed Objects for References to Deleted Objects 18-10
18.3.10 Reviewing DCL Command Procedures for References to Deleted Objects 18-11
18.3.11 Rights for Deleting Objects 18-13

19 Connecting and Managing Remote Devices

19.1 The RDF Installation 19-1
19.2 Configuring RDF 19-1
19.3 Using RDF with MDMS 19-2
19.3.1 Starting Up and Shutting Down RDF Software 19-2
19.3.2 The RDSHOW Procedure 19-2
19.3.3 Command Overview 19-2
19.3.4 Showing Your Allocated Remote Devices 19-2
19.3.5 Showing Available Remote Devices on the Server Node 19-2
19.3.6 Showing All Remote Devices Allocated on the RDF Client Node 19-3
19.4 Monitoring and Tuning Network Performance 19-3
19.4.1 DECnet Phase IV 19-3
19.4.2 DECnet-Plus (Phase V) 19-4
19.4.3 Changing Network Parameters 19-4
19.4.4 Changing Network Parameters for DECnet (Phase IV) 19-4
19.4.5 Changing Network Parameters for DECnet-Plus(Phase V) 19-5
19.4.6 Resource Considerations 19-6
19.4.7 Controlling RDF's Effect on the Network 19-7
19.4.8 Surviving Network Failures 19-8
19.5 Controlling Access to RDF Resources 19-9
19.5.1 Allow Specific RDF Clients Access to All Remote Devices 19-9
19.5.2 Allow Specific RDF Clients Access to a Specific Remote Device 19-9
19.5.3 Deny Specific RDF Clients Access to All Remote Devices 19-9
19.5.4 Deny Specific RDF Clients Access to a Specific Remote Device 19-10
19.6 RDserver Inactivity Timer 19-10
19.7 RDF Error Messages 19-10

20 MDMS Management Operations

20.1 Managing Volumes 20-1
20.1.1 Volume Life Cycle 20-1
20.1.2 Volume States by Manual and Automatic Operations 20-2
20.1.2.1 Creating Volume Object Records 20-2
20.1.2.2 Initializing a Volume 20-3
20.1.2.3 Allocating a Volume 20-3
20.1.2.4 Holding a Volume 20-4
20.1.2.5 Freeing a Volume 20-4
20.1.2.6 Making a Volume Unavailable 20-4
20.1.3 Matching Volumes with Drives 20-4
20.1.4 Magazines for Volumes 20-4
20.1.5 Symbols for Volume Attributes 20-5
20.2 Managing Operations 20-5
20.2.1 Setting Up Operator Communication 20-6
20.2.1.1 Set OPCOM Classes by Node 20-6
20.2.1.2 Identify Operator Terminals 20-6
20.2.1.3 Enable Terminals for Communication 20-6
20.2.2 Activities Requiring Operator Support 20-6
20.3 Serving Clients of Managed Media 20-7
20.3.1 Maintaining a Supply of Volumes 20-7
20.3.1.1 Preparing Managed Volumes 20-7
20.3.2 Servicing a Stand Alone Drive 20-9
20.3.3 Servicing Jukeboxes 20-9
20.3.3.1 Inventory Operations 20-9
20.3.4 Managing Volume Pools 20-10
20.3.4.1 Volume Pool Authorization 20-11
20.3.4.2 Adding Volumes to a Volume Pool 20-11
20.3.4.3 Removing Volumes from a Volume Pool 20-11
20.3.4.4 Changing User Access to a Volume Pool 20-12
20.3.4.5 Deleting Volume Pools 20-12
20.3.5 Taking Volumes Out of Service 20-12
20.3.5.1 Temporary Volume Removal 20-12
20.3.5.2 Permanent Volume Removal 20-12
20.4 Rotating Volumes from Site to Site 20-13
20.4.1 Required Preparations for Volume Rotation 20-13
20.4.2 Sequence of Volume Rotation Events 20-13
20.5 Scheduled Activities 20-14
20.5.1 Logical Controlling Scheduled Activities 20-15
20.5.2 Job Names of Scheduled Activities 20-15
20.5.3 Log Files for Scheduled Activities 20-15
20.5.4 Notify Users When Volumes are Deallocated 20-16

21 MDMS High Level Tasks

21.1 Creating Jukeboxes, Drives, and Volumes 21-1
21.2 Deleting Jukeboxes, Drives, and Volumes 21-4
21.3 Rotating Volumes Between Sites 21-5
21.4 Servicing Jukeboxes Used for Backup Operations 21-7

A Preparing For Disaster Recovery

A.1 Efficiently Configuring Your System A-1
A.2 Preparing for Disaster Recovery A-2

A.3 Recovering ABS From A Disaster Situation A-4
A.4 Recovering ABS Client Nodes A-6

B ABS Time Formats

B.1 Start Time Format B-1
B.2 Explicit Interval B-2

C ABS Cleanup Utilities

C.1 Database Cleanup Utility C-1
C.1.1 Starting Up the Database Cleanup Utility C-1
C.1.2 Changing the Database Cleanup Utility Default Behavior C-1
C.1.3 Database Cleanup Utility Log File C-2
C.1.4 Shutting Down the Database Cleanup Utility C-2
C.2 Catalog Cleanup Utility C-2
C.2.1 Starting Up the Catalog Cleanup Utility C-2
C.2.2 Changing the Catalog Cleanup Utility Default Behavior C-2
C.2.3 Catalog Cleanup Utility Log File C-3
C.2.4 Shutting Down the Catalog Cleanup Utility C-3
C.2.4.1 Restarting the Catalog Cleanup Utility C-4
C.2.5 ABS Catalog Cleanup Utility Process C-4

D Log–n Backup Schedules

E ABS Worksheets

E.1 Storage Policy Worksheet E-1
E.2 Environment Policy Worksheet E-2
E.3 Save Request Worksheet E-3

F Troubleshooting

F.1 Logical Names Provide Additional Tracing F-1
F.2 Troubleshooting Assistance for NT Clients F-1
F.3 Verifying NT and UNIX Client Quotas F-2
F.4 Considerations for Saving Large Disks on UNIX and NT Clients F-2
F.5 Using The Same Volume Set For Multiple Types of ABS Clients F-4
F.6 ABS Log Files F-4
F.7 New Logical Name Added To Increase Stack Size On Alpha Systems F-5
F.8 Additional Error Messages F-5
F.9 Upgrading ABS F-5
F.10Logical To Assist with Server Connection Problems F-5
F.11AUDIT Flags in ABS$POLICY_CONFIG.DAT F-5
F.12Troubleshooting MDMS Related Problems F-5
F.13Information Required When Reporting Problems F-6

G ABS Error Messages

H MDMS Error Messages

I SLS and ABS Comparisons

J SLS To ABS Conversion

J.1 SLS To ABS Conversion J-1
J.2 Why Convert from SLS to ABS? J-1
J.2.1 Consolidated Policy Management J-2
J.2.2 More Intuitive Policy Organization J-2
J.2.3 Better Logging and Diagnostic Capabilities J-2
J.2.4 UNIX and NT Clients J-2
J.2.5 Automatic Full and Incremental Operations J-2
J.2.6 More versatile User requested Operations J-3
J.2.7 Disk Storage Classes J-3
J.3 SLS and ABS System Backup Policy Overview J-3
J.3.1 SLS Policy with ABS Equivalents J-3
J.3.1.1 System Backup Policy Configuration J-3
J.3.1.2 Defining Your System Backup Policy J-4
J.3.1.3 Restoring Data J-4
J.3.1.4 Media Management J-5
J.3.2 ABS Overview with SLS Equivalents J-5
J.3.2.1 Policy Configuration J-5
J.3.2.2 Storage Class J-6
J.3.2.3 Execution Environment J-7
J.3.2.4 Save Request J-9
J.3.2.5 Restore Request J-10
J.3.2.6 Catalog J-11
J.4 SLS and ABS Operation Overview J-11
J.4.1 Scheduling J-12
J.4.1.1 SBK Symbols for Scheduling J-12
J.4.1.2 ABS Scheduler Interface Options J-12
J.4.2 Types of Operations J-13
J.4.2.1 System Backups J-13
J.4.2.2 Full and Incremental Operations J-15
J.4.2.3 Selective Operations J-15
J.4.2.4 User Requested Operations J-16
J.4.3 Media and Device Management J-17
J.4.3.1 New Media Manager J-17
J.4.3.2 Volume Set Management J-18
J.4.3.3 Consistency of Volume and Drive Management J-18
J.4.4 Cataloging J-19
J.4.4.1 SLS History Sets J-19
J.4.4.2 ABS Catalogs J-19
J.4.4.3 Restoring data with ABS from SLS History Sets J-19
J.5 Conversion Process J-20
J.5.1 Steps for Conversion J-20
J.5.1.1 Convert the MDMS Database J-20
J.5.1.2 Determine your use of SLS J-20
J.5.1.3 Converting SLS System Backups to ABS J-21
J.5.1.4 Converting User Backup policy J-26
J.5.1.5 Monitor ABS Activity J-26
J.5.1.6 Restoring from SLS History Sets J-27
J.6 Conversion Utility Reference J-27
J.6.1 Command Syntax J-27
J.6.2 Output Command File naming and contents J-27
J.7 SBK Symbols in ABS Terminology J-28

J.8 ABS Policy Attributes in SBK Terminology J-31

K Differences Between MDMS Version 2 and MDMS Version 3

K.1 Comparing STORAGE and MDMS Commands K-1
K.2 MDMS V2 Forms Interface Options K-2
K.3 TAPESTART.COM Command Procedure K-4

L Sample Configuration of MDMS

L.1 Configuration Order L-1
L.1.1 Configuration Step 1 Example - Defining Locations L-2
L.1.2 Configuration Step 2 Example - Defining Media Type L-2
L.1.3 Configuration Step 3 Example - Defining Domain Attributes L-2
L.1.4 Configuration Step 4 Example - Defining MDMS Database Nodes L-3
L.1.5 Configuration Step 5 Example - Defining a Client Node L-5
L.1.6 Configuration Step 6 Example - Creating a Jukebox L-5
L.1.7 Configuration Step 7 Example - Defining a Drive L-5
L.1.8 Configuration Step 8 Example - Defining Pools L-7
L.1.9 Configuration Step 9 Example - Defining Volumes using the /VISION qualifier L-7

M Converting SLS/MDMS V2.X to MDMS V3

M.1 Operational Differences Between SLS/MDMS V2 & MDMS V3 . . . . . M-11
M.1.1 Architecture M-11
M.1.2 MDMS Interfaces M-12
M.1.3 Rights and Privileges M-13
M.1.4 The MDMS Domain M-13
M.1.5 Drives M-14
M.1.6 Jukeboxes M-15
M.1.7 Locations M-16
M.1.8 Media Types M-16
M.1.9 Magazines M-17
M.1.10 Nodes M-17
M.1.11 Groups M-18
M.1.12 Pools M-18
M.1.13 Volumes M-18
M.1.14 Remote Devices M-110
M.2 Converting SLS/MDMS V2.X Symbols and Database M-110
M.2.1 Executing the Conversion Command Procedure M-111
M.2.2 Resolving Conflicts During the Conversion M-111
M.3 Things to Look for After the Conversion M-114
M.4 Using SLS/MDMS V2.x Clients With the MDMS V3 Database M-118
M.4.1 Limited Support for SLS/MDMS V2 during Rolling Upgrade M-118
M.4.2 Upgrading the Domain to MDMS V3 . . . . . M-118
M.4.3 Reverting to SLS/MDMS V2 . . . . . M-119
M.4.4 Restrictions M-120
M.5 Convert from MDMS Version 3 to a V2.X Volume Database M-120

N Using ABS to Backup Oracle Databases

N.1 Example Oracle Database N-1

N.2 Backing up a Closed Database N-2
N.2.1 Creating ABS Environment and Storage Policies for a Closed Database Backup N-2
N.2.2 Creating ABS Save Requests for a Closed Database Backup N-4
N.3 Backing Up an Open Database N-5
N.3.1 Creating ABS Environment and Storage Policies for an Open Database Backup N-6
N.3.2 Creating ABS Save Requests for an Open Database Backup N-10

Table 2–1 Differences Between ABS OpenVMS and UNIX or NT Clients 2-3
Table 3–1 Deciding What Data To Save 3-3
Table 3–2 Archive Type 3-4
Table 3–3 Customizing ABS Provided Storage Policies 3-6
Table 3–4 Customizing ABS Provided Environment Policies 3-7
Table 5–1 System Backup Process 5-1
Table 5–2 User Backup Process 5-3
Table 5–3 Major Differences Between System and User Backup Operations 5-4
Table 5–4 Creating Storage and Environment Policies for OpenVMS Client System Backup Operations . 5-6
Table 5–5 Creating System Backup Save or Restore Requests For OpenVMS Client 5-7
Table 5–6 Creating Storage and Environment Policies for OpenVMS Client User Backup Operations . . . 5-8
Table 5–7 Creating Save Requests for OpenVMS Client User Backup Operations 5-9
Table 5–8 Creating Storage and Environment Policies for NT/UNIX Client System Backup Operations 5-10
Table 5–9 Creating Storage and Environment Policies for NT/UNIX Client System Backup Operations 5-11
Table 6–1 Displaying ABS GUI on an OpenVMS System 6-1
Table 6–2 Displaying the GUI On an NT System Using eXcursion and DCL Commands 6-2
Table 6–3 Displaying ABS GUI Using eXcursion Menu Options 6-2
Table 7–1 Creating an ABS Storage Policy 7-2
Table 7–2 Selecting Tape or Disk Storage 7-3
Table 7–3 Options to Save the Data 7-5
Table 7–4 Enabling Access Control to the Storage Policy 7-7
Table 8–1 Creating an ABS Environment Policy 8-2
Table 8–2 Selecting the Notification Options 8-3
Table 8–3 Enabling Access to an ABS Environment Policy 8-8
Table 9–1 Adding Disk or File Names To A Save Request 9-2
Table 9–2 Correctly Entering the Disk Name or File Name 9-2
Table 9–3 Enabling Access To An ABS Save Request 9-10
Table 10–1 Adding Disk or File Names To A Restore Request 10-1
Table 10–2 Entering The Correct Syntax For A Restore Request 10-2
Table 10–3 Enabling Access Control To A Restore Request 10-8
Table 12–1 Requirements for Modifying and Deleting Policies and Requests 12-1
Table 12–2 Modifying or Deleting an ABS Policy or Request 12-2
Table 13–1 Entering The Correct Syntax For A Lookup Operation 13-1
Table 13–2 Finding Saved Data By Date 13-3
Table 15–1 Creating an ABS Catalog For SLS Restores 15-4
Table 15–2 Moving Target Catalogs to a Different Disk 15-6
Table 16–1 MDMS Object Records and What they Manage 16-1
Table 17–1 MDMS Database Files and Their Contents 17-2
Table 17–2 How to Back Up the MDMS Database Files 17-3
Table 17–3 Processing MDMS Database Files for an Image Backup 17-4
Table 17–4 How to Move the MDMS Database 17-5
Table 17–5 MDMS$SYSTARTUP.COM Logical Assignments 17-6
Table 17–6 Network Node Names for MDMS$DATABASE_NODES 17-7
Table 17–7 Default Volume Management Parameters 17-11
Table 17–8 Adding a Node to an Existing Configuration 17-12
Table 17–9 Actions for Configuring Remote Drives 17-18
Table 17–10 Changing the Names of Managed Devices 17-18
Table 18–1 Operational CLI Commands 18-2
Table 18–2 Operational Actions With the GUI 18-4
Table 18–3 Reviewing and Setting MDMS Rights 18-6
Table 18–4 Low Level Rights 18-10
Table 18–5 Reviewing Managed Objects for References to Deleted Objects 18-10

Table 18–6 Reviewing DCL Commands for References to Deleted Objects 18-11
Table 19–1 How to Change Network Parameters 19-5
Table 20–1 MDMS Volume State Transitions 20-2
Table 20–2 Setting Up Operator Communication 20-6
Table 20–3 Operator Management Features 20-6
Table 20–4 Configuring MDMS to Service a Stand Alone Drive 20-9
Table 20–5 How to Create Volume Object Records with INVENTORY 20-10
Table 20–6 Sequence of Volume Rotation Events 20-13
Table 21–1 Creating Devices and Volumes 21-2
Table 21–2 Deleting Devices and Volumes 21-4
Table 21–3 Rotating Volumes Between Sites 21-6
Table 21–4 Servicing Jukeboxes 21-8
Table A–1 Disaster Recovery Tasks A-2
Table A–2 Recovering ABS A-4
Table B–1 Start Time Formats B-1
Table C–1 Shutting Down the Database Cleanup Utility C-2
Table C–2 Defining the Catalog Cleanup Utility Logical Names C-3
Table E–1 Storage Policy Worksheet E-1
Table E–2 Environment Policy Worksheet E-2
Table E–3 Save Request Worksheet E-3
Table F–1 Assigning a System Variable for NT Troubleshooting F-2
Table F–2 Modifying the Blocking Factor F-3
Table F–3 ABS Log Files F-4
Table I–1 Comparing SLS and ABS Backup Attributes I-1
Table J–1 DCL Symbols and ABS Equivalent J-3
Table J–2 Storage Class Parameter and SBK File Equivalent J-6
Table J–3 ABS and SBK Equivalent J-7
Table J–4 Save Request and SBK Equivalent J-9
Table J–5 Restore Request Parameter Information J-10
Table J–6 ABS Parameter and SLS Equivalent J-11
Table J–7 Storage Class Parameter J-23
Table J–8 Execution Environment Parameter J-24
Table J–9 SBK Symbols in ABS Terminology J-28
Table J–10 ABS Storage Classes and SLS SBK Equivalent J-32
Table J–11 ABS Execution Environment Parameter and SLS SBK Equivalent J-33
Table J–12 ABS Save Request Parameter and SLS SBK Equivalent J-34
Table K–1 Comparing MDMS Version 2 and Version 3 Commands K-1
Table K–2 Comparing MDMS V2 Forms and MDMS V3 Features K-2
Table K–3 Comparison of TAPESTART.COM to MDMS Version 3 Features. K-4
Table M–1 Volume Attributes M-18
Table M–2 Symbols in TAPESTART.COM M-112
Table M–3 Things to Look for After the Conversion M-114

Figure 1–1 ABS Operational Environment 1-3
Figure 1–2 ABS Save or Restore Request 1-5
Figure 1–3 ABS Catalogs 1-7
Figure 2–1 ABS OpenVMS Client-Server Configuration 2-1
Figure 2–2 ABS UNIX or NT Client Configuration 2-3
Figure 3–1 ABS Policy 3-2
Figure 3–2 Storage Policy/Archive Type Association 3-5
Figure 4–1 Central Security Domain on an OpenVMS Cluster 4-2
Figure 4–2 Centralized Backup Management Domain On An OpenVMS Cluster 4-4
Figure 4–3 Distributed Backup Management Domain on an OpenVMS Cluster 4-5
Figure 4–4 Combined Backup Management Domains on an OpenVMS Cluster 4-7
Figure 6–1 ABS Main Window 6-2
Figure 12–1 Modify or Delete Requests And Policies Window 12-2
Figure 17–1 The MDMS Domain 17-2
Figure 17–2 Groups in the MDMS Domain 17-10
Figure 17–3 Jukebox Topology 17-16
Figure 17–4 Magazines 17-17
Figure 17–5 Volume Locations 17-19
Figure 17–6 Named Locations 17-20
Figure 20–1 Volume States 20-1
Figure 20–2 Magazines 20-5
Figure 20–3 Pools and Volumes 20-11
Figure 21–1 Configuring Volumes and Drives 21-1
Figure 21–2 Volume Rotation 21-6
Figure 21–3 Magazine Placement 21-7
Figure A–1 Special Save Request A-4
Figure D–1 Log–n Backup Schedules D-1

Preface

Intended Audience

This document is intended for storage administrators who are experienced OpenVMS system managers. This document should be used in conjunction with the *Introduction to OpenVMS System Management* manual.

Conventions

The following conventions are used in this document:

Convention	Description
{}	In format command descriptions, braces indicate required elements.
[ ]	In format command descriptions, square brackets indicate optional elements of the command syntax. You can omit these elements if you wish to use the default responses.
boldface type	Boldface type in text indicates the first instance of a term defined in the Glossary or defined in text.
italic type	Italic type emphasizes important information, indicates variables, indicates complete titles of manuals, and indicates parameters for system information.
`Starting test ...`	This type font denotes system response, user input, and examples.
Ctrl/*x*	Hold down the key labels Ctrl (Control) and the specified key simultaneously (such as Ctrl/Z).
PF1 *x*	The key sequence PF1 *x* instructs you to press and release the PF1 key, and then press and release another key (indicated here by *x*).
n	A lowercase italic *n* denotes the generic use of a number. For example, 19*nn* indicates a four-digit number in which the last two digits are unknown.
x	A lowercase *x* denotes the generic use of a letter. For example, *xxx* indicates any combination of three alphabetic characters.

Related Products

The following related products may be mentioned in this document:

Product	Description
HSM	HSM refers to Hierarchical Storage Management for OpenVMS software.
MDMS	MDMS refers to Media and Device Management Services for OpenVMS software.
OpenVMS	OpenVMS refers to OpenVMS operating system.
SMF	SMF refers to Sequential Media Filesystem for OpenVMS software.
SLS	SLS refers to Storage Library System for OpenVMS software.

Associated Documents

The following documents are part of Archive Backup System for OpenVMS documentation set:

- *Archive Backup System for OpenVMS Installation Guide*
- *Archive Backup System for OpenVMS Guide to Operations*
- *Archive Backup System for OpenVMS Command Reference Guide*

Part I

ABS Operations

This part of *Archive Backup System for OpenVMS Guide to Operations* contains information about Archive Backup System for OpenVMS software.

1

What Is Archive Backup System for OpenVMS?

Archive Backup System for OpenVMS (ABS) is a software product that allows you to save and restore data in a heterogeneous environment. ABS provides you with the ability to perform anything from full system backup operations to user-requested or user-created backup operations. ABS ensures data safety and integrity by providing a secure environment for their save and restore operations.

All of ABS features are not available, if you only have an ABS-OMT license. The following lists the features that are not implemented with ABS-OMT license:

- Command Line Interface (CLI)
- Support for remote tape devices using RDF
- Support for OpenVMS clients
- Creation of storage policies
- Creation of environment policies
- Save request scheduling options except for the following:
  - On demand (ON_DEMAND)
  - Weekly full/daily incremental (DAILY_FULL_WEEKLY)
  - One time only (ONE_TIME_ONLY)
- Support of STK silos and DCSC jukeboxes
- Support for the following scheduling options:
  - POLYCENTER Scheduler V2.1b (DECscheduler) (DECSCHEDULER)
  - External scheduler (EXT_SCHEDULER)
  - External queue manager (EXT_QUEUE_MANAGER)

ABS enables you to implement a backup policy that allows you to save the data through automatic or repetitively scheduled save operations. It also enables you to save data randomly using a one-time-only save operation. ABS allows you to use different scheduler interface options to schedule requests. This feature allows you to customize the scheduling of the save or restore request to your system configuration.

Save and restore operations are accomplished by using two of the policy objects recognized by ABS, a save request and a restore request. These policy objects allow you to save data from online to either a offline volume or to another disk and, if necessary, allows you to restore that data to either its original location or to a different output location.

ABS uses a media manager called Media and Device Management Services (MDMS) for OpenVMS. The MDMS software is provided with ABS software and must be installed and configured

before installing ABS. Together, ABS and MDMS minimize the amount of user interaction required to locate saved data and to manage volumes and tape drives used for ABS save and restore operations.

ABS tracks the location of data when saved as a result of an ABS save request. This information is kept in an ABS catalog. Upon request, ABS accesses the catalog to locate or restore the data and coordinates the media management responsibilities with MDMS.

In this chapter:

This chapter provides information about the various components that comprises an ABS operational environment. The information includes the following items:

- ABS Policy Objects
- ABS Catalogs
- ABS Log Files
- Archive File System
- Backup Agent
- ABS Interfaces

1.1 ABS Operational Environment

ABS operational environment contains the following components:

- ABS policy objects - ABS policy objects define physical locations of saved data, the criteria under which save and restore requests are performed, and the save and restore requests themselves. The manner in which you implement these ABS policy objects results in your ABS policy.

  ABS policy objects are described in Section 1.2.

- ABS policy database - ABS database contains the definitions for all ABS policy objects and makes reasonable decisions based upon the configuration of your managed storage environment and your ABS policy definitions.

- ABS catalogs - ABS catalogs are the components of ABS software that contain the history information about ABS save requests. Catalogs consist of one or more databases that contain the records of data saved using ABS. Those records enable you to locate and restore data that was saved using ABS.

  ABS catalogs are described in Section 1.3.

- Archive file system - An archive file system is the component that provides a method of accessing the file system that stores ABS save sets. ABS save sets are stored on media that reside in an archive file system. ABS supports the following archive file systems:

  – MDMS

  – Files-11

  The archive file system is described in Section 1.4.

- Backup agent - A backup agent is the agent that performs the actual data movement operation. For OpenVMS systems, the backup agents are the OpenVMS BACKUP Utility and the RMU Backup Utility. For UNIX and NT clients, the supported backup agent is gtar (tape archiver). ABS uses gtar because most UNIX and NT systems support it.

Backup agents are described in Section 1.5.

- An optional 3rd party scheduler product. By default ABS uses the OpenVMS Queue Manager to schedule requests.
- Interfaces - ABS provides the following interfaces:
  - Graphical user interface (GUI)
  - Command line interface (CLI)

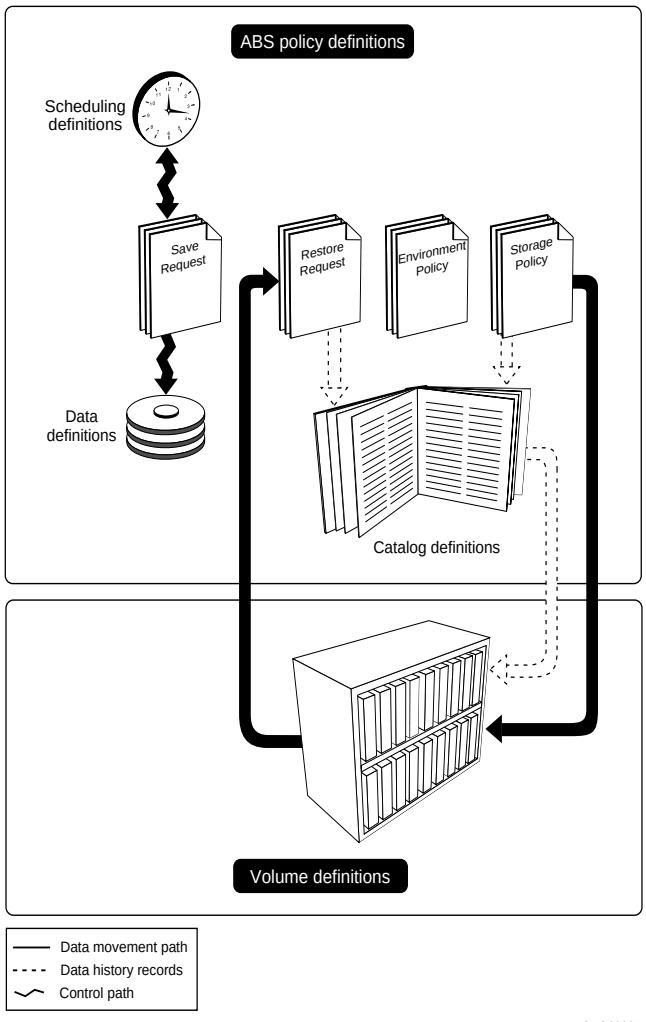
Note

Not available with the ABS-OMT license.

ABS interfaces are described in Section 1.10.

Figure 1–1 illustrates some of the components of ABS environment.

Figure 1–1 ABS Operational Environment



In Figure 1–1, the following items are also illustrated:

- Data definitions - These definitions allow you to specify the data (disk names, file names, or sets of disk names or file names) that you want to save or restore.
- Media definitions - Media definitions are defined in MDMS, ABS storage policies must specify the type of media to use for its save operations which you have previously defined in MDMS. Media definitions consist of either an MDMS media type name or an MDMS volume set name, or if you are saving to a disk (Files–11), the disk name and directory specification.

1.2 ABS Policy Objects

The following ABS policy objects are components of ABS software:

- Save requests
- Restore requests
- Storage policies
- Environment policies

The following sections describe ABS policy objects.

1.2.1 Save Request

A save request defines the data to be saved and executes upon immediate invocation or through an automatic, repetitive schedule. You can create save requests using either ABS GUI, or CLI.

A save request defines the following criteria:

- The data to back up
- The type of data to back up (VMS file, Oracle Rdb database, storage area, UNIX file or NT file)
- The type of save request (full, incremental, selective, scheduled, and so forth)
- When to save the data (start time and interval)
- Where to save the data (which storage policy to use)
- The length of time to keep the data (retention period or expiration date)
- The owner of the save request
- Who can access a save request (ensures data safety)
- What environment policy to use to execute the save request
- Whether to perform pre- or postprocessing commands

To meet your storage management requirements, you will need to create save requests that fulfill those requirements. Chapter 9, *Creating Save Requests* describes how to plan for and create a save request.

1.2.2 Restore Request

A restore request restores data from offline storage back to online storage. Restore requests are created using either the GUI or DCL and are executed immediately or at a specified time.

A restore request defines the following criteria:

- The data to restore
- The type of data to restore (VMS file, Oracle Rdb database, storage area, UNIX file or NT file)

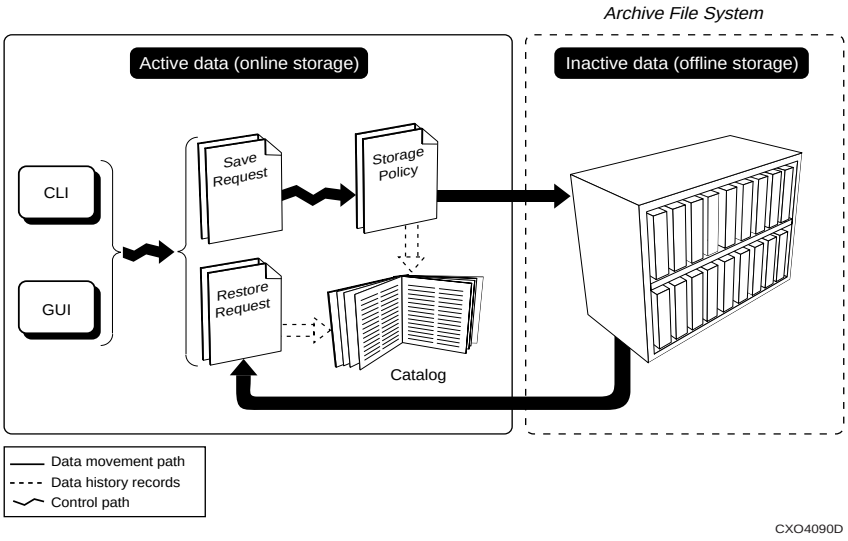
- The type of restore request (full, incremental, or selective)
- Where to restore the data (optional output location other than the original location)
- Where the data resides (storage policy)
- Who can access a restore request (ensures data security)
- What environment policy to use to execute the restore request
- Whether to perform pre- or postprocessing commands

More information:

To meet your storage management requirements, you will need to create restore requests that fulfill those requirements. Chapter 10, *Creating Restore Requests* describes how to plan for and create a restore request.

Figure 1–2 illustrates the path of a save or restore request.

Figure 1–2 ABS Save or Restore Request



A save or restore request is invoked through the GUI or through the CLI (DCL).

IF the request is a . . .	THEN the data is . . .
Save request	Saved from online storage to the storage policy. An ABS catalog records the location of the saved data.
Restore request	Restored back to online storage. ABS searches the catalog for the location of the data (storage policy), loads the appropriate volume, and restores the data.

1.2.3 Storage Policy

A storage policy defines the volumes (media type) and archive characteristics where you can safely store data. Each storage policy has a unique name, contains a set of archive characteristics, and is created and configured by users who have the proper privileges and access right identifiers (typically the storage administrator). A storage policy allows you to specify a simple storage policy name rather than a complicated set of characteristics.

Each storage policy defines the following:

- Which type of archive file system to use (MDMS or Files–11)
- If the archive file system is MDMS, the MDMS media type, tape pool, and location
- How long to keep the data stored in a particular storage policy (retention period or expiration date)
- Who is allowed to access the storage policy (ensures data safety)
- Who is allowed write data to and read data from the storage policy (ensures data safety)
- Which catalog contains the information about the data stored in the storage policy
- How long to use a volume set
- How many save or restore requests can be executed simultaneously

After the installation of ABS is complete, ABS provides storage policies. Section 3.3.3 lists the storage policies provided by ABS installation procedure.

1.2.4 Environment Policy

The environment policy defines the criteria under which save and restore requests are executed. The criteria defined in an environment policy include:

- Who to notify when a backup or restore operation has successfully completed (or failed)
- The number of drives to use for the save request
- Whether to use the environment policy for system backups, long term archives, and so forth
- The owner of the environment policy
- Who is allowed access to the environment policy (ensures data security)
- Default data safety checks to perform during backup or restore operations (such as Full, Redundancy, CRC, or a combination thereof)
- Whether to enable log and listing file.
- How often to retry the backup or restore operation before requiring user intervention
- Whether to perform job-wide pre- or post-processing commands
- UNIX compression, file system span, and symbolic link options
- Original object actions
- Locking options

After the installation of ABS is complete, ABS provides several environment policies. Section 3.4.1 lists the environment policies provided by ABS installation procedure.

1.3 ABS Catalogs

An ABS catalog is a database that contains history information about save requests and can be assigned to one or more storage policies. Each time a save request is initiated through a particular storage policy, the save request history is recorded in ABS catalog associated with the storage policy.

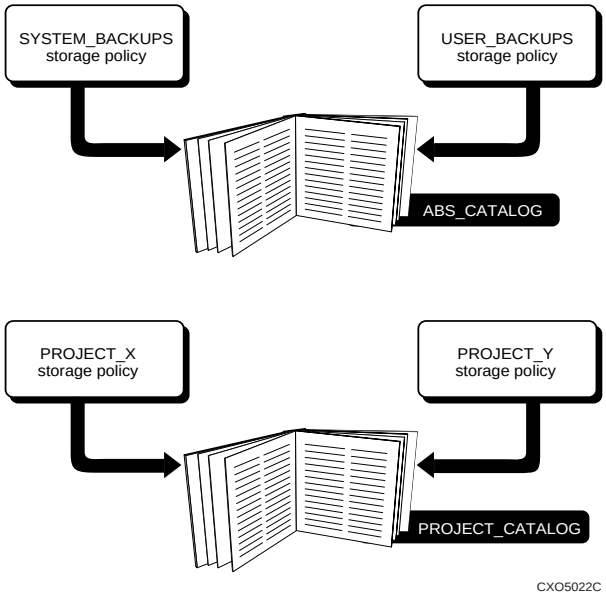
The information contained in an ABS catalog includes:

- The name of the data that was saved
- The date and time the data was saved
- The save set name where the data is located

- The original location of the data
- The owner of the data

Figure 1–3 shows the relationship between an ABS catalog and an ABS storage policy.

Figure 1–3 ABS Catalogs



After the installation of ABS is complete, ABS provides a default catalog named ABS_CATALOG. By default, this catalog is associated with all storage policies unless it is changed by the creator of the storage policy. All ABS catalogs, both the default catalog and user-created catalogs, support lookup and restore capabilities.

More information:

To meet your storage management requirements, you may need to create catalogs other than the one provided by ABS. Chapter 15, *Creating ABS Catalogs* describes how to plan for and create an ABS catalog.

1.4 Archive File System

An archive file system is the file system that stores ABS save sets. ABS enables you to specify which archive file system to use (storage policy) to store ABS save sets.

ABS supports the following archive file systems:

- MDMS - ABS is integrated with MDMS to support the management of removable media and devices.
- Files-11 - ABS supports backup and restore operations to and from disk devices or magneto optical media.

1.5 Backup Agent

ABS uses various backup agents to save and restore data. The backup agent is determined by the type of data, such as VMS files, Oracle Rdb databases, Oracle Rdb storage areas, UNIX files, or NT files. The backup agent is responsible for the actual data movement operation, while ABS is responsible for invoking the correct backup agent and recording the information about the save operation.

ABS supports the following backup agents:

- OpenVMS BACKUP Utility - For OpenVMS files, ABS uses the OpenVMS BACKUP Utility.
- RMU Backup Utility - For Oracle Rdb databases and storage areas, ABS uses the RMU Backup Utility.
- gtar - For UNIX and NT files, ABS uses gtar (tape archiver).

1.6 Hierarchical System Management for OpenVMS Support

ABS supports systems that have HSM installed. ABS and HSM can be configured together to permit HSM to perform shelving and /or preshelving of an OpenVMS file data, while nightly backups under ABS will copy only the file system metadata. This cooperative configuration is referred to as "Backup Via Shelving".

To facilitate this cooperative operation, the VMS BACKUP engine that ABS employs uses the following command qualifiers:

- /[NO]SHELVED - Specifying the negative form of this qualifier directs VMS BACKUP to omit shelved data from the ABS backup operation, and prevents unintended unshelving of shelved data. Specifying the qualifier in the positive form overrides the cooperative operation and causes the shelved data to be restored before ABS performs the backup operation. This may be useful in some circumstances, such as archiving.
- /[NO]PRESHELVED - Specifying the negative form of this qualifier directs VMS BACKUP to omit preshelved data from ABS backup operation, and removes unnecessary time and media overhead from the ABS backup operation. Specifying the qualifier in the positive form overrides the cooperative operation, and directs VMS BACKUP to make an additional copy of the preshelved data.

On systems where HSM is installed, ABS exhibits the following default behavior:

- For save requests that specify individual OpenVMS files, ABS unshelves the data in order to save it. So that ABS will ignore shelved data on these types of save requests, specify the /NOSHELVED qualifier and/or the /NOPRESHELVED qualifier in the Agent Options field.
- For save requests that specify an entire OpenVMS disk or one of the combined scheduling options (such as Weekly Full with Daily Incremental), ABS ignores shelved and preshelved data. No action is necessary to bypass this data.

1.7 ABS Supports Stacker Configured Devices

ABS supports stacker configured devices when it encounters free volumes already mounted on the drive. ABS will use the volume if it meets the media type criteria.

1.8 ABS Provides Fast Tape Positioning

ABS provides fast tape positioning so that the speed of positioning the tape is considerably faster. When positioning to the end of a volume that has a large amount of data stored on it, the difference in time will be significant from versions of ABS prior to V2.2. Positioning time for a restore request requires less time depending on the file location on the volume.

Note

When using a tape located on a remote drive (RDF device), fast tape positioning may not be used. To disable fast tape positioning on your server node or client node, define a logical name on each node:

```
$ DEFINE/SYSTEM ABS_NO_FAST_SKIP TRUE
```

If you are utilizing a tape drive that does not support the fast tape positioning, you may see errors such as:

```
ABS_SKIPMARKS_FAILED, Skip tape marks failure
```

In those cases, define the logical ABS_NO_FAST_SKIP on the node where the failures occur.

1.9 Scheduler Interface Options

ABS allows the use of different scheduler interfaces. By default ABS uses the programming interface to the OpenVMS Queue Manager to schedule save and restore requests. These are the scheduler interface options which can be used:

- INT_QUEUE_MANAGER (default) - uses a programming interface to OpenVMS Queue Manager
- EXT_QUEUE_MANAGER - uses DCL commands to interface with the OpenVMS Queue Manager by calling a command procedure
- EXT_SCHEDULER - uses DCL commands to interface with the 3rd party scheduler product by calling a command procedure
- DECSCHEDULER - uses a programming interface to POLYCENTER Scheduler (DECscheduler) V2.1b

Note

The internal queue manager scheduler interface is the only scheduler interface available with the ABS-OMT license.

The scheduler interface is invoked when a save or restore request is created, you can either start the request immediately (the only option for a restore request) or implement a repetitive schedule for save requests.

The scheduler interface is used to:

- Automate and manage ABS jobs that run repeatedly, such as ABS save requests.
- Separate all scheduling specifics such as the schedule interval, completion notification, and job dependencies from the job’s main task.
- Capture events through a logging system, so you can generate accounting and historical reports. This may vary depending on the scheduler interface.
- Execute all requests remotely as well as locally - invisible to the user.

For more information on the scheduling options, see Chapter 11, *Scheduling Requests*.

1.10 ABS Interfaces

ABS provides the following interfaces:

Interface	Description
ABS GUI	ABS provides a graphical user interface (GUI) for users whose systems support X Window System™ for Motif®. This interface displays a main menu and pull-down/pop-up windows that allow you to create, delete, set, or show ABS policy objects. This GUI allows you to perform all ABS functions and operation.
CLI	ABS also provides a Command Line Interface (CLI), which is the Digital Command Line (DCL) interface, for users who prefer this type of interface, or for users whose systems do not support X Window System for Motif or have access to a web server. **Note:** Not available with the ABS-OMT license.

2

ABS Client-Server Technology

ABS provides a client-server technology that ensures ABS policy database is secure and available only to authorized ABS clients. ABS server is the node or OpenVMS Cluster system where ABS policy engine and policy database resides.

ABS recognizes OpenVMS, UNIX, and NT clients. The following sections describe the differences between the functions and responsibilities of these types of ABS clients.

Note

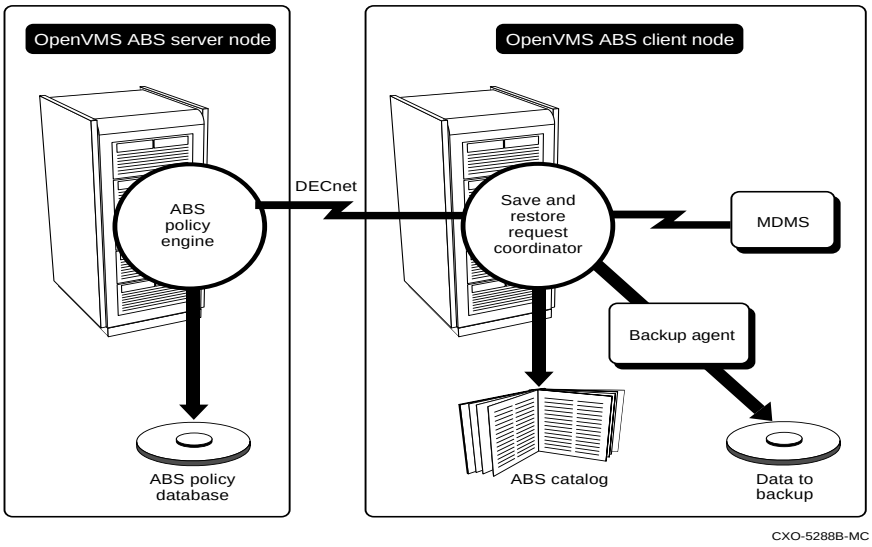
OpenVMS client support is not available with an ABS-OMT license.

2.1 ABS OpenVMS Client-Server Configuration

Figure 2–1 illustrates an OpenVMS client-server configuration as interpreted by ABS software. In this illustration, the following components are shown:

- An OpenVMS node that is ABS server node
- An OpenVMS node that is a ABS client.

Figure 2–1 ABS OpenVMS Client-Server Configuration



In Figure 2–1, ABS interprets OpenVMS client-server configuration as follows:

- ABS OpenVMS server node enables ABS OpenVMS client node to access ABS policy database through the DECnet communications software. ABS policy database resides on ABS server node.

- The save and restore request coordinator, ABS catalogs, and the backup agent (in this instance, the OpenVMS BACKUP Utility) all reside on ABS OpenVMS client node.
- Communications with the media management software (MDMS) are initiated by ABS OpenVMS client.

2.2 ABS UNIX or NT Client Configuration

Figure 2–2 illustrates a UNIX or NT client-server configuration. In this illustration, the following components are shown:

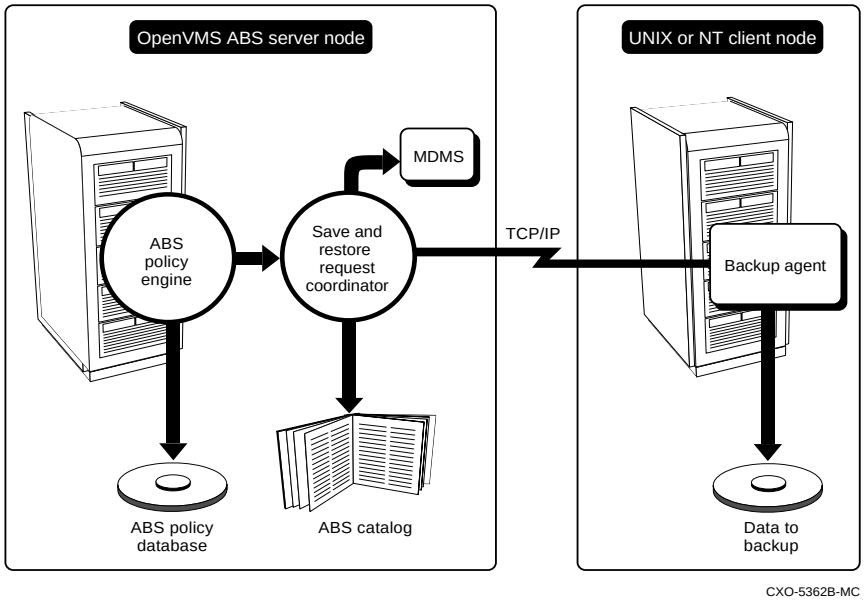
- An OpenVMS node that is ABS server
- A UNIX or NT node that is ABS client.

In Figure 2–2, ABS interprets the UNIX or NT client configuration as follows:

- ABS OpenVMS server node enables ABS UNIX or NT client node to access ABS policy database through the TCP/IP Services networking software. ABS policy database resides on ABS server node.
- The save and restore request coordinator and ABS catalogs reside on ABS OpenVMS server node.
- The backup agent (gtar) resides on the UNIX or NT client node.

  The save and restore coordinator on the OpenVMS server node coordinates the communications to the backup agent on ABS UNIX or NT client node. The save and restore coordinator controls the data movement between the UNIX or NT client node and MDMS, and it coordinates the catalog entries for the UNIX or NT data on ABS server node.
- Communications with the media management software (MDMS) for ABS UNIX or NT client are initiated by ABS OpenVMS server.
- ABS server node is also the node of execution. This node launches the save or restore operation.

Figure 2–2 ABS UNIX or NT Client Configuration



The main difference between a ABS UNIX or NT client and a ABS OpenVMS client is shown in Table 2–1.

Table 2–1 Differences Between ABS OpenVMS and UNIX or NT Clients

Type of ABS Client	ABS Server Node Responsibility	ABS Client Node Responsibility
OpenVMS	• Contains ABS policy database and policy engine	• Contains the save and restore coordinator • Contains ABS catalogs • Initiates communications with the MDMS software for itself • Contains the data to back up • Contains OpenVMS backup agent
UNIX or NT	• Contains ABS policy database and policy engine • Contains the save and restore coordinator • Contains ABS catalogs • Initiates communications with the MDMS software for the UNIX or NT client node.	• Contains the client-specific backup agent • Contains the data to back up

The components required to configure a UNIX or NT client system are a gtar executable file (provided with ABS software) and the TCP/IP Services networking software (pre-requisite for UNIX or NT client support). See *Archive Backup System for OpenVMS Installation Guide* for instructions about installing and configuring a UNIX or NT client system.

3

Customizing Your ABS Policy

There are several decisions you must make to customize Archive Backup System for OpenVMS (ABS) software so it fulfills your storage management requirements. Customizing ABS policy objects provided by the installation procedure or creating new ABS policy objects is the method of defining your ABS policy.

Note

Creating ABS policy objects is not available with ABS-OMT license. However, a default storage policy and environment policy is supplied that may be changed to meet your storage management requirements. The default policies are:

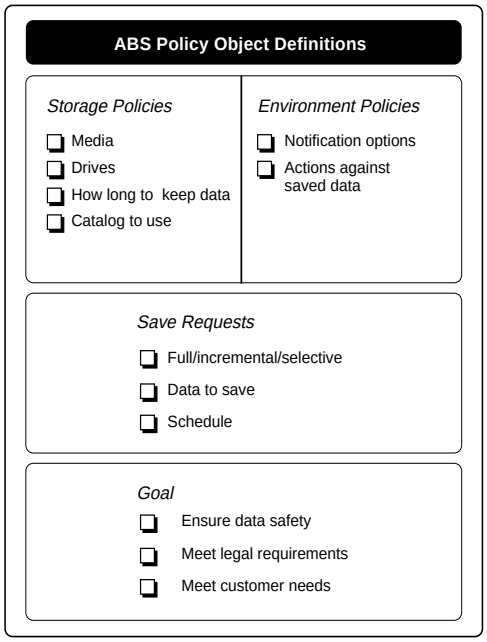
• OMT_BACKUPS - storage policy
• OMT_BACKUPS_ENV - environment policy

Consider the following items when planning your ABS policy:

- What data needs to be saved to ensure the stability and integrity of your enterprise?
- When and how often should that data be saved?
- Where do you want to safely store the saved data?
- How (in what environment) should that data be saved or restored?

To configure an ABS policy that meets your specific needs, weigh the considerations described in the following sections and then create or modify ABS policy objects that meet those needs. Figure 3–1 shows the options you can set on ABS policy objects that will implement an ABS policy that meets your storage management requirements.

Figure 3–1 ABS Policy



The following sections provide information about the decisions you must make to configure an ABS policy that meets your storage management requirements.

3.1 Deciding What Data to Save

Part of configuring your ABS policy is deciding your backup strategy. Save requests enable you to specify the data that you need to save, either to ensure data safety or to meet business requirements. You may need to create several different save requests to ensure complete implementation of your backup strategy.

Once your save requests are created, you can modify or delete those save requests (provided they meet deletion criteria) in order to maintain your ABS policy.

ABS supports the following types of save requests:

- Full - A full save request typically saves every instance of data on an entire disk. For OpenVMS disks, the include specification would be an OpenVMS disk name. For UNIX or NT disks, the include specification would be UNIX or NT path name. The type of save request would be full.
- Selective - A selective save request specifies only certain files on a disk, known as selected data. This is typically done for users who cannot create their own save requests, for projects that require saving its data, or for long-term archiving purposes.
- Incremental - An incremental save request saves only data that has been created or modified since the last full save request successfully completed. The include specification would be identical to the full save request. However, this type of save request is not recommended in ABS. A better option is to create a full save request and using one of the combined scheduling options provided by ABS. Chapter 9, *Creating Save Requests* describes these options.

Table 3–1 provides some guidelines for deciding which type of save requests you need to create to make sure that you are correctly implementing your backup strategy.

Table 3–1 Deciding What Data To Save

IF you want to make sure that...	THEN the include specification on the save request should contain ...	ABS Policy
All of your data is saved and recoverable	A disk name or multiple disk names (up to twentyfour)	A full save request ABS supports system backup operations and provides a variety of combined scheduling options.Section 9.3.2 describes the scheduling options available in ABS.
Data related to a particular project or group is saved and recoverable	A file name or multiple file names (up to twentyfour)	A selective save request ABS supports saving only selected data from one or more disks.
Your business policy allows users to create their own save and restore requests	A file name or multiple file names (up to twentyfour)	A user-created selective save request ABS enables you to support user-requested save and restore requests by providing a storage policy that is accessible to authorized users.
Data is retained for an extended period of time, typically to meet legal requirements	A disk or file name of the selected data that you want to archive. The save request must reference a storage policy specifically configured for retaining data for an extended period of time.	A selective save request ABS enables you to save data that meets your archiving requirements, and to delete the original data from online storage if desired. Chapter 7, *Creating Storage Policies* describes how to create a storage policy with the characteristics for long-term storage. Chapter 8, *Creating Environment Policies* describes how to create an environment policy that will delete the online data once the save request has successfully completed.

3.2 Deciding When to Save Data

Once you have decided what data you need to save, you must consider when and how often to save that data. ABS offers a variety of scheduling options. It also allows you to interrupt those schedules when necessary. Consider the following items before deciding when and how often to schedule save requests:

- When and how often you want to save data, such as daily, weekly, monthly, only during nonpeak user hours, and so forth.
- When you want to restrict save requests from executing, such as on holidays, plant closure days, or vacation days.
- When you want to limit save requests to execute only on specific days, such as only at the close of each quarter of business.

Each save request enables you to specify the start time and the interval at which you want to execute the save request. These options on a save request are Start Time and Schedule. This, along with other save request options, enables you to set up a backup strategy that fulfills your storage management needs.

Chapter 9, *Creating Save Requests* explains how to create save requests and Section 9.3.2 describes the scheduling options available for those save requests.

3.3 Deciding Where to Save Data

Storage policies define a set of characteristics for data that is saved using ABS. Storage policy characteristics include where (on which volumes) to store data, how long to store the data, which catalog to use for recording the location of data, and whether to enable or restrict access to the data by other users.

As the storage administrator, you can allow access to or restrict access from a storage policy, or you can create storage policies for the sole purpose of allowing users to create their own save and restore requests. The manner in which you allow access to a storage policy determines who can create save and restore requests using that storage policy.

A storage policy also controls volume selection. Each storage policy has an archive type associated with it to specify how a volume is located. An archive type includes the type of a volume (either tape or disk) that the storage policy uses.

ABS supports the following archive types:

- Media and Device Management Services for OpenVMS (MDMS)
- Files–11

Table 3–2 describes the archive types that ABS supports, the type of volumes supported by the archive type, and the storage policy association.

Table 3–2 Archive Type

Archive Type	Type of volume	Storage Policy Association
MDMS	Removable. Examples are TA90, TK85, TK70, and so forth.	Media type Pool Location Volume Set Name
Files–11	Fixed. Example is an OpenVMS disk.	Disk name and directory specification

Each site has its own reasons for using specific types of volumes to store saved data. For example, the type of volume that you choose for long-term storage may have different characteristics than the type of volume that you choose for short-term storage or disaster recovery storage.

The following sections describe the supported archive types and their association with a storage policy.

3.3.1 MDMS Archive Type

MDMS is an archive type that manages volumes and tape drives. When it is associated with a storage policy, MDMS uses volumes that are defined as follows:

- Media-Type - The media type is defined in MDMS. Media types are also associated with a specific drive or list of drives. ABS can use any of the media types defined in MDMS.

  Requirement:

If you select MDMS for the storage policy's archive type, you must specify a previously defined MDMS media type.

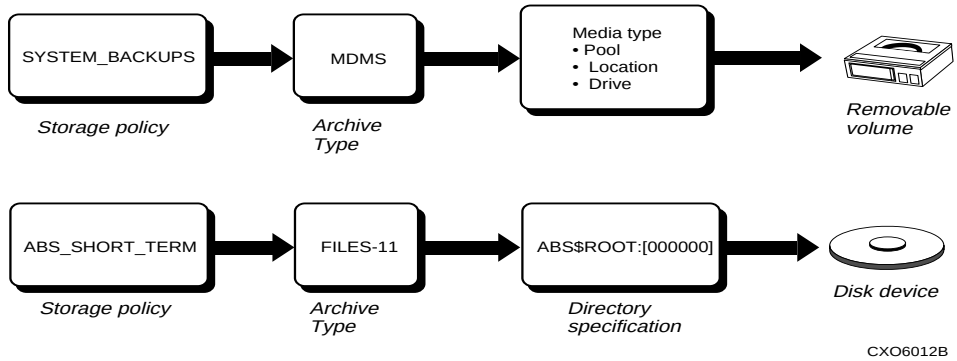
- Pool - A pool is a pool of volumes in a free state that are available for write operations. Pools are available to specific users (such as ABS) and must be previously created in MDMS.
- Drives - A drive can be a single drive or list of drives. However, MDMS associates the media type with a drive or list of drives. You would only use the Drive option if you want to change the drive or drives associated with the media type. It is recommended that you do not use the Drive option in ABS.
- Location - A location is a site specific location assigned to an MDMS volume.

3.3.2 Files–11 Archive Type

Files–11 is an archive type that allows you to store saved data to an OpenVMS File-11 structured online disk. Because disk storage is very costly, you must consider the long-term effects of using this type of archive type. The recommended and default archive type is MDMS.

Figure 3–2 shows how a storage policy is associated with an archive type, and how the archive type is associated with the type of volume that the storage policy uses.

Figure 3–2 Storage Policy/Archive Type Association



- SYSTEM_BACKUPS storage policy - This storage policy is provided by ABS and uses MDMS for its archive type. You can store data saved using ABS on volumes managed by MDMS. Volume selection is determined by the assignments to the Media Type.
- ABS_SHORT_TERM storage policy - This is an example of a user-created storage policy that uses FILES-11 as its archive type. Although this could be costly, you may want to store data for short period of time before deleting it, or use it as a temporary storage area.
- UNIX_BACKUP storage policy - This storage policy is provided by ABS for UNIX client save requests. This storage class uses MDMS for its archive type.

Note

Creation of storage policies is not available with an ABS-OMT license. The default policy is OMT_BACKUPS.

3.3.3 Customizing the Storage Policies Provided by ABS

After the installation of ABS has successfully completed, the following storage policies are resident on your system:

- ABS_ARCHIVE

- DISASTER_RECOVERY
- SYSTEM_BACKUPS
- USER_BACKUPS
- UNIX_BACKUPS

Note

The only storage policy available with the ABS-OMT license is OMT_BACKUPS.

With the exception of USER_BACKUPS, all the storage policies provide by ABS installation procedure have the same characteristics. To customize a storage policy, see Table 3–3. Table 3–3 uses ABS_ARCHIVE storage policy as an example.

Table 3–3 Customizing ABS Provided Storage Policies

Step	Action

1.	Make sure you have the proper ABS access rights identifiers assigned to your process so that you can modify the storage policy:
	`$ SET DEF SYS$SYSTEM`
	`$ RUN AUTHORIZE`
	
	`UAF>`
	`UAF> GRANT/ID ABS_CREATE_STORAGE_CLASS SMITH`
	`%UAF-I-GRANTMSG, identifier ABS_CREATE_STORAGE_CLASS granted to SMITH`
	
	`UAF> EXIT`
	`$`

2.	Make sure you have the proper access controls set on the storage policy object so that you can modify it:
	`$ ABS SET STORAGE ABS_ARCHIVE -`
	`_$ /ACCESS=(USER_ID=NODE01::SMITH,ACCESS="CONTROL+READ+WRITE+SET +DELETE+SHOW")`

3.	After installation, the only user who can access any of the storage policies (except USER_BACKUPS) is the user who installed ABS software. To allow other users access to the storage policy, add users and modify the access controls:
	`$ ABS SET STORAGE -`
	`_$ ABS_ARCHIVE/ACCESS=(USER_ID=*::*,ACCESS="READ+WRITE+SHOW")`

4.	If MDMS is the archive type, enter the media type to use for this storage policy.
5.	Modify any of the storage policy's other attributes (except its name) to reflect your storage management needs. Storage policy attributes are described in Chapter 7, *Creating Storage Policies*.

If you do not want to modify the storage policies provided by ABS, you may need to create additional storage policies to implement your ABS policy so that is meets your storage management needs. See Chapter 7, *Creating Storage Policies* for instructions about creating new storage policies.

3.4 Deciding How to Move Data

Up to this point, you have considered what data to save (save request), when to save the data (save request start time and schedule), and where you will store the data (storage policy). Now

you must consider how you want to execute save and restore requests, known as the environment policy

An environment policy defines the characteristics of the environment in which save and restore requests execute. Those characteristics include such things as who to notify, how many drives to use, listing and logging options, and so forth.

3.4.1 Customizing the Environment Policies Provided By ABS

After the installation of ABS has successfully completed, ABS provides the following environment policies:

- ABS_ARCHIVE_ENV
- DISASTER_RECOVERY_ENV
- SYSTEM_BACKUPS_ENV
- USER_BACKUPS_ENV
- UNIX_BACKUPS_ENV
- DEFAULT_ENV

Note

The only environment policy available with theABS-OMT license is OMT_BACKUPS_ENV.

With the exception of USER_BACKUPS_ENV, all of the environment policies have the same characteristics. To customize the environment policies to meet your storage management needs, see the instructions in Table 3–4.

Table 3–4 Customizing ABS Provided Environment Policies

Step	Action

1.	Make sure you have the proper ABS access rights identifiers assigned to your process so that you can modify the environment policy:
	`$ SET DEF SYS$SYSTEM`
	`$ RUN AUTHORIZE`
	
	`UAF>`
	`UAF> GRANT/ID ABS_CREATE_EXECUTION_ENV SMITH`
	`%UAF-I-GRANTMSG, identifier ABS_CREATE_EXECUTION_ENV granted to SMITH`
	
	`UAF> EXIT`
	`$`

2.	Make sure you have the proper access controls set on the environment policy object so that you can modify it:
	`$ ABS SET ENVIRONMENT ABS_ARCHIVE_ENV -`
	`_$ /ACCESS=(USER_ID=NODE01::SMITH,`
	`ACCESS="CONTROL+READ+WRITE+SET+DELETE+SHOW")`

3.	After installation, the only user who can access any of the environment policies (except USER_BACKUPS_ENV) is the user who installed ABS software. To allow other users access to the environment policy, add users and modify the access controls:
	`$ ABS SET ENVIRONMENT -`
	`_$ ABS_ARCHIVE_ENV/ACCESS=(USER_ID=*::*,ACCESS="READ+WRITE+SHOW"`

Table 3–4 Customizing ABS Provided Environment Policies

Step	Action
4.	Modify any of the environment policy's attributes (except its name) to reflect your storage management needs. Environment policy attributes are described in Chapter 8, *Creating Environment Policies*.

If you do not want to modify the environment policies provided by ABS, you may need to create additional ones to meet your storage management needs. See Chapter 8, *Creating Environment Policies* for instructions about creating new environment policies.

3.4.2 Changing the Policy Engine Location

When you install ABS, you are asked for a list of the nodes where the policy engine will run. This list is put into the ABS$SYSTEM:ABS$POLICY_CONFIG.DAT file. There is a separate line for each nodename.

Note

With the ABS-OMT license, the policy engine will only run on the node with the ABS-OMT license. However, you must have the node in the ABS$SYSTEM:ABS$POLICY_CONFIG.DAT so the policy engine will start.

If you later decide to remove one of the nodenames, you must edit this file and remove the line for that node (i.e. node1::):

```
ABS$POLICY_ENGINE_LOCATION = node1::
```

Be sure that there is at least one line specifying the ABS$POLICY_ENGINE_LOCATION in this file or ABS will not run.

If you wish to add another policy engine node, you may also add a line to this file. But, you must be sure that the policy engine image (ABS$SYSTEM:ABS$POLICY_ENGINE.EXE) is available on that node.

4

Data Safety

The information in this chapter describes how to configure a secure operating environment for ABS. It also provides example scenarios for setting up central security domains and backup management domains, the related ABS policy objects, and the access controls to set to allow access to ABS policy objects.

Note

OpenVMS client support is not available with the ABS-OMT license.

When considering data safety, you must make the following decisions:

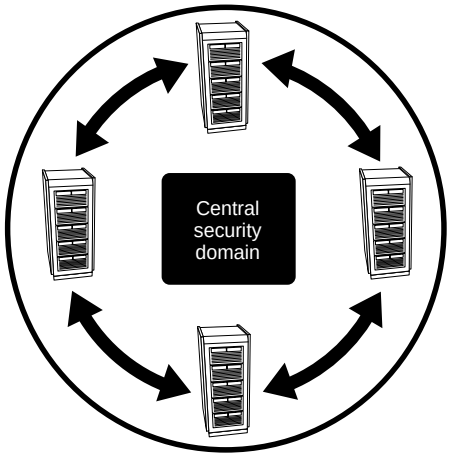
- Which node or OpenVMS Cluster™ system will contain ABS server software - This system becomes the central security domain for ABS software.
- Will you install more than one ABS server - You can place ABS server software on a single node or OpenVMS Cluster system, or you can place multiple ABS servers across several nodes or OpenVMS Cluster systems. These separate ABS servers are not cooperative among each other, but they can share the same MDMS database for media management purposes.
- Where will you allow control of ABS data movement operations - Any node or OpenVMS Cluster system that is allowed to create ABS save requests is considered a backup management domain. Backup management domains can be centralized, or can be distributed among several systems.

4.1 Central Security Domain

A central security domain is the node or OpenVMS Cluster system where ABS server software is installed, and where ABS policy database resides. After the installation of ABS, all ABS policy objects are located in ABS policy database. The central security domain controls creating, modifying, and deleting all ABS policy objects, especially storage and environment policies.

Depending upon your business needs, you can choose to have either a single central security domain or multiple central security domains.

Figure 4–1 Central Security Domain on an OpenVMS Cluster



In this configuration:

- ABS server software is installed on a OpenVMS Cluster system, this OpenVMS Cluster system becomes the central security domain and contains ABS policy database.
- This OpenVMS Cluster is managed by one storage administrator who is responsible for:
  - Ensuring that only authorized users can create, modify, or delete ABS policy objects.
  - Preventing nonprivileged users from accessing data for which they are not authorized to access.

Assumptions

In a distributed environment, ABS assumes that the systems on which it executes are, as a whole, reasonably secure. This means that only trusted backup management personnel (such as the storage administrator or an authorized operator) with a direct need are authorized with the following items:

- Physical access to OpenVMS systems and volumes
- Elevated privileges that enable granting themselves access rights required by ABS, such as ABS_BYPASS, ABS_CREATE_STORAGE_CLASS, ABS_CREATE_EXECUTION_ENV, and ABS_LOOKUP_ALL. See *Archive Backup System for OpenVMS Installation Guide* for descriptions about ABS access right identifiers.
- Elevated privileges that enable them to modify ABS server's database access protection sets (APS).

User Backup Restrictions

ABS imposes the following restrictions for user backup operations:

- A nonprivileged user cannot save or restore data other than his own.
- A nonprivileged user can save or restore his own data, given the user has been granted the appropriate access controls on ABS storage and environment policies.
- The storage administrator must authorize what operations a nonprivileged user can perform. The storage administrator is responsible for creating environment policies that specify whether nonprivileged users can:

  - Save data under their own user profile (uses an environment policy that specifies <REQUESTER> or the user's own name)
  - Save data under another user profile (uses an environment policy that specifies ABS or some other user)
  - Restore data under their own user profile, whether the data was saved by the user or as a result of a system backup operation (uses an environment policy that specifies <REQUESTER> or the user's own name)
  - Restore data under another user profile (uses an environment policy that specifies <REQUESTER> or the user's own name)
  - A user profile must not be propagated onto different nodes. That is, a user-created save request must be executed on the same node or OpenVMS Cluster where the save request is submitted.
- The pre- and post- processing commands must be executed in the context of the requester.
- The request's log file must be accessible to the requester.

Note

This may not be possible, depending upon how scheduler in use opens the log file. If the log file cannot be made accessible, copy the log file to the user's SYS$LOGIN directory if possible.

- User cannot own their own archive resources (volume sets, magazines, and so forth). All archive resources must be owned by ABS account and managed by ABS on behalf of the users.
- Executing a save or restore request in the requester's context does not preclude the type of save or restore operation. That is, a user save request may be a full, selective, or incremental, and user restore request of these types are allowed as well.

Note

The ability of a user to execute any ABS backup or restore operation depends upon the user's ability to access the online data. For example, unless a user has access to a disk, the user would not be able to create a full save or restore request.

- The performance of a save or restore request executed in the context of the user should be equivalent to the performance of the same operation executed in the context of ABS account.
- The number of simultaneous user save requests is constrained by the maximum number of simultaneous write operations on the storage policy. The number of simultaneous restore requests is constrained by the number of compatible drives.

4.2 Backup Management Domains

A backup management domain is ABS's concept of backup management control. Any node or OpenVMS Cluster system that can create ABS save requests is considered a backup management domain. These systems have ABS client software installed. You can restrict the control of backup management to a single backup management domain, or you can enable control to several backup management domains. Typically, each backup management domain is controlled by one storage administrator.

ABS conceptually categorizes the following backup management domains:

- Centralized backup management domain

- Distributed backup management domain
- Combined backup management domain

The following sections describe how to configure central security domains and backup management domains to create a secure environment for ABS operations. Each section contains a scenario that details a hardware configuration, the placement of the central security domain, associated ABS policy objects, and the security controls that should be set on those policy objects to minimize the potential for unauthorized access to data.

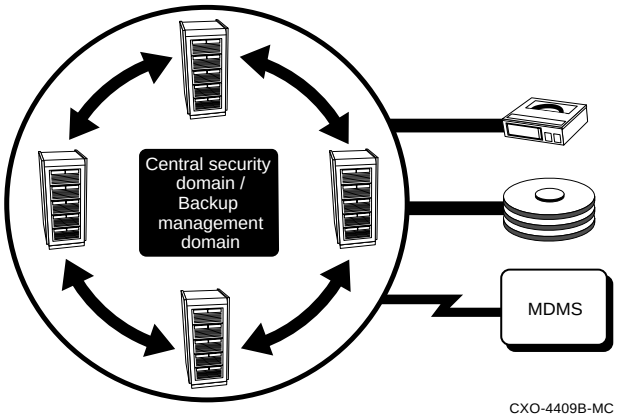
4.2.1 Centralized Backup Management Domain

A centralized backup management domain allows backup policy and schedules to be created only within a single backup management domain. This backup management domain is also the central security domain. A centralized backup management domain allows you to control your ABS policy in one, centralized location. However, this backup management domain cannot create save or restore requests for remote nodes or OpenVMS Cluster systems outside the backup management domain.

Scenario: Centralized Backup Management Domain on a Single Node or OpenVMS Cluster System

In this scenario, storage and environment policies are created and maintained within the central security domain. A single backup management domain that is confined to the central security domain, can create, modify, and delete save requests. This represents ABS backup management control in its simplest form.

Figure 4–2 Centralized Backup Management Domain On An OpenVMS Cluster



In this configuration:

- ABS server software resides on a OpenVMS Cluster system, this OpenVMS Cluster system is the central security domain.
- The backup management domain is confined to the central security domain, restricting backup management control to one storage administrator.
- Save and restore request objects can be created only within the backup management domain.
- The data movement path is contained within the backup management domain.
- Access to all storage and environment policies is confined to the central security domain, and READ/WRITE/SHOW access may be granted to users with OpenVMS OPERATOR or SYSTEM privileges.

Example:

Both storage and environment policies must have the following access controls set:

```
$ ABS SET POLICY_NAME -
_$ /ACCESS_CONTROL=(USER=NODE_NAME::USER_NAME,ACCESS="READ+WRITE+SHOW")
```

Where:

- *POLICY_NAME* is the name of the storage or environment policy
- *NODENAME* is the node name of the central security domain
- *USERNAME* is the name of the user who can access ABS policy. In most cases, it is ABS.

4.2.2 Distributed Backup Management Domain

The distributed backup management domain allows ABS policy objects to be stored in a central location (the central security domain), but responsibility for backup management control (creating and modifying save requests) is distributed among remote nodes or remote OpenVMS Cluster systems.

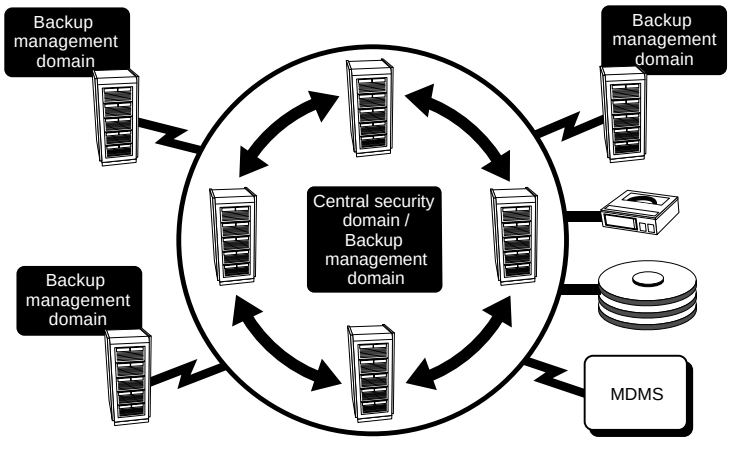
This type of backup management control distributes backup management responsibility to backup management domains outside the central security domain.

Scenario: Distributed Backup Management Domain

In this scenario, storage and environment policies are defined and maintained only from the central security domain. Save and restore requests can be created from the central security domain or from the backup management domain. These remote backup management domains are connected to the central security domain through the DECnet software. The save and restore requests reside on the central security domain (in ABS policy database), but the save or restore operation is executed on the remote system.

Restriction:
A backup management domain cannot create a save request for another backup management domain. A backup management domain can only create a save request for itself.

Figure 4–3 Distributed Backup Management Domain on an OpenVMS Cluster



In this configuration:

- ABS server software is located on a OpenVMS Cluster system, this OpenVMS Cluster system contains ABS policy database and is the central security domain.
- Backup management domains are distributed among the three remote systems, distributing backup management among more than one storage administrator if so desired. Save and restore requests can be created on either the central security domain or on the backup management domains.
- The data movement path can be local to or remote from the backup management domain. For example, the remote backup management domain can use either a local backup device or a remote backup device.
- Some or all storage and environment policies have READ/WRITE/SHOW access granted to users with OpenVMS OPERATOR or SYSTEM privileges from the remote systems.

  Examples:

  – Storage and environment policies that can be accessed by all the remote backup management domains but only by a specific user must have the access controls set as shown in the following example:

  ```
  $ ABS SET POLICY_NAME -
  _$ /ACCESS_CONTROL=(USER=*::USER_NAME,ACCESS="READ+WRITE+SHOW")
  ```

  – Storage and environment policies intended for use by a specific remote system (in this example, backup management domain NODE_A) must have the following access controls set:

  ```
  $ ABS SET POLICY_NAME -
  _$ /ACCESS_CONTROL=(USER=CLSTRA::USER_NAME,ACCESS="READ+WRITE+SHOW")
  $ ABS SET POLICY_NAME -
  _$ /ACCESS_CONTROL=(USER=NODEA::USER_NAME,ACCESS="READ+WRITE+SHOW")
  ```

- Storage and environment policies intended for use by more than one remote system (backup management domain NODEB and NODEC, but not NODEA) must have the following access controls set:

  ```
  $ ABS SET POLICY_NAME -
  _$ /ACCESS_CONTROL=(USER=NODEB:USER_NAME,ACCESS="READ+WRITE+SHOW")
  $ ABS SET POLICY_NAME -
  _$ /ACCESS_CONTROL=(USER=NODEAC:USER_NAME,ACCESS="READ+WRITE+SHOW")
  ```

  Where:

  – *POLICY_NAME* is the name of the storage or environment policy

  – *NODENAME* is the node name of the central security domain

  – *USER_NAME* is the name of the user who can access ABS policy. In most cases, it is ABS.

4.2.3 Combined Backup Management Domain

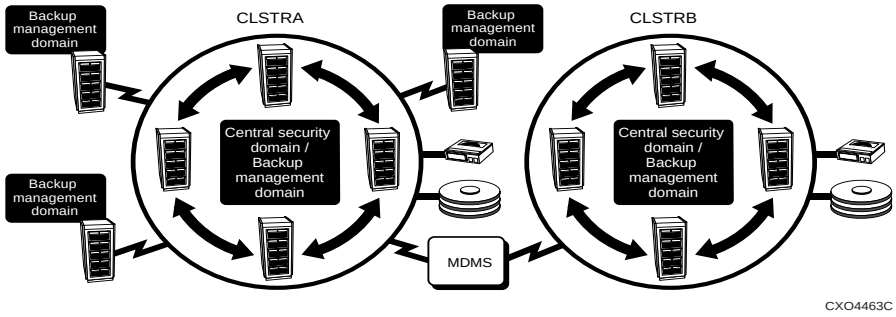
Depending upon your business needs, you may choose to combine backup management control strategies. You can install ABS server software on multiple nodes or OpenVMS Cluster systems, creating multiple central security domains.

Scenario: Combined Backup Management Domain

In this scenario, a network has two OpenVMS Cluster systems that have ABS server software installed. Because of the two ABS servers, there are two different central security domains on the network.

The OpenVMS Cluster system (CLSTRA) has several backup management domains while the other OpenVMS Cluster system (CLSTRB) has only one backup management domain, the central security domain.

Figure 4–4 Combined Backup Management Domains on an OpenVMS Cluster



In this configuration:

- ABS server software resides on two separate OpenVMS Cluster systems in the network. Both of these systems become central security domains.
- Both central security domains share one common MDMS volume database.
- Each of the central security domains has the MDMS client software installed.
- CLSTRA - The first central security domain (CLSTRA) has several remote backup management domains that distribute backup management control to multiple storage administrators. Storage and environment policies may enable READ/WRITE/SHOW access to some or all of the remote systems.

Examples:

1. On CLSTRA, storage and environment policies that can be accessed by all the backup management domains but a specific user must have the following access controls set:

```
$ ABS SET POLICY_NAME -
_$ /ACCESS_CONTROL=(USER=*::USER_NAME,ACCESS="READ+WRITE+SHOW")
```

2. Storage and environment policies intended for use by only one remote system (NODEA) must have the following access controls set:

```
$ ABS SET POLICY_NAME -
_$ /ACCESS_CONTROL=(USER=NODEA::USER_NAME,ACCESS="READ+WRITE+SHOW")
```

3. Storage and environment policies intended for use by more than one remote system (NODEB and NODEC, but not NODEA) must have the following access controls set:

```
$ ABS SET POLICY_NAME -
_$ /ACCESS_CONTROL=(USER=NODEB::USER_NAME,ACCESS="READ+WRITE+SHOW")

$ ABS SET POLICY_NAME -
_$ /ACCESS_CONTROL=(USER=NODEAC::USER_NAME,ACCESS="READ+WRITE+SHOW")
```

4. CLSTRB - This OpenVMS Cluster system is a central security domain that has only one, single backup management domain. This backup management domain gives backup management control to only one storage administrator. All storage and environment policies within the central security domain must enable READ/WRITE/SHOW access from the central security domain node.

   Example:

   Storage and environment policies on the central security domain (CLSTRB) must have the following access controls set:

   ```
   $ ABS SET POLICY_NAME -
   _$ /ACCESS_CONTROL=(USER=CLSTRB::USER_NAME,ACCESS="READ+WRITE+SHOW")
   ```

5

Backup Strategies

ABS's strategy for protecting data is to provide the most common type of backup strategies that customers require, system backups and user backups.

- System backups are typically the most common type of backups that customers implement to protect their data. These backups are typically done on a daily basis to ensure that the data is recoverable when necessary.
- User backups are provided for the customer who wants to enable individual users to back up their own data as needed. ABS provides the ability to meet both backup strategies.

The information in this chapter describes how ABS defines its system and user backup strategies.

Note

The following features are not available with the ABS-OMT license:

• Creation of storage and environment policies
• Support for OpenVMS clients

OMT_BACKUPS storage policy and OMT_BACKUPS_ENV environment policy are the only policies available with the ABS-OMT license.

5.1 How ABS Implements Its System Backup Strategy

To implement a system backup strategy, ABS provides policy objects that define the characteristics required for system backup operations. These ABS policy objects are the SYSTEM_BACKUPS storage policy and the SYSTEM_BACKUPS_ENV environment policy.

When a save request is created that uses the SYSTEM_BACKUPS storage policy, ABS creates volume sets (according to the consolidation criteria) that are owned by ABS and places the data specified by the save request on those volumes.

5.1.1 System Backup Process

Table 5–1 describes how system backup operations are implemented using ABS

Table 5–1 System Backup Process

Stage	Action
1.	ABS provides the default storage policy named SYSTEM_BACKUPS. To this storage policy, you must assign the media type that you want to use for system backup operations. In relation, the type of volume is associated with a drive or list of drives compatible with the volume. This association is done by defining read-write and read-only media types supported for each drive object.

Table 5–1 System Backup Process

2.	ABS provides the default environment policy named SYSTEM_BACKUP_ENV. This environment policy's user profile specifies ABS as the user. All save requests that use this environment run only in the context of ABS.
3.	ABS writes the data to an ABS save set on a volume set owned by ABS. Only users with privileged accounts can access the data on those volumes.
4.	A privileged user creates a save request using SYSTEM_BACKUPS storage policy and SYSTEM_BACKUPS_ENV environment policy (or equivalent user-created storage and environment policies).
5.	ABS determines whether to allow the save request to execute by checking the access controls enabled on the storage and environment policies, as well as by checking the user assigned to the user profile on the environment policy.
6.	When a privileged user creates a save request, ABS creates subprocesses in the context of ABS.
7.	The save request is executed in a job under the ABS account, regardless of the user who created the request.
8.	ABS creates catalog entries for the saved data. The owner information is parsed from the backup agent's output listing, so the current process context does not matter.
9.	Process context is set back to ABS when media management services are needed. This ensures appropriate access checks in the media manager (MDMS) are passed.
10.	Process context is set back to ABS when all save and restore requests have completed.

5.2 How ABS Implements Its User Backup Strategy

To implement a user backup strategy, ABS provides storage and environment policies that define the characteristics that enable users to create their own save and restore requests. These default policy objects are the USER_BACKUPS storage policy and the USER_BACKUPS_ENV environment policy.

5.2.1 User Process Context

ABS enables a storage administrator to create a save request in the context of a specific user, but without granting that user the privileges necessary to execute the save request (such as BYPASS or CMKRNL).

Create an environment that specifies a user profile, and add an additional privilege mask that grants the necessary privileges required to execute the save request. This restricts the user to executing save requests using only that environment policy.

5.2.2 User Profile Process

When a save or restore request executes in an environment policy that has a user profile assigned to it, ABS creates the subprocess where the request executes. This subprocess contains the user name, UIC, privileges, and access right identifiers of the user who is specified in the user profile.

The privileges associated with the subprocess defaults to those identified in the UAF record for the user. However, ABS also allows an additional privilege mask to be specified in the user profile, and places these additional privileges in the authorized privilege mask of the subprocess.

Therefore, if the subprocess executes a SET PROC/PRIV command (in one of the PROLOGUE commands) specifying any additional privileges, it will be granted.

Note

When you create a user profile for an environment policy, users with READ or WRITE access to the environment policy can submit save requests that execute in the context of the user specified in the user profile. Therefore, access control to an environment policy that has a user profile assigned to it must be carefully configured and controlled.

5.2.3 User Backup Process

Table 5–2 describes the process used by ABS for user backup operations.

Table 5–2 User Backup Process

Stage	Action
1.	ABS provides the default storage policy named USER_BACKUPS. To this storage policy you must assign the media type that you want to use for user-created save and restore requests. In relation, the media type is associated with a drive or list of drives compatible with the media type. This association is done by defining read-write and read-only media types supported for each drive object.
2.	ABS provides an environment policy named USER_BACKUP_ENV. This environment policy's user profile specifies <REQUESTER> as the user. Access control is set to *::*, meaning all nodes and all users can access the USER_BACKUPS_ENV environment policy. All save requests that use this environment policy run in the context of the requesting process.
3.	A nonprivileged user creates a save or restore request using USER_BACKUPS storage policy and USER_BACKUPS_ENV environment policy (or equivalent user-created storage and environment policies).
4.	A nonprivileged user must have WRITE access control enabled on the storage policy to create a save request and READ access control enabled to the storage policy to create a restore request.
5.	A nonprivileged user must have SHOW access control enabled on the environment policy. This allows the requester to the environment policy, but the access controls enabled on the storage policy determines what type of operations can be performed (save or restore request).

Table 5–2 User Backup Process

6.	ABS determines whether to perform a user backup operation by checking the access controls enabled on the environment policy as well as by checking the user assigned to the user profile: • If the user profile for the environment policy specifies a user name of <REQUESTER>, ABS uses the requester's account information. • If the user profile for the environment policy specifies a user name (such as ABS), the user creating the save or restore request must have additional access controls enabled on the environment policy that allows him to emulate the user when submitting save or restore request: – To emulate another user during a backup operation, the requester must have WRITE access to the environment. – To emulate the user during a restore operation, the requester must have READ access to the environment. – If the requester does not have either READ or WRITE access to the environment, ABS uses the requester's default profile. – If the requester has READ/WRITE access to the environment and if a "*" is specified for the node name in the user profile of the execution environment, the save can only be run on the node it is created. **Note**: This implementation provides the ability to set up a default profile for a storage policy with a privileged user specified. If end-users are not granted WRITE or READ access to the environment policy, they can save or restore data to or from the storage policy using their own profile. The user's profile is obtained when SHOW access control is enabled on the environment policy.
7.	When a user creates a save request, ABS creates subprocesses in the context of the requester.
8.	The save request is executed as a job under ABS account, regardless of the user who created the request.
9.	ABS creates catalog entries for the saved data. The owner information is parsed from the backup agent's output listing, so the current process context does not matter.
10.	Process context is set back to ABS when media management services are needed. This ensures appropriate access checks in the media manager (MDMS) are passed.
11.	ABS writes the data to an ABS save set on a volume set owned by ABS. Only users with privileged accounts or the original requester can access the data on those volumes.
12.	Process context is set back to ABS when all save or restore requests have completed.

5.3 Differences Between System and User Backup Operations

Table 5–3 shows the major differences between a system and user backup operation.

Table 5–3 Major Differences Between System and User Backup Operations

System Backup Operation	User Backup Operation
The context of the user process submitting the save request is set to ABS.	The context of the user process submitting the save request is set to the original requester (the user creating the save request).

Table 5–3 Major Differences Between System and User Backup Operations

Storage Policy = SYSTEM_BACKUPS	Storage Policy = USER_BACKUPS[a]
Environment Policy = SYSTEM_BACKUPS_ENV	Environment Policy = USER_BACKUPS_ENV[b]
SYSTEM_BACKUPS_ENV / PROFILE=(CLUSTER=*,NODE=*,USER-NAME=ABS)	USER_BACKUPS_ENV /PROFILE=(CLUSTER=*,NODE=*,USER-NAME=<REQUESTER>)
User must be a privileged user	Can be a nonprivileged user

a. The storage policy named USER_BACKUPS uses the environment policy named USER_BACKUPS_ENV.
b. The environment policy named USER_BACKUPS_ENV specifies a user profile, and the user profile specifies the user as <REQUESTER>. This causes the save or restore request to execute in the context of the user who creates the request.

Note

If an environment policy specifies an explicit user, the save or restore request is executed in the context of the user assigned in the user profile. Therefore, access control to any environment policy that specifies a user profile needs to be carefully configured and controlled. Section explains the details of submitting save requests in the context of another user's process.

5.4 Configuring ABS for OpenVMS Client Backup Operations

To configure ABS policy objects to be able to perform OpenVMS client system and user backups, review the following sections.

5.4.1 Creating ABS Policy Objects For OpenVMS Client System Backup Operations

You can modify storage and environment policies provided by ABS as described in the other chapters or you can create additional storage and environment polices for OpenVMS client system backup operations.

The following list describes how the access controls must be set on the storage and environment policies intended for OpenVMS client system backup operations:

- On the storage policy, enable all access controls to ABS
- On the environment policy:
  - Specify a user profile with the username of ABS
  - Enable all access controls to ABS

```
$ ABS SET ENVIRONMENT_POLICY/ACCESS_CONTROL= -
_$ (USER=*::ABS,ACCESS="READ+WRITE+SHOW+SET+DELETE+CONTROL")
```

- To create system save and restore requests from ABS OpenVMS server for ABS OpenVMS client system, or to create system save and restore requests on ABS OpenVMS client system, you must first create (or modify) storage and environment policies to meet those needs.

The examples in show you how to create ABS storage and environment policies that enable system backup operations for an ABS client node.

Table 5–4 Creating Storage and Environment Policies for OpenVMS Client System Backup Operations

Step	Action

1.	Create (or modify) a storage policy that allows access to ABS OpenVMS client node. For example, create a storage policy named CLIENTNODE_SYSTEM_BACKUPS:
	`ABS> CREATE STORAGE_CLASS CLIENTNODE_SYSTEM_BACKUPS -`
	`_ABS>/ACCESS=(USER_ID=*::ABS,ACCESS="READ+WRITE+SET+SHOW+CON-`
	`TROL+DELETE")`
	Restrictions:
	• You can only create (or modify) a storage policy on ABS server node (central security domain).
	• The storage policy node of execution must be ABS OpenVMS server node (SVNODE::). By default, ABS obtains the node on which you create the storage policy. Because you can only create a storage policy on ABS server node, the current node (server node) becomes the node of execution.
	• The creating account must have ABS_CREATE_STORAGE_CLASS access rights identifier granted. See Chapter 3 in *Archive Backup System for OpenVMS Installation Guide* for instructions about granting access rights identifiers.
	The storage policy must reference the archive file system resources (MDMS or FILES–11) that are available to ABS OpenVMS client node. It is possible for ABS OpenVMS client node to utilize MDMS to store data on a remote device (such as on ABS server node). See Chapter 19, *Connecting and Managing Remote Devices* for information about configuring remote tape drives.

2.	Create (or modify) a corresponding environment that also allows access to ABS OpenVMS client node:
	`ABS> CREATE ENVIRONMENT CL_SYSTEM_BACKUPS_ENV -`
	`_ABS> /ACCESS=(USER=*::ABS,ACCESS="READ+WRITE+SET+`
	`SHOW+CONTROL+DELETE")/PROFILE=(CLUSTER=*,NODE=*,USERNAME=ABS)`
	Restrictions:
	• You can only create (or modify) an environment policy on ABS server node.
	• The account that is creating or modifying the environment policy must have ABS_CREATE_EXECUTION_ENV access rights identifier granted. See *Archive Backup System for OpenVMS Installation Guide* for instructions about granting access rights identifiers.
	Because the environment policy CLIENTNODE_SYSTEM_BACKUPS_ENV specifies a user profile with the user name ABS, only system backup operations in the context of ABS are allowed.

5.4.2 Creating Save Requests for OpenVMS Client System Backup Operations

You can create system save requests for an OpenVMS client node from either ABS OpenVMS server node or from ABS OpenVMS client node. Table 5–5 describes how to do both.

Note

Do not combine a full disk save with an incremental disk save to create a single save set on OpenVMS systems.

Table 5–5 Creating System Backup Save or Restore Requests For OpenVMS Client

Step	Action

1.	**From ABS Server Node**:
	
	To create a system save or restore request for an OpenVMS client node (CLNODE) from ABS server node, on ABS server node (SVNODE) create a request that uses the storage policy named CLIENTNODE_SYSTEM_BACKUPS and the environment policy named CLIENTNODE_SYSTEM_BACKUPS_ENV. Specify ABS OpenVMS client node as the source node (where the data to save resides):
	
	`$ ABS SAVE/STORAGE_CLASS=CLIENTNODE_SYSTEM_BACKUPS/FULL`
	`_$ DISK$USER1:/SOURCE_NODE=CLNODE::`
	
	Result:
	The save or restore operation is executed on the remote client node CLNODE.
	
	Requirement:
	To create a save or restore request that executes on the OpenVMS client node, the creating account on ABS server node must have ABS_CREATE_REMOTE_JOBS access rights granted.

2.	**From the OpenVMS Client Node**:
	
	To create or submit a system save or restore request for an OpenVMS ABS client from ABS OpenVMS client node, on the client node create a request that uses the storage policy named CLIENTNODE_SYSTEM_BACKUPS and the environment policy named CLIENTNODE_SYSTEM_BACKUPS_ENV. Specify OpenVMS client node as the source node for (where the data resides):
	
	`$ ABS SAVE/STORAGE=CLIENTNODE_SYSTEM_BACKUPS/FULL -`
	`_$ DISK$USER1:/SOURCE_NODE=CLNODE::`
	
	Result:
	The save or restore request is executed on the remote client node CLNODE.

5.4.3 Creating ABS Policy Objects for OpenVMS Client User Backup Operations

You can modify existing storage and environment policies, or you can create other storage and environment policies for OpenVMS client user backup operations.

The following list describes how the access controls must be set on the storage and environment policies intended for user backup operations:

- On the storage policy:
  - Enable SHOW access to users who are allowed to access the storage policy
  - Enable WRITE access to users who are allowed to create save requests
  - Enable READ access to users who are allowed to create restore requests
- On the environment policy:
  - Specify a user profile with the username of <REQUESTER> (for all users) or the username of an individual user

– Enable SHOW access to all users for the <REQUESTER> user profile, or to a specific user for an individual user profile

The user must have WRITE access to the storage policy to create a save request, and READ access to the storage policy to create a restore request. Thus, setting SHOW access on the environment policy allows the requester to use this default profile, but the access on the storage policy determines what type of operations the user can perform.

To create user save and restore requests for ABS OpenVMS client system, you must first create (or modify) storage and environment policies on ABS server system that meet those needs.

In Table 5–6, the examples shows you how to create ABS storage and environment policies so that a nonprivileged user can create a save or restore request for an OpenVMS client.

Table 5–6 Creating Storage and Environment Policies for OpenVMS Client User Backup Operations

Step	Action

1.	Create (or modify) a storage policy for user backup operations. For example, you could name the storage policy CLIENTNODE_USER_BACKUPS. This storage policy allows access to user SMITH from the node named CLNODE. In this example, user SMITH would be allowed to create save and restore requests using the following storage policy:
	`ABS> CREATE STORAGE_CLASS CLIENTNODE_USER_BACKUPS -`
	`_ABS> /ACCESS=(USER_ID=CLNODE::SMITH,ACCESS="READ+WRITE+SET+SHOW+`
	`DELETE+CONTROL")`
	
	Restrictions:
	• You can only create (or modify) a storage policy on ABS server node (central security domain).
	• The node of execution for the storage policy must be ABS OpenVMS server node (SVNODE::). By default, ABS obtains the node from which you create the storage policy. Because you can only create a storage policy on ABS server node, the current node (server node) becomes the node of execution.
	• The creating account must have ABS_CREATE_STORAGE_CLASS access rights identifier granted. See *Archive Backup System for OpenVMS Installation Guide* for instructions about granting access rights identifiers.

2.	Enter ABS SHOW STORAGE_ClASS command of CLIENTNODE_USER_BACKUPS to see the access controls: `ABS SHOW STORAGE_CLASS CLIENTNODE_USER_BACKUPS/BRIEF` Access Right -- CLNODE::SMITH Access Granted -- READ, WRITE, SET, SHOW, DELETE, CONTROL

3.	Make sure the storage policy uses the media type that is available to ABS OpenVMS client node. It is possible for ABS OpenVMS client node to utilize MDMS to store data on a remote device (such as on ABS server node):
	`$ ABS CREATE STORAGE_CLASS`
	`CLIENTNODE_USER_BACKUPS/TYPE_OF_MEDIA=TK85`
	
	See Chapter 19, *Connecting and Managing Remote Devices* for information about configuring remote tape drives.

Table 5–6 Creating Storage and Environment Policies for OpenVMS Client User Backup Operations

4.	Create a corresponding environment policy that allows access to the user SMITH:
	`ABS> CREATE ENVIRONMENT CLIENTNODE_USER_BACKUPS_ENV -`
	`_ABS> /ACCESS=(USER_ID=CLNODE::SMITH,ACCESS="READ+WRITE+SET+SHOW+ -`
	`_ABS> CONTROL+DELETE")/PROFILE=(CLUSTER=*,NODE=CLNODE,USER-NAME=SMITH)`
	
	Because the environment CLIENTNODE_USER_BACKUPS_ENV specifies a user profile with the user name SMITH and the node name CLNODE::, ABS performs user backup operations in the context of the user SMITH on node CLNODE::.
	
	Note:
	If the node CLNODE:: is a part of an OpenVMS Cluster, you can specify the OpenVMS Cluster alias name instead of an asterisk (*).
	
	Restrictions:
	• You can only create (or modify) an environment on ABS server node (central security domain).
	• The creating account must have ABS_CREATE_EXECUTION_ENV access rights identifier granted. See *Archive Backup System for OpenVMS Installation Guide* for instructions about granting access rights identifiers.

5.4.4 Creating Save Requests for OpenVMS Client User Backup Operations

You can create save requests for the OpenVMS ABS client node only from the client node itself. Table 5–7, describes how to do this:

Table 5–7 Creating Save Requests for OpenVMS Client User Backup Operations

Step	Action

1.	**From ABS Client Node**:
	Create a save request on OpenVMS client node (CLNODE::) that uses the storage policy named CLIENTNODE_USER_BACKUPS and the environment policy named CLIENTNODE_USER_BACKUPS_ENV. Specify the source node for the save or restore request as OpenVMS client node (node where the data resides).
	`ABS> SAVE/STORAGE_CLASS=CLIENTNODE_USER_BACKUPS -`
	`_ABS> DISK$USER1:[SMITH]*.*/SOURCE_NODE=CLNODE::`
	
	Result:
	The save request is executed on the remote client node CLNODE.

5.5 Configuring ABS for NT and UNIX Client Backup Operations

To configure ABS policy objects to be able to perform NT or UNIX client system backup operations, review the following sections.

Note

You may have problems saving files greater than 2 GB on Unix and Windows NT systems. On Unix systems, please refer to the documentation provided with the gtar you are using. The gtar version that we supply for Windows NT and the source that we provide for Unix do not accommodate files greater than 2 GB in size.

5.5.1 Creating ABS Policy Objects For NT and UNIX Client System Backup Operations

You can modify storage and environment policies provided by ABS as described in other chapters or you can create additional storage and environment polices for NT or UNIX client system backup operations.

The following list describes how the access controls must be set on the storage and environment policies intended for OpenVMS client system backup operations:

- On the storage policy, enable all access controls to ABS
- On the environment policy:
  - Specify a user profile with the username of ABS
  - Enable all access controls to ABS

```
$ ABS SET ENVIRONMENT_POLICY -
_$ /ACCESS_CONTROL=(USER=*::ABS,ACCESS="READ+WRITE+SHOW+SET+DELETE+
CONTROL")
```

- To create system save and restore requests from ABS OpenVMS server for ABS OpenVMS client system, or to create system save and restore requests on ABS NT or UNIX client system, you must first create (or modify) storage and environment policies to that meet those needs.

The examples in Table 5–8 show you how to create ABS storage and environment policies that enable system backup operations for an ABS NT or UNIX client node.

Table 5–8 Creating Storage and Environment Policies for NT/UNIX Client System Backup Operations

Step	Action

1.	Create (or modify) a storage policy that allows access to ABS NT or UNIX client node. For example, create a storage policy named NTNODE_SYSTEM_BACKUPS:
	`ABS> CREATE STORAGE_CLASS NTNODE_SYSTEM_BACKUPS -`
	`_ABS> /ACCESS=(USER_ID=*::ABS,ACCESS="READ+WRITE+SET+SHOW+`
	`CONTROL+DELETE")`
	
	Restrictions:
	• You can only create (or modify) a storage policy on ABS OpenVMS server node (central security domain).
	• The storage policy node of execution must be an ABS server node. By default, ABS obtains the node on which you create the storage policy. Because you can only create a storage policy on ABS server node, the current node (server node) is the default.
	• The creating account must have ABS_CREATE_STORAGE_CLASS access rights identifier granted. See *Archive Backup System for OpenVMS Installation Guide* for instructions about granting access rights identifiers.

Table 5–8 Creating Storage and Environment Policies for NT/UNIX Client System Backup Operations

2.	Create (or modify) a corresponding environment policy that also allows access to ABS NT or UNIX client node:
	`ABS> CREATE ENVIRONMENT NTNODE_SYSTEM_BACKUPS_ENV -`
	`_ABS> /ACCESS_CONTROL=(USER_ID=*::ABS,ACCESS="READ+WRITE+SET+SHOW+CONTROL+DELETE") -`
	`_ABS> /PROFILE=(CLUSTER=*,NODE=*,USERNAME=ABS)`
	Restrictions:
	• You can only create (or modify) an environment policy on ABS server node.
	• The account that is creating or modifying the environment policy must have ABS_CREATE_EXECUTION_ENV access rights identifier granted. See *Archive Backup System for OpenVMS Installation Guide* for instructions about granting access rights identifiers.
	Because the environment policy NTNODE_SYSTEM_BACKUPS_ENV specifies a user profile with the user name ABS, only system backup operations in the context of ABS are allowed. This is a restriction of NT and UNIX backup operations.

5.5.2 Creating Save Requests for NT and UNIX Client System Backup Operations

You can create system save requests for an NT or UNIX client node from ABS OpenVMS server node. Table 5–9 describes how to do this:

Table 5–9 Creating Storage and Environment Policies for NT/UNIX Client System Backup Operations

Step	Action

1.	**From ABS server node**:
	To create an NT or UNIX system save or restore request for an NT or UNIX client node (NTNODE) from ABS server node, on the server node (SVNODE) create a save request that uses the storage policy named NTNODE_SYSTEM_BACKUPS and the environment policy named NTNODE_SYSTEM_BACKUPS_ENV. Specify ABS NT or UNIX client node as the source node (where the data to save resides):
	`$ ABS SAVE/STORAGE_CLASS=NTNODE_SYSTEM_BACKUPS/FULL "C:\" -`
	`_$ /OBJECT=WINDOWS_NT_FILES_GTAR /SOURCE_NODE=ntnode`
	Result:
	The save or restore operation is executed on the server node SVNODE.

5.6 Oracle Rdb Databases and Storage Areas Backup Operations

ABS supports backup and restore operations of Oracle Rdb databases and storage areas. ABS uses "file types" to distinguish the type of data being saved or restored. This ensures that ABS invokes the correct backup agent for the save or restore request. File types are an option on the GUI, or if you are using DCL, specified by using the /OBJECT_TYPE qualifier.

ABS provides the following file types for Oracle Rdb databases and storage areas:

- RDB_V4.2_Database
- RDB_V4.2_Storage_Area
- RDB_V5.1_Database

- RDB_V5.1_Storage_Area
- RDB_V6.0_Database
- RDB_V6.0_Storage_Area
- RDB_V6.1_Database
- RDB_V6.1_Storage_Area
- RDB_V7.0_Database
- RDB_V7.0_Storage_Area

The following example shows how you might create a full save request that performs subsequent nightly incremental saves of an Oracle Rdb database:

```
$ ABS SAVE "DISK$RDB_DISK:[RDB_60_DATABASE]RDB_60_DATABASE.RDB" -
_$ /OBJECT_TYPE="RDB_V6.0_DATABASE"/INTERVAL=DAILY_FULL_WEEKLY/START=22:00 -
_$ /STORAGE=SYSTEM_BACKUPS/NAME=RDB_60_DATABASE_BACKUP
```

Result:
This command causes ABS to create a job for the request named RDB_60_DATABASE_BACKUP that runs every night at 22:00 (10:00 p.m.). The first time the job runs, ABS does a full backup operation of the Oracle Rdb database. Each subsequent night for the next six nights, ABS performs an incremental backup operation of the Oracle Rdb database. On the seventh night, the cycle is repeated. All save sets are written to the storage policy named SYSTEM_BACKUPS.

Night 1 at 22:00	Night 2-6 at 22:00	Night 7 at 22:00
Full Save Request	Incremental Save Request	Full Save request Cycle repeats

5.6.1 Saving Individual Storage Areas

To save individual storage areas in an Oracle Rdb database, you must add the /INCLUDE qualifier to the Oracle Rdb database name specified in the save request. You must include this qualifier whether you are using the GUI or the DCL interface.

The following example shows how to back up only AREA3 of an Oracle Rdb database using DCL:

```
$ ABS SAVE -
_$ "DISK$RDB_DISK:[RDB_60_DATABASE]RDB_60_DATABASE.RDB/INCLUDE=AREA3"-
_$ /OBJECT_TYPE="RDB_V6.0_STORAGE AREA"/INTERVAL=DAILY_FULL_WEEKLY -
_$ /START=21:00/STORAGE=SYSTEM_BACKUPS/NAME="RDB_60_DB_AREA3_BACKUP"
```

Result:
This command causes ABS to create a job for the request named RDB_60_DB_AREA3_BACKUP that runs nightly at 21:00 (9:00 p.m.). The first time the job runs, ABS performs a full backup operation of the storage area. Each subsequent night for the next six nights, ABS performs an incremental backup operation of the storage area. On the seventh night, the cycle is repeated. All save sets are written into the storage policy named SYSTEM_BACKUPS.

Night 1 at 21:00	Night 2-6 at 21:00	Night 7 at 21:00
Full save request	Incremental save request	Full save request cycle repeats

If you want to specify more than one storage area, you can include a comma-separated list of storage area names:

Example:

```
$ ABS SAVE-
_$"DISK$RDB_DISK:[RDB_60_DATABASE]RDB_60_DATABASE.RDB/INCLUDE=(AREA1,AREA3)"
_$/OBJECT_TYPE="RDB_V6.0_STORAGE_AREA"/INTERVAL=DAILY_FULL_WEEKLY-
_$/START=21:00$_/STORAGE=SYSTEM_BACKUPS$_/NAME="RDB_60_DB_AREA3_BACKUP"
```

5.6.2 Catalog Entries

For a save request that specifies an Oracle Rdb database, ABS creates multiple entries in ABS catalog:

- One for the Oracle Rdb database itself
- One for each storage area

Because ABS creates catalog entries for each storage area within a Oracle Rdb database, you can restore any individual storage area from the save set that contains the Oracle Rdb database data.

5.6.2.1 Oracle Rdb Database Catalog Entries:

Catalog entries for the Oracle Rdb database have the same format as a file specification:

```
DISK:[DIRECTORY]DATABASE_NAME.RDB
```

5.6.2.2 Oracle Rdb Storage Area Catalog Entries:

Catalog entries for the Oracle Rdb storage areas consist of the Oracle Rdb database name to which the storage area belongs, plus an /AREA qualifier indicating the name of the storage area:

```
DISK:[DIRECTORY]DATABASE_NAME.RDB/AREA=STORAGE_AREA_NAME
```

For example, suppose an Oracle Rdb V6.0 database named RDB_60_DATABASE resides in DISK$RDB_DISK:[DATABASE]. Also suppose that this database contains three storage areas named AREA1, AREA2 and AREA3. If you create a save request that specifies this Oracle Rdb database, the save request also saves the storage areas within the database.

In this situation, ABS creates the following catalog entries:

```
Object type: RDB_V6.0_DATABASE
Object name: DISK$RDB_DISK:[DATABASE]RDB_60_DATABASE.RDB

Object type: RDB_V6.0_STORAGE_AREA
Object name: DISK$RDB_DISK:[DATABASE]RDB_60_DATABASE.RDB/AREA=AREA1

Object type: RDB_V6.0_STORAGE_AREA
Object name: DISK$RDB_DISK:[DATABASE]RDB_60_DATABASE.RDB/AREA=AREA2

Object type: RDB_V6.0_STORAGE_AREA
Object name: DISK$RDB_DISK:[DATABASE]RDB_60_DATABASE.RDB/AREA=AREA3
```

If you create a save request that specifies only a storage area, ABS only creates entries in the catalog for the storage area and not for its associated Oracle Rdb database.

5.6.3 Searching for Storage Areas in the Catalog

To find catalog entries for Oracle Rdb storage areas, use one of the following methods using either the GUI or DCL:

1. The following DCL example specifies the correct storage area syntax:

```
$ ABS LOOKUP "DISK$RDB_DISK:[DATABASE]RDB_60_DATABASE.RDB/AREA=AREA2" -
_$ /OBJECT_TYPE="RDB_V6.0_STORAGE_AREA"
```

Note

If you use DCL for the lookup operation, you must enclose the file type in quotation marks. This is because the /AREA qualifier is part of the catalog entry and not a qualifier on the DCL command line.

2. The following DCL example finds all storage areas for a specific database:

```
$ ABS LOOKUP “DISK$RDB_DISK:[DATABASE]RDB_60_DATABASE.RDB/AREA=*” -
_$ /OBJECT_TYPE=”RDB_V6.0_STORAGE_AREA”
```

3. The following GUI example finds all storage areas with the name AREA3 for any Oracle Rdb database on DISK$RDB_DISK::

```
Object Type: ALL
Object Name: DISK$RDB_DISK:[*]*.RDB/AREA=AREA3
```

Restriction:
You cannot use any wildcard characters in the disk name.

4. The following GUI example finds all instances of saved data on a specific disk (including storage areas and Oracle Rdb databases):

```
Object type: ALL
Object name: DISK$RDB_DISK:[*]*.*
```

5.6.4 Restoring Storage Areas and Databases

Using ABS, you can restore entire Oracle Rdb databases or individual storage areas. There are two types of restore requests, full and selective.

Recommendation:
To restore Oracle Rdb databases or storage areas, it is recommended that you create a full restore request. This causes ABS not only to restore the most recent full backup of the data, but also to apply any subsequent incremental backup save sets.

Example of restoring an RDB database using DCL:

```
$ ABS RESTORE ”DISK$SLSRMU2:[RDB_60_DATABASE]RDB_60_DATABASE.RDB” -
_$/OBJECT_TYPE=”RDB_V6.0_DATABASE”/FULL
```

Requirement:
The files associated with the Oracle Rdb database must have been deleted from the disk before issuing this command, or on the DCL command line include the /CONFLICT=NEW qualifier.

Example of restoring an Oracle Rdb storage area using DCL:

```
$ ABS RESTORE -
_$”DISK$SLSRMU2:[RDB_60_DATABASE]RDB_60_DATABASE.RDB/AREA=SLS_DEV1_AREA5” -
_$/OBJ=”RDB_V6.0_STORAGE_AREA”/FULL
```

Requirement
The files associated with the storage area must have been deleted from the disk before issuing this command, or on the DCL command line include the /CONFLICT=NEW qualifier.

5.7 Cataloging Copied Backup Savesets

ABS supports cataloging information from copied backup savesets on tape, or from tapes created by VMS Backup into ABS catalogs. This allows you to lookup and restore files from savesets created outside of ABS.

To allow this functionality, a new object_type, VMS_SAVESET was created.

Restrictions:

- The saveset must reside on a tape.
- Only VMS Backup savesets may be cataloged.
- The tape volume must be moved into MDMS and allocated to ABS so that ABS may reference the volume.
- A separate catalog and storage_class should be created for the saveset information.

To catalog the saveset information, you must create a save request with the tape_volume, a colon , and the saveset name, or wildcard, as the include specification (ie. Tape001:mysaveset.sav), an object type of VMS_SAVESET, and a selective movement type:

```
$ ABS SAVE tape001:mysaveset.sav –
/OBJECT_TYPE=VMS_SAVESET –
/SELECTIVE –
/NAME=mysaveset_catalog –
/STORAGE_CLASS=my_sc –
/ENVIRONMENT=my_env –
/START=01-JUL-2000

 or

$ ABS SAVE tape001:* –
/OBJECT_TYPE=VMS_SAVESET –
/SELECTIVE –
/NAME=mysaveset_catalog –
/STORAGE_CLASS=my_sc –
/ENVIRONMENT=my_env –
/START=01-JUL-2000
```

ABS will load the tape listed in the include specification, then do a Backup/List of the contents, loading the information into the ABS catalog defined in the storage_class. The original date of the saveset will be preserved in the catalog.

Recommended Implementation:
It is recommended that you create a new catalog to store this data. You also should create a new storage_class to be used by these cataloging operations.

This will allow you to restore from the copied tapes or from the original tapes by selecting the appropriate storage_class for the restore request.

For example:

Several ABS save requests were saved on tape ABS000 using the SYSTEM_BACKUPS storage_class. Saveset Manager was used to copy that tape to another tape, TAP000.

Before cataloging the data, do the following:

- Create a new catalog called COPIED_TAPES. Create a storage_class called COPIED_SC which points to the catalog COPIED_TAPES.
- Create a save request specifying TAP000:* for the include specification, and give it an object type of VMS_SAVESET and the COPIED_SC storage_class.

ABS will execute the request, cataloging the information in the COPIED_TAPES catalog.

To restore the data which is on ABS000 or TAP000, decide which copy you wish to restore and specify the appropriate storage_class in the restore request. For example, to restore from the original tapes, specify the SYSTEM_BACKUPS storage_class. To restore from the copy, specify the COPIED_SC storage_class. The ABS LOOKUP command with the /FULL qualifier will show the volumes used for the data.

Note

If the information about the original and copied savesets is put into the same catalog, they will have exactly the same archived date. This could cause confusing when restoring the data because ABS may not choose the tapes you wish to use for the request. To make it easier to restore, it is recommended to use a separate catalog (as described above).

6

Displaying ABS Graphical User Interface

ABS GUI allows you to create, modify and delete ABS policies and requests. It also allows you to find data that was previously saved using ABS.

ABS enables you to display GUI on the OpenVMS server or client system, on an NT client system, or on any system that supports X Window System™ for Motif®.

6.1 Displaying ABS GUI On an OpenVMS System

Use the procedure to display ABS GUI on an OpenVMS system.

Table 6–1 Displaying ABS GUI on an OpenVMS System

Step	Action
1.	Set display to your current node. **Example**: `$ SET DISPLAY/CREATE/NODE=OpenVMS_node_name` **Note**: Make sure that you have added the node from which you are accessing the GUI to the session manager security option.
2.	Enter the following command: `$ ABS/INTERFACE=DECWINDOWS` **Result**: ABS displays ABS GUI main window. See Figure 6–1.

Figure 6–1 ABS Main Window

6.2 Displaying ABS GUI on an NT System

To display the GUI on an NT system using the eXcursion™ software, follow the procedure in one of the following tables:

- Table 6–2–Describes how to display the GUI using eXcursion and DCL commands.
- Table 6–3–Describes how to display the GUI using the eXcursion menu options.

Table 6–2 describes how to display the GUI using eXcursion and DCL commands.

Table 6–2 Displaying the GUI On an NT System Using eXcursion and DCL Commands

Step	Action
1.	Click the eXcursion icon, select Applications from the menu choices. Click OpenVMS server node and execute ABS software.
2.	Enter the password of the account invoking the application (typically SYSTEM).
3.	Once you have logged into ABS OpenVMS server node, set display to the NT client node: `$ SET DISPLAY/CREATE/NODE=NT_nodename/TRANSPORT=TCPIP`
4.	Enter the following command to invoke ABS GUI: `$ ABS/INTERFACE=DECWINDOWS`

Table 6–3 describes how to display ABS GUI using the eXcursion Menu options.

Table 6–3 Displaying ABS GUI Using eXcursion Menu Options

Step	Action
1.	Click eXcursion icon, select Control Panel.

Table 6–3 Displaying ABS GUI Using eXcursion Menu Options

Step	Action
2.	Click Accounts from the labeled tabs displayed at the top of the window.
3.	Enter the OpenVMS ABS server node name in the Account Alias: and Host: areas.
4.	Select either Username and enter the user name of the account that will be creating save and restore requests (typically SYSTEM) or select Prompt for Username. If you select Prompt for Username, you will be prompted for the user name each time you invoke the GUI from eXcursion.
5.	Select either Password and enter the password for the Username or select Prompt for Password. If you select Prompt for Password, you will be prompted for the password each time you invoke the GUI from eXcursion.
6.	Click Add.
7.	Click Applications from the labeled tabs.
8.	Enter ABS GUI in the Application Alias: area.
9.	Enter the following command in the Command area: `$ RUN ABS$SYSTEM:ABS_UI.EXE`
10.	Enter the OpenVMS server node name that you entered in Accounts window.
11.	Click Add, click Run.
12.	If you selected Prompt for Username, enter the user name of account that will be creating save and restore requests (typically SYSTEM).
13.	If you selected Prompt for Password, enter the password for the account on OpenVMS ABS server node.

6.3 Standard X Window for Motif Buttons

ABS GUI windows contain the following standard buttons:

- OK - Prompts for confirmation about submitting the operation.
- Cancel - Cancels the operation.
- Submit - Submits the creation or modification of a request or policy to ABS policy database.
- Help - Displays Help in Bookreader format about the operation that you are performing. For example, if you are saving data, when you click Help, ABS displays information about creating a save request.

7

Creating Storage Policies

A storage policy defines the type of archive file system, type of media, and archive characteristics for ABS save sets. ABS provides the following preconfigured storage policies:

- ABS_ARCHIVE
- DISASTER_RECOVERY
- SYSTEM_BACKUPS
- USER_BACKUPS
- UNIX_BACKUPS

Note

Storage policy creation is not available with the ABS-OMT license.

Archive Backup System for OpenVMS Installation Guide describes the characteristics of these preconfigured storage policies. To meet your storage management needs, you may have to create additional storage policies. Review the information in this chapter to determine how to create a storage policy that meets your site-specific needs.

This chapter describes the following information:

- Instructions for using the worksheets provided in Appendix E
- Instructions for creating an ABS storage policy using ABS graphical user interface (GUI)

Note

If you are using the command line interface (CLI) to create a storage policy, refer to *Archive Backup System for OpenVMS Command Reference Guide* for the command syntax and qualifier descriptions.

The command line interface is not available with the ABS-OMT license.

7.1 Using ABS Policy Worksheets

Chapter 7, Chapter 8, and Chapter 9 are designed for use with the worksheets provided in Appendix E. Use the worksheets as scratch areas to help you create and manage your ABS storage and environment policies.

Follow these steps to use the worksheets along with the information in this chapter:

Step 1. Remove the worksheets from Appendix E.

If possible, make copies of the worksheets so that you can keep the originals for future configurations or modifications.

Step 2. Review Chapter 7, Chapter 8, and Chapter 9.

Note

Each worksheet in Appendix E corresponds with an ABS policy. The following chart maps each ABS policy object with its corresponding worksheet

ABS Policy	Corresponding Worksheet
Chapter 7, *Creating Storage Policies*	Table E-1,
Chapter 8, *Creating Environment Policies*	Table E-2,
Chapter 9, *Creating Save Requests*	Table E-3

Step 3. Determine the GUI fields that you need to configure for your ABS policies and record those fields on the worksheets.

Step 4. When you have completed the worksheets, create your ABS policy according to the completed worksheets.

7.2 Requirements

To create a storage policy, you must:

- Have ABS_CREATE_STORAGE_CLASS access rights identifiers enabled on your process.
- Be logged into ABS server node.

7.3 Creating an ABS Storage Policy

Use the procedure in Table 7–1 to create an ABS storage policy.

Table 7–1 Creating an ABS Storage Policy

Step	Action
1.	Display the GUI (see Chapter 6, *Displaying ABS Graphical User Interface*).
2.	Click Define Storage Policy from the main GUI window. **Result**: ABS displays the Define Storage Policy window
3.	Review the following sections for each of the options for the Define Storage Policy. To determine your site-specific requirements, use the worksheet provided in Table E-1.

7.4 Storage Policy Name

Each storage policy must have a unique name and be made up of alphanumeric characters, hyphens (-), underscores (_), or a combination thereof. Assign the storage policy a name that reflects the purpose of the storage policy. For example, to meet your long-term storage requirements you may want to create storage policies with the names of 1_YEAR, 5_YEARS, PROJECT_X_DATA, and so forth.

Character Limit:
The storage policy name cannot exceed 31 characters. Because the matching environment policy name is typically the storage policy name appended with _ENV, it is recommended that a storage policy name not exceed 27 characters.

To make sure that you retain the data for the amount of time indicated in the storage policy name, set the Retain Data For option to match those time frames. See Section 7.6 for information about setting the retention period.

7.5 Save Data To

The Save Data To option enables you to choose whether you want to save data to a volume in the MDMS database, or whether you want to save data to an OpenVMS disk.

Use the procedure in Table 7–2 to select where you want to store data for this storage policy.

Table 7–2 Selecting Tape or Disk Storage

Step	Action
1.	Click the box next to Tape or Disk. Your selection becomes highlighted.
2.	If you selected Tape Options, click the box next to Tape Options and see Section 7.5.1 for instructions.
3.	If you selected Disk Options, click the box next Disk Options and enter the OpenVMS disk and directory specification in the Root Directory to Save Data To box: `ABS$ROOT:[000000]`

7.5.1 Tape Options

By selecting Tape Options, you are instructing ABS to use volumes and drives managed by the Media and Device Management software (described in *Archive Backup System for OpenVMS Installation Guide* and in Part II of this manual):

- Section 7.5.1.1describes how to assign a specific MDMS media type to the storage policy.
- Section 7.5.1.2 describes how to use a pool of volumes.
- Section 7.5.1.3 describes how to use specific tape drives associated with specific media types.
- Section 7.5.1.4 describes how to use volumes from a specific MDMS location.
- Section 7.5.1.5 describes how to specify the criteria under which ABS will create new volume sets.

7.5.1.1 Media Type

For the Media Type option, enter the media type name (previously configured in MDMS) that you want this storage policy to use. Specifying a media type name instructs ABS to use only this type of media for any save requests that use this storage policy.

Requirement:
Supplying a valid media type is required.

Example of how a media type is defined in MDMS:
```
$ MDMS CREATE MEDIA_TYPE TK85K
```

Additional Information:
For additional information about media types, drives, pools, and location, refer to Chapter 18, Basic MDMS Operations.

7.5.1.2 Pool

The Pool option enables you to enter a pool name that has been previously created in MDMS. A pool contains free volumes that only certain users can access, such as ABS. Chapter 18 describes

how to create pools in MDMS. Even though a media type may be available to several pools, you may want to restrict a storage policy to only one pool of volumes.

7.5.1.3 Drives

The Drives option allows you to enter a specific drive or comma-separated list of drives to use for this storage policy. Specify the drive name as MDMS drive name rather than the VMS device name.

Example:
```
DRIVE2,DRIVE3,DRIVE4
```

where the following drive names are defined as follows in MDMS:

- `MDMS CREATE DRIVE DRIVE2 /DEVICE=$4$MUA892`
- `MDMS CREATE DRIVE DRIVE3 /DEVICE=$4$MUA893`
- `MDMS CREATE DRIVE DRIVE4 /DEVICE=$4$MUA894`

Recommendation:
Do not assign drive names to this option, but instead, configure MDMS so that the appropriate media types are paired with the desired drives.

7.5.1.4 Location

The Location option enables you to specify the location of the volumes that you want to assign to the storage policy. The location is defined in MDMS.

7.5.1.5 Criteria Under Which ABS Creates Volume Sets

The criteria that ABS uses to create volume sets is determined by the options described in the following sections. It is important to note that the values in all of these options are desired values and not always absolute.

For example, if all of the disk or file names specified in a save request have not been backed up before the specified criteria is met, ABS continues to perform the backup operation to the same volume set even though the criteria may have been exceeded during the save operation.

See the *ABS Command Reference Manual* for more information on setting the criteria using DCL. The qualifier in the ABS CREATE STORAGE_CLASS command is /CONSOLIDATION=(INTERVAL,COUNT,SIZE).

7.5.1.5.1 Days Before Creating a New Volume Set –Enter the number of days (in OpenVMS time format) that you want between the creation of a new volume set. If you assign the value 10, a new volume set is created every ten days.

The default number of days is 7.

7.5.1.5.2 Save Sets Per Volume Set –Enter the number save sets you want to allow per volume set. For example, if you set this option to one (1), ABS creates a new volume set for each save set. If this value is set to 10, ABS creates a new volume set for every ten save sets.

Recommendation:
It is recommended that you enter zero (0) for this option. By entering zero, a new volume set is created based upon the values assigned to the criteria options as described in Section 7.5.1.5.1 and Section 7.5.1.5.3.

Default:
The default value for this option is zero (0).

7.5.1.5.3 Volumes Per Volume Set – Enter the maximum number of volumes that you want to allow per volume set.

7.5.1.6 Clear Volume Set List From Storage Policy

This option is only available when you modify an existing storage policy. During the create process, this option is grayed out.

There may be a time when you want to remove the reference to a particular volume set and start a new one. ABS allows you to do this. Follow these steps:

Step 1. Follow the steps in Table 12–2 to access the Modify or Delete Policies & Requests window. Select the storage policy from which to clear the volume set name.

Step 2. Click Tape Options

Result:
ABS displays the Tape Options window.

Step 3. Click Clear Volume Set

Result:
ABS displays a list of volume set names that will be cleared from the storage policy.

Step 4. Click OK to clear the list of volume set names.

Result:
The volume set list disappears and ABS displays the following message:

```
Volume Set will be cleared when you clicked OK on the Tape Options window
```

Step 5. Click OK on the Tape Options window.

Result:
ABS displays the following message:

```
Clearing of Volume Set for Storage Policy storage_policy_name Succeeded.
```

7.6 Retain Data For

Use the Retain Data For option to assign the period of time to retain data saved using this storage policy. ABS provides two options, Days and Expiration On. These options are mutually exclusive. You can assign the number of days to retain the data, or you can assign an exact date that you want the saved data to expire. ABS default retains data for 365 days.

Use one of the following options to define how long to save the data:

Table 7–3 Options to Save the Data

Option	Action
Retain For	To use the Retain For option, click DAYS, click the box next to it. Enter the number of days to retain the data. For example, to meet legal requirements, you may have data that you need to retain for 5 years. Select Days and enter 1851 (365 days X 5). **Maximum Number of Days**: The maximum number of days is 9999.

Table 7–3 Options to Save the Data

Option	Action
Expire On	To use the Expire On option, click Expire On and select Today, Tomorrow, or Specific Date. If you select Specific Date, click the date box and enter the date and time in standard OpenVMS format. For example, to have the saved data expire on March 13, 2000 at 2:00 p.m, enter the following format: `13-MAR-2000 14:00`

7.7 Catalog and Execution Node

Each ABS storage policy uses an ABS catalog to record the history of saved data. Specify ABS catalog that you want this storage policy to use, and execute the save request on the node specified for the Execute Save Operation On option.

7.7.1 Selecting ABS Catalog

ABS uses its catalogs to locate data saved using ABS. ABS provides a default catalog named ABS_CATALOG.

To select the catalog to use for the storage policy, click the box next to Write History Information To and select one of ABS catalogs from the list.

Note

You can assign the same catalog name to multiple storage policies.

Requirements:
If you wish to use a catalog other than one of the default catalogs provided by ABS, make sure that you:

- Create the catalog before you assign it to the storage policy
- Place the catalog in ABS$CATALOG directory

Chapter 15, *Creating ABS Catalogs* provides information about creating ABS catalogs.

7.7.2 Selecting the Node of Execution

To select the node on which to execute the save request, click the box next to Execute Save Operation On and select the node name from the list of node name. If the node name does not appear in the list, select Other and enter the node name.

7.8 Number of Streams

This option allows you to configure the storage policy so that you can execute more than one save request simultaneously.

For example, if you create three save requests that are scheduled to start at the same time, those save requests can run simultaneously provided there are enough media management resources (such as tape drives and free volumes) to support multiple backup operations. In this case, you would set the Number of Streams option to 3. Valid values range from 1 to 36.

To increase or decrease this option, place the pointer on the slide rule and hold down MB1, slide up to decrease the value, slide down to decrease the value.

Default:
The default value is 1.

7.9 Storage Policy Access Control

The Storage Policy Access Control option enables you to authorize users to access the storage policy, and to enable those users with certain types of access controls.

The user who creates the storage policy is automatically granted access to it, and he is granted all of the access controls (READ, WRITE, SHOW, SET, DELETE, and CONTROL). To add other users and to provide them with access control to the storage policy, use the procedure described in Table 7–4.

Table 7–4 Enabling Access Control to the Storage Policy

Step	Action
1.	Click Add
2.	Click Node Name to add the node name of the user
3.	Click User Name to add the user's name
4.	Click the box next the access control that you want to enable for the user you are adding or modifying. See the following section for access control descriptions. • Read - Users with Read access control can restore data using the storage policy. • Write - Users with Write access control can save data using the storage policy. • Delete - Users with Delete access control can delete the storage policy object if the number of catalog references is set to zero (0). • Set - Users with Set access control can modify any attribute of the storage policy object, including access control. • Show - Users with Show access control can show the storage policy object. • Control - Users with Control access control can modify the access control for the storage policy object, but not any of its other attributes.

7.10 Submitting the Storage Policy

Once you have entered all the information for the storage policy, do the following:

1. Click OK on the main window to submit the storage policy to ABS policy database.

   Result:
   ABS displays the Submit Storage Policy window, this window contains the basic information for the storage policy.

2. Click OK to submit the storage policy, or click Cancel to cancel the submit operation. If you cancel, the main window reappears and you can modify any information, or click Cancel to cancel creating the storage policy.

8

Creating Environment Policies

An environment policy defines the environment in which save and restore requests are executed. ABS provides the following default environment policies:

- ABS_ARCHIVE_ENV
- DEFAULT_ENV
- DISASTER_RECOVERY_ENV
- SYSTEM_BACKUPS_ENV
- USER_BACKUPS_ENV
- UNIX_BACKUPS_ENV

Note

Environment policy creation is not available with the ABS-OMT license.

Archive Backup System for OpenVMS Installation Guide describes the characteristics of these default environment policies. To meet your storage management needs, you may have to create additional environment policies. Review the information in this chapter to determine how to create an environment policy that meets your site-specific needs.

The information in this chapter includes:

- Instructions for using the worksheets provided in Appendix
- Instructions for creating an ABS environment policy using ABS graphical user interface (GUI)

Note

If you are using the command line interface (CLI) to create an environment policy, refer to Archive Backup System for OpenVMS Command Reference Guide for the command syntax and qualifier descriptions.

The command line interface is not available with the ABS-OMT license.

8.1 Using ABS Policy Worksheets

Chapter 7, Chapter 8, and Chapter 9 are designed for use with the worksheets provided in Appendix E. Use the worksheets as described in Section 7.1.

8.2 Requirements

To create an ABS environment policy, the creating process must:

- Have ABS_CREATE_EXECUTION_ENV access right identifier enabled.

- Be logged into ABS server node. See *Archive Backup System for OpenVMS Installation Guide* for information about ABS server.

8.3 Creating an ABS Environment Policy

Use the procedure described in Table 8–1 to create an ABS environment policy:

Table 8–1 Creating an ABS Environment Policy

Step	Action
1.	Display the GUI (see Chapter *6, ABS Graphical User Interface)*
2.	Click Define Environment Policy from the main GUI window. **Result**: ABS displays the Define Environment Policy window
3.	Review the following sections for each of the options for the Define Environment Policy. To determine your site-specific requirements, use the worksheet provided in Table E-1.

8.4 Environment Policy Name

Each environment policy must have a unique name and be made up of alphanumeric characters, hyphens (-), underscores (_), or a combination thereof. The environment policy name should have a matching storage policy name with the characters _ENV appended to it. For example, if you create a storage policy named 5_YEARS, the matching environment policy name should be 5_YEARS_ENV. ABS will look for a matching storage and environment policy names. If you do not have a matching environment name, then ABS will use the default environment policy named DEFAULT_ENV.

Restriction:
The environment policy name cannot exceed 31 characters.

8.5 Save and Restore Environment Options

The Save and Restore Environment Options enable you to set up the conditions under which save and restore requests will execute using this environment policy. the following sections describe these conditions and how to set them.

8.5.1 Who to Notify

The Who to Notify option enables you to specify several methods of notification and conditions under which notifications about ABS save and restore requests are executed. You can notify personnel when ABS save and restore request complete successfully, or when and why they fail.

8.5.1.1 How to Notify and Who to Notify

To enable these options, use the procedure in Table 8–2

Table 8–2 Selecting the Notification Options

Step	Action
1.	Click the Form of Notification box. ABS supplies the following options: • OPCOM - Choose this option (default) to notify the operator using a valid OPCOM class (TAPES) • VMS Mail - Choose this option to notify a specific user through a VMS mail account • None - Choose this option if you do not want any method of notification
2.	If you select OPCOM, ABS automatically fills in the VMS class with TAPES.
3.	If you select VMS Mail, click the box adjacent to VMS Mail Address to Receive Notification and enter a valid user name: NODESV::SMITH
4.	If you select NONE, the environment policy will not send a notification under any conditions.

8.5.1.2 When to Notify

An ABS environment policy provides several conditions as when to notify. To set these conditions, do the following:

- Click the box adjacent to Conditions Under Which to Send Notification. Select one or more of the following options:
  - Starting - Notifies when an ABS save or restore request begins
  - Completing - Notifies when an ABS save or restore request has completed
  - Warnings - Notifies if an ABS save or restore request encounters any warning errors
  - Errors - Notifies if an ABS save or restore request encounters any type of errors
  - Fatal Errors - Notifies if an ABS save or restore request encounters fatal errors only

8.5.1.3 Type of Notification

An ABS environment policy allows you to select the type of notification message to display. To set the type of notification, do the following:

- Click the box adjacent to Type of Message to Return and select one of the following options:
  - Brief - Contains only basic information (default)
  - Normal - Contains a moderate amount of information
  - Verbose - Contains the maximum amount of information

8.5.2 Data Verification

The Data Verification options provides you with the ability to specify default data safety checks for ABS save and restore requests that use this environment policy. Data verification include checks such as full data verification, redundancy checks, and tape read verification. This option enables you to ensure data safety by performing various types of data verification during execution of ABS save and restore requests.

To enable one or more Data Verification options, do the following:

- Click the box next to the option you want to enable. Select one or more of the following options:
  - Full Data Verification - Instructs ABS to reread all data and compare it to what is on the disk.
  - Cyclic Redundancy Check - Performs a cyclic redundancy check (CRC) and writes it for each data block on the tape. This enables detection of a bad block during restore operations.
  - XOR Redundancy Groups – If the CRC check detects a bad block during a restore operation, the XOR mechanism allows recovery of the block by using redundancy groups of blocks (written during the save operation). The OpenVMS Backup Utility is the only backup agent currently provided by ABS that provides an XOR mechanism. It is enabled by default with a redundancy group size of 10 blocks.

8.5.3 Listing

The Listing option allows you to specify the default listing file behavior for save and restore requests that use this ABS environment policy.

- Click on one of the following options:
  - None - Does not generate a listing file.
  - Brief - Generates a brief listing file.

Note

A brief listing file is ABS specific and lists each disk or file name saved during a save operation.

  - Full - Generates a full listing file.

8.5.4 Pre- and Post- Processing Commands

The Processing Commands option enables you to specify pre- and post- processing commands that will execute once. A preprocessing command executes once, prior to the start of the save or restore request, and a postprocessing command executes once, after the completion of the save or restore request. This option accepts a platform specific string.

ABS generates logical names that may be used from within the prologue and epilogue (pre- and post- processing commands) for the entire save request. These may be referenced in a command procedure which is executed as a prologue or epilogue for the environment policy.

These logical names are defined in the process JOB table. They exist during the save request. Once the save request is complete the logicals will no longer be available.

ABS_SAVE_REQUEST_NAME	Name of the save request.
ABS_REQUEST_TYPE	SAVE or RESTORE.
ABS_STORAGE_CLASS	Name of the storage class used by the request.
ABS_EXECUTION_ENVIRONMENT	Name of the execution environment used by the request.
ABS_NODE_NAME	Node name specified in the request.
ABS_OUTPUT_DEVICE	The name of the device used by the save request. If multiple output devices have been used this logical will be a comma separated list.

Restrictions:

- When entering the pre- or post- processing command, an OpenVMS string is limited to 80 characters.
- A postprocessing command will execute only if the save or restore request completes successfully.

8.5.5 Original File

The Original File option enables you to select the default behavior for the original data (OpenVMS disk or file) being saved.

To set the original file option, do this:

- Click on the box located next to one of the following options:
  - Record Backup Date - Sets the backup date in the file header information
  - No Change - Does not change the original online data
  - Delete Original Data - Deletes the data from the online disk when the save request has completed. This option is typically set when using ABS enviroment policy for archiving data.

8.5.6 Retry Options

The Retry Options enables you to specify the number of times and how often a save or restore request should be retried before operator intervention is required.

- Select one of the following options
  - Retry Count - The number of times the save or restore request should be retried prior to activating the notification options. Valid values are 0 to 10,000.
  - Minutes between Retry Attempts - The amount of time (in minutes) between retry attempts. Valid values are 1 to 60.

Restriction:
If the retry count is set to 0 (zero), the minutes between retry attempts also must be set to 0 (zero).

Default:
If you do not assign a value to this option, the default values are 3 for Retry Count and 15 minutes for Minutes between Retry Attempts.

Hint:
Each time a retry attempt occurs, ABS enviroment policy generates a warning message. If you want to be notified of the retry attempt, select Warnings Occur in the When option (refer to section Section 8.5.1).

8.5.7 User Profile

The User Profile option enables you to configure an environment policy that allows save and/or restore requests to run in the context of either ABS process or the user's process provided the users are authorized to create their own save and restore requests.

Restrictions:

- To create a user profile for an environment policy, you must have the following OpenVMS privileges enabled:
  - SYSPRV
  - CMKRNL

- All UNIX save and restore requests must be performed in the context of the ABS process. The environment policies used for UNIX operations must specify the user as ABS for the user profile. You cannot create UNIX or NT backup operations in the context of a user's process.
- To set the environment policy so that save requests run in the context of ABS process, click User Profile and then click ABS process.
- To set the environment policy so that save requests run in the context of the user's process, click User Profile and then click User process. Be sure to authorize users access as described in Section 8.6.

8.5.8 Open Files

The Open Files option allows you to save files that are open during the execution of the save request. If you select Hot Backup, this allows you to save Oracle Rdb databases that are open. Ignore Writers enables you to save OpenVMS, UNIX, and NT files that are open.

8.5.9 Tape Drives

The Tape Drives option enables you to set the number of tape drives used for each ABS save and restore request that use this environment policy. If the requested number of tape drives is available, ABS allocates those drives for the save or restore request. If the requested number of tape drives is not available, ABS will use the amount of tape drives available.

For example, if you create a save request that uses an environment policy where the number of tape drives is set to 3, ABS allocates three drives when the save request job begins. This means those three drives are unavailable to other applications during the time the save request is executing.

Default:
The default value is 1. This value cannot be 0 (zero).

Recommendation:
Use the default value of 1. Allocating more than one drive per save request will constrain the tape drives. This means those tape drives will not be available to other applications while the save request is executing.

8.5.10 Compression

ABS supports the following types of compression for UNIX clients:

- No Compression
- UNIX Compression
- GZIP Compression

Recommendation:
It is recommended that you use the default UNIX environment policy or create an ABS environment policy with the desired compression options for all of your UNIX save requests. Do not mix compression types for UNIX save requests.

For example, use the same ABS environment policy for all of your UNIX save requests. This way, all UNIX data saved using ABS will have the same compression option set.

If you use different types of compression options on different save requests (UNIX save requests use different ABS environment policies), the saved data will be compressed differently. When you attempt to restore the UNIX data, you must know which compression option was used to save the original UNIX data. ABS is unable to restore the file without being instructed to use the proper compression.

Restriction:
This qualifier is not valid for NT client save and restore operations.

Default:
The default ABS setting is No Compression.

8.5.11 Links Option

ABS provides you with the ability to either back up the UNIX symbolic links only, or to follow the UNIX symbolic links and back up the data as well.

Restriction:
This option is valid only for UNIX client operations.

Default:
The default ABS setting is Links Only.

8.5.12 Span Filesystems

ABS allows you to save only the root file system (such as the disk the root directory resides on), or an entire filesystem type if the filesystem spans physical devices. ABS supports the following options:

- None
- All

Restrictions:

- This qualifier is restricted to UNIX files.
- This qualifier is not valid for OpenVMS or NT client backup operations.

Default:
None.

8.6 Environment Policy Access Control

The Environment Policy Access Control option enables you to authorize users to access ABS environment policy, and to enable those users with certain types of access control.

The user who creates the environment policy is automatically granted access to it, and the user is granted all the access controls (READ, WRITE, SHOW, SET, DELETE, and CONTROL). To add other users and to provide them access to ABS environment policy, use the procedure in Table 8–3.

Table 8–3 Enabling Access to an ABS Environment Policy

Step	Action
1.	Click Add
2.	Click Node Name to add the users's node name
3.	Click User Name to add the user's name
4.	Click the box next the access control that you want to enable for the user you are adding or modifying: • Read - Users with Read access control can restore data using the environment policy. • Write - Users with Write access control can save data using the environment policy. • Delete - Users with Delete access control can delete the environment policy object if the number of catalog references is set to zero (0). • Set - Users with Set access control can modify any attribute of the environment policy object, including access control. • Show - Users with Show access control can show the environment policy object. • Control - Users with Control access control can modify the access control for the environment policy object, but not any of its other attributes.

8.7 Submitting the Environment Policy

Once you have entered all the information for the environment policy, do the following:

1. Click OK on the main window to submit the environment policy to ABS policy database.

   Result:
   ABS displays the Submit Environment Policy window; this window contains the basic information for ABS environment policy.

2. Click OK to submit the environment policy, or click Cancel to cancel the submit operation.

   If you cancel, the main window reappears and you can modify any information. Click Cancel to cancel creating ABS environment policy object.

9

Creating Save Requests

Save requests are ABS policy objects that define the data that you want to save, and when to save that data. A save request uses an ABS policy that defines the volume on which the data will be saved (Storage Policy), and the conditions under which the save request will execute (Environment Policy).

Note

The following ABS features are not available with the ABS-OMT license:

- **Support for OpenVMS clients**
- **Save request scheduling options except for the following:**
  - **On demand (ON_DEMAND)**
  - **Weekly full/daily incremental (DAILY_FULL_WEEKLY)**
  - **One time only (ONE_TIME_ONLY)**

9.1 Save Request Name

Each save request must have a unique name and be made up of alphanumeric characters, hyphens (-), underscores (_), or a combination thereof. The default save request name is the user name appended by the current date and time.

To change the default save request name:

- Double-click the default save request name to select it (the name gets highlighted)
- Enter the new save request name

Restriction:
A save request name cannot exceed 40 characters. To satisfy restriction of DecScheduler ABS truncates the log file name to 38 characters, hence it is recommended that you limit the save request name to 38 characters.

9.2 What Data To Save

Use this option to specify the file name, disk name, or set of file or disk names that you want to save:

- File names can be OpenVMS, Oracle Rdb, UNIX, or NT file names.
- Disk names can be OpenVMS or NT disk names.

Note

When creating a new save request, What Data To Save area is blank by default. Once you have added a file name or disk name, that name appears in the What Data To Save display area.

To add a file name, disk name, or a set of file or disk names to the save request, use the procedure in Table 9–1.

Table 9–1 Adding Disk or File Names To A Save Request

Step	Action
1.	Click Add... in What Data To Save area. **Result**: ABS displays What Data To Save window and defaults to the What to Save option.
2.	In the left-hand column, select the type of data to save. For example, to save an entire OpenVMS disk, click Entire OpenVMS Disk. To save an NT file, click NT Individual File.
3.	Select the node where the file or disk resides. This is the first option in the right-hand column. If the node is not available from the listed nodes, click Other and enter the node name in the box.
4.	Enter the file name or disk name that you want to save. You may enter multiple file or disk names (up to eight) as a comma-separated list. However, to add a list of comma-separated file names, they must all be the same file type (OpenVMS, Oracle Rdb, NT, or UNIX). **Note**: To correctly enter the disk or file name, see Table 9–2. **Restrictions**: For a list of restrictions regarding adding disk names and file names to a save request, see Section 9.2.1
5.	To exclude a specific file or set of files from the save request, click the box next to Data to Exclude and enter the file name. For example, enter *.COM to exclude all files with a .COM file extension.

To correctly enter the disk or file name according to file type, see the instructions in Table 9–2.

Table 9–2 Correctly Entering the Disk Name or File Name

Type of Data	Correct Syntax

VMS Files	To save an entire OpenVMS disk, enter the disk name with the trailing colon:
	
	`DISK$USER1:`
	
	To save an individual OpenVMS file or set of files select Individual OpenVMS Files, click Individual VMS Files in the left-hand column and enter the file name or file names using the following syntax:
	
	`DISK$1:*.COM,DISK$2:*.COM`
	
	Note:
	OpenVMS disk and file names are not case-sensitive.

Oracle Rdb Database	To save an Oracle Rdb database, click RDB Database in the left-hand column, select the Oracle Rdb version, and enter the disk and file name using the following syntax: `DISK$1:[USER1_RDB]SITE_PERSONNEL.RDB` **Note**: Oracle Rdb disk and file names are not case-sensitive.

Table 9–2 Correctly Entering the Disk Name or File Name

Type of Data	Correct Syntax
Oracle Rdb Storage Area	To save an Oracle Rdb storage area, click Rdb Storage Area, select the Rdb version, enter the file name using the following syntax: `DISK$1:[USER1_RDB]SITE_PERSONNEL.RDB/INCLUDE= - ACCOUNTING` **Note**: Oracle Rdb disk and file names are not case-sensitive.

UNIX Files	To save the directory /abs and all subdirectories and files underneath it, click UNIX Disk or Directory Tree in the left-hand column and enter the file name using one of the following syntaxes:
	`/abs/`
	`/abs`
	`/abs/*`
	To save an individual file, click UNIX Individual File in the left-hand column and enter the file name using the following syntax:
	`/usr/usr1/file.c`
	Requirement:
	UNIX file names are case sensitive. You must enter the file name exactly as it was created on the UNIX system.

NT Files	To save the NT directory named \usr and all subdirectories and files underneath it, click NT Disk or Directory Tree in the left-hand column and enter the file name using one of the following syntaxes:
	`c:\usr\`
	`c:\usr`
	`c:\usr\*`
	Note:
	NT file names are not case sensitive.

9.2.1 Save Request Restrictions

ABS imposes the following restrictions when adding disk names or file names to a save request:

- Do not specify more than twenty four file names or disk names per save request.
- Do not specify more than one OpenVMS client node name in a single save request.

  For example, do not create a save request that specifies NODEA::DISK$USER and NODEB::DISK1. In this example, ABS starts the save operation on NODEA and then attempts to save the disk named DISK1 (not located on NODEA), and the save request fails.
- Specify wildcard characters only when creating a save request for backup agents that support wildcard characters, such as the VMS BACKUP Utility or gtar for UNIX or NT files.
- Do not specify wildcard characters for an Oracle Rdb database or storage area specification:
  - Oracle Rdb database - An Oracle Rdb database specification requires an OpenVMS disk name, the directory specification, and Oracle Rdb database file name:

    `DISK$USER1:[USER1_RDB]SITE_PERSONNEL.RDB`
  - Oracle Rdb storage area - An Oracle Rdb storage area requires an OpenVMS disk name, the directory specification, the Oracle Rdb database file name, the storage area

name, and the /INCLUDE qualifier:

```
DISK$USER1:[USER1_RDB]SITE_PERSONNEL.RDB/INCLUDE=ACCOUNTING
```

- Specify only system-wide logical names or physical device names - If the OpenVMS disk name contains a concealed logical name, the concealed logical name must be a system-wide logical name or physical device name. An ABS save request translates the logical names to the first concealed logical name found. However, if this concealed logical name is in the process table and not in the system-wide logical table, access to the logical name is not available to ABS, and the save or restore operation will fail.

  Recommendation:

  – Unless you are creating a save request for an entire OpenVMS disk, specify the full path name. Include the disk name, directory name, and file name:

  ```
  DISK$1:[USER1]LOGIN.COM
  ```

  – If you are creating a save request for an entire OpenVMS disk, specify only the disk name and include the trailing colon:

  ```
  DISK$USER1:
  ```

- Bound volume set - When creating a save request for a bound volume set, enter only the first disk name assigned to the bound volume set. ABS recognizes each disk assigned to the bound volume set.

  Restoring a bound volume set:
  To restore a bound volume set, see the restrictions described in Section 10.2.1.

- Do not submit UNIX and NT save requests from an ABS VMS client node. Only submit them on ABS server nodes.

9.2.2 Pre- and Post- Processing Commands

Use the pre- and post- processing commands to submit commands that take action either before or after each file or disk name entered for the save request.

Enter a command string for these options:

Preprocessing command:

```
$ @SHUTDOWN_DB.COM
```

Postprocessing command:

```
$ @STARTUP_DB.COM
```

ABS generates these logical names for use within a prologue or epilogue (pre- or post- processing command) for a SAVE or RESTORE request. A set of "_n" logicals will be generated for each include specification in the request. The series will begin with 1 and continue for each include specification within the request. These may be referenced in a command procedure which is executed as a prologue or epilogue for the save or restore request.

These logical names are defined in the process JOB table. They exist during the save request. Once the save request is complete the logicals will no longer be available.

ABS_OS_OBJECT_SET_n	Include specification.
ABS_OS_OBJECT_TYPE_n	Object type for the object set.

ABS_OS_DMT_n	Data movement type for the object set. Valid values are FULL, SELECTIVE, INCREMENTAL.
ABS_OS_INCREMENTAL_LEVEL_n	If the data movement type is INCREMENTAL, the ABS_OS_INCREMENTAL_LEVEL_n value defines the incremental level of the save request. The string will be in the format "Level n Operation" where n is the incremental level. If the data movement type is FULL the incremental level will be "Full Operation". If the data movement type is a base operation the incremental level will be "Base (Full) Operation". If the data movement type is selective the incremental level will be "Selective Operation".
ABS_OS_VOLUME_SET_n	Name of the volume set used for this request. If the request is to disk the value will be the disk and directory the saveset has been written to.
ABS_OS_START_RVN_n	The starting Relative Volume Number within the volume set that this object set references. The value of this logical will be 0 if the request is going to disk.
ABS_OS_LAST_RVN_n	The relative volume number within the volume set that was last used to save the saveset created by this object set. The value of this logical will be 0 if the request is going to disk. Note that this logical will not be valid for the prologue command. For the prologue command the value will be "Not yet determined" It will be defined for the epilogue command.
ABS_OS_START_FILE_POSITION_n	The starting file position of the saveset on the relative volume number within the volume set. This logical will indicate how many tapes marks to skip to find the start of the saveset on the volume. The value of this logical will be 0 if the request is going to disk.
ABS_OS_SAVESET_NAME_n	The name of the saveset that has been generated by ABS. This saveset contains the data saved based upon the object set.
ABS_OS_SAVESET_FORMAT_n	The format of the saveset. This will determine whether the saveset is in a VMS backup format, gtar format, or RMU backup format.
ABS_OS_STATUS_n	The ABS status of the include specification request.

The logical name ABS_OS_OBJECT_NUMBER is defined in the process table of the backup sub-process to indicate the serial number of the include spec (_n) being backed up in the current sub-process.

Restrictions:

- When entering the pre- or post- processing command, an OpenVMS string is limited to 80 characters.

- You can only enter one pre- and post- processing command per save request.
- A postprocessing command will execute only if the save request completes successfully.

Default:

If you do not assign a string to this option, ABS does not execute any pre- or post processing commands.

9.2.3 Selection Criteria

Enter the specific selection criteria for the disk or file name you are adding to the save request. You can save data specifically by a before or since date.

- Before - Any version of the file created or modified before the specified date.
- Since - Any version of the file created or modified since the specified date.

9.2.4 Agent Qualifier

Enter any backup agent specific qualifiers for the save request. Backup agent qualifiers are determined by the type of file or disk that you are backing up, such as OpenVMS, Oracle Rdb, UNIX, or NT files. Using this qualifier may supersede qualifiers set by ABS. Use this qualifier with extreme caution.

9.3 When to Save Data

ABS provides the option of immediately executing the save request (described in Section 9.3.1), or setting up the save request to execute on a repetitive schedule (described in Section 9.3.2). ABS provides several scheduling options for a save request.

9.3.1 Immediately Executing the Save Request

To execute the save request immediately follow these steps:

Step 1. Click Start and select NOW (default)

Step 2. Click Schedule and select One Time Only (default)

9.3.2 Repetitive Scheduling of Save Request

To schedule a save request to execute on a repetitive schedule, use the following procedure:

1. Click Start and select one of the following options:
   - Now - Submits the save request as soon as you click OK, on the main Save Request window.
   - Today - Submits the save request today at the time specified for the Start Time.
   - Tomorrow - Submits the save request tomorrow at the time specified for the Start Time.
   - OpenVMS Time - Submits the save request per the OpenVMS time specification.

     Click the date box and enter the date and time in standard OpenVMS format. For example, to start the save request on March 13, 2000 at 2:00 p.m, use the following format:

     13-MAR-2000 14:00
2. Click Schedule. See the following descriptions of the scheduling options:
   - One Time Only - Executes the save request one time only according to the option specified for Start Time.

     After the save request has successfully completed and 72 hours have passed, ABS

Database Cleanup Utility routine deletes the job from the scheduler database and from the ABS database. See Appendix C for a description of ABS Database Cleanup Utility.

– On Demand - This option submits the save request to the scheduler and executes the save request once according to the option specified for Start Time. The difference between One Time Only and On Demand is that the ABS Database Cleanup Utility does not delete the save request from the ABS policy database.

  For scheduler options INT_QUEUE_MANAGER and EXT_QUEUE_MANAGER set a new start time to execute the save request.

  For scheduler option DECSCHEDULER use the RUN command to resubmit the save request:
  SCHEDULE> RUN job-name/USER=ABS

  Where job-name is the name of the On-Demand save request.

  For scheduler option EXT_QUEUE_MANAGER use the appropriate command to run the job.

– Daily - Executes a save request once per day according to the option specified for Start Time.

– Weekly Full/Daily Incremental - This scheduling option enables you to create a single save request that executes a full backup operation once per week on the day specified, and an incremental backup operation for each subsequent day after the full backup operation is successful. ABS performs the full backup operation on a fixed day of the week during the 7-day cycle.

  The Weekly Full/Daily Incremental Process:

  For example, if the save request starts the full backup operation on Monday, ABS will always perform the full backup operation on Monday for that particular save request. This happens even if some of the subsequent incremental backup operations fail.

  Example A:

Day	Type
Monday	Full
Tuesday	Level 1
Wednesday	Level 2
Thursday	Level 3
Friday	Level 4
Saturday	Level 5
Sunday	Level 6
Monday	Full

If that full backup operation fails, the cycle is repeated until a successful, full backup operation is achieved. ABS considers success and qualified success as a successful completed operation. ABS considers all other status as a failed operation.

Example B:

Day	Date and Time Run	Type	Result
Monday	31-MAR-1997 02:00	Full	Failure
Tuesday	01-APR-1997 02:00	Full	Failure
Wednesday	02-APR-1997 02:00	Full	Success
Thursday	03-APR-1997 02:00	Level 3	Success
Friday	04-APR-1997 02:00	Level 4	Failure
Saturday	05-APR-1997 02:00	Level 5	Success
Sunday	06-APR-1997 02:00	Assume skipping this day using a 3rd party scheduler	
Monday	07-APR-1997 02:00	Full	Success

Note

If you are manually setting up your 3rd party scheduler to skip special days, ABS skips the next level of an incremental backup operation. In Example B, ABS skips Sunday and does not perform the Level 6 incremental backup operation. ABS resumes the full backup operation again on Monday, and the schedule once again repeats itself.

Notice also in Example B that ABS repeats the full backup operation until a successful full backup operation is achieved on Wednesday. If one of the incremental backup operations fail, ABS skips to the next level of the incremental backup operations. Unlike repeating the full backup operation, ABS does not repeat the same level of incremental backup operations during the 7-day cycle.

In the Example B, the Level 4 incremental backup operation failed on Friday. On Saturday, ABS resumes with a Level 5 incremental backup operation. However, the contents of the incremental backup operations are correct because ABS will back up all new or modified files since the last successful full backup or the last successful lower level incremental backup operation.

The save log file will contain the following backup command issued by ABS for Saturday, 05-APR-1997:

```
$ BACKUP/.../SINCE="03-APR-1997 02:00:00.00"
```

Because the last successful lower level incremental backup operation was performed on 03-APR-1997, all changes to any file since the date and time specified in the BACKUP command are included in the backup operation.

– Weekly - Executes the save request once per week according to the date and time specified for the start time.

– Biweekly - Executes the save request once every two weeks according to the date and time specified for the start time.

– Monthly - Executes the job the first time on the date and time specified in the start time field. Subsequent jobs are scheduled on the first day of each month.

– Quarterly - Executes the job the first time on the date and time specified in the start time field. Subsequent jobs are scheduled to execute on the first day of the quarter (3 month period).

– Semi-annually - Executes the job the first time on the date and time specified in the start time field. Subsequent jobs are scheduled to execute on the first day of the next months.

– Annually - Executes the job the first time on the date and time specified in the start time field. Subsequent jobs are scheduled to execute on the first day of the calendar year.

– Log-2 - ABS executes a full backup operation on day 1, and an incremental backup operation on day 2. On day 3, ABS executes an extended incremental backup operation. An extended incremental backup operation backs up any file modified since the last full or extended incremental backup operation. See Appendix D for an illustrated view of Log-n backup schedules.

– Log-3 - ABS executes a full backup operation on day 1, and an incremental backup operation on days 2 and 3. On day 4, ABS executes an extended incremental backup operation. An extended incremental backup operation backs up any file modified since the last full or extended incremental backup operation.

Advantages of Log-*n* backup operations:

Performing Log-*n* backup operations require less restore operations to fully restore a lost or corrupted disk volume. The higher the number of Log-*n*, the less restore operations you need to perform. Log-*n* backup operations are configured on a 32-day schedule. See Appendix D for an illustrated view of Log-*n* backup operations.

– Explicit Interval - This option enables you to submit the save request using a specific scheduler interval. If you select Explicit, you must enter a scheduler time format valid for the scheduler being used.

– Never - Never submits the save request and does not call the scheduler to create a job. For example, you may need to create one or more save requests before you determine their schedule. To submit the save request, modify the save request and change the scheduling option.

Depending on the selected scheduling option and the use of a 3rd party scheduler product, the Explicit Interval option allows to specify more flexible intervals. The Explicit Interval is passed as a string to the scheduler in use. Consult your scheduler's manual for more information.

9.4 Where and How

ABS enables you to create and use specific policies that define which volumes and tape drives to use for save requests, and in what type of environment to execute the save requests.

A save request must use a storage policy and an execution policy. ABS supplies default policies, but you can use different policies if you so choose.

- Storage Policy - A storage policy defines the volumes and tape drives to use for save operations. It also defines how long to retain the data and when to create new volume sets. See Chapter 7, *Creating Storage Policies* for instructions about setting up data retention and volume consolidation criteria.
- Execution Policy - An execution policy defines the environment in which to execute the save request, such as what context to run the save request (user context or ABS), who to notify when the save request completes, under which error conditions to notify a user, and so forth. See Chapter 8, *Creating Environment Policies* for more details.

To change the storage policy or execution policy, click on the box next to each policy and select one of the policies displayed in the list box.

9.5 Save Request Access Control

The Save Request Access Control option enables you to authorize other users to access the save request, and to enable those users with specific access controls.

The default is the user who creates the save request.

Table 9–3 Enabling Access To An ABS Save Request

Step	Action
1.	Click Add...
2.	Click Node Name to add the users's node name
3.	Click User Name to add the user's name
4.	Click the box next to the access control that you want to enable for the user you are adding or modifying: • Read - Users with Read access control can show a save request. • Write - Users with Write access control can show a save request. • Delete - Users with Delete access control can delete the save request. • Set - Users with Set access control can modify any attribute of the save request, including access control. • Show - Users with Show access control can show the save request • Control - Users with Control access control can modify the access control for the save request, but not any of its other attributes.
5.	Once you have entered the information for the save request, click OK on the main window to submit the save request to ABS policy database. **Result**: ABS displays the Submit Save Request window, this window contains the basic information for the save request. See Section 9.6 for details about submitting a save request.

9.6 Submitting the Save Request

Once you have entered all of the information for the save request, click OK on the main Save Request screen to submit the save request to ABS policy database.

Result:
ABS displays the Submit Save Request screen, this screen contains the basic information for the save request.

- Click Submit to submit the save request.
- Click Cancel to cancel to submit operation.

  If you click Cancel, the main screen reappears with the information you entered. You either can modify this information and resubmit the save request, or you can click Cancel to cancel creating the save request.

10

Creating Restore Requests

Restore requests are ABS policy objects that define the data that you want to restore. A restore request references an ABS catalog that contains the backup information about that data. It also uses an ABS environment policy that defines the conditions under which the restore request will execute.

10.1 Restore Request Name

Each restore request has a unique name and is made of alphanumeric characters, hyphens (-), underscores (_), or a combination thereof. The default restore request name is the user name appended by the current date and time.

To change the default restore request name:

Step 1. Double-click the default restore request name to select it (the name becomes highlighted)

Step 2. Enter the new restore request name

Character Limit:

A restore request name cannot exceed 40 characters. To satisfy restriction of DecScheduler ABS truncates the log file name to 38 characters, hence it is recommended that you limit the restore request name to 38 characters.

10.2 What Data To Restore

Use this option to specify the file name, disk name, or set of file or disk names that you want to restore:

- File names can be OpenVMS, Oracle Rdb, UNIX, or NT file names.
- Disk names can be OpenVMS or NT disk names.

Upon initial creation of a new restore request, the What Data To Restore area is, by default, blank. Once you have added a file name or disk name, it appears in the What Data To Restore area.

To add a file name, disk name, or a set of file or disk names, use the procedure in Table 10–1.

Table 10–1 Adding Disk or File Names To A Restore Request

Step	Action
1.	Click Add... in the What Data To Restore area. **Result**: ABS displays the What Data To Restore window and defaults to the What to Restore option.
2.	In the left-hand column, select the type of data to restore. For example, to restore an entire OpenVMS disk, click Entire OpenVMS Disk. To restore an NT file, click NT Individual File.

Table 10–1 Adding Disk or File Names To A Restore Request

Step	Action
3.	Enter the file name or disk name that you want to restore. You may enter multiple file or disk names (up to eight) as a comma-separated list. However, to add a list of comma-separated file names, they must all be the same file type (OpenVMS, Oracle Rdb, NT, Or UNIX). **Note**: To correctly enter the disk names and file names for the restore request, see Table 10–2. **Restrictions**: For a list of restrictions regarding adding disk names and file names to a restore request, see Section 10.2.1
4.	To exclude a specific file or set of files from the restore request, click on the box Data to Exclude and enter the file name. For example, enter *.COM to exclude all files with a .COM file extension.

Table 10–2 describes how to enter the correct syntax for a restore request according to the file type.

Table 10–2 Entering The Correct Syntax For A Restore Request

File Type	Syntax

VMS Files	To restore an entire OpenVMS disk, enter the disk name and include the trailing colon:
	
	`DISK$USER1:`
	
	To restore an individual OpenVMS file or set of files, select Individual OpenVMS Files in the left-hand column and enter the file name or file names using the following syntax:
	
	`DISK$1:*.COM,DISK$2:*.COM`
	
	Note:
	OpenVMS disk and file names are not case-sensitive.

Oracle Rdb Database	To restore an Oracle Rdb database, click Rdb Database in the left-hand column, select the Rdb version, and enter the disk and file name using the following syntax: `DISK$1:[USER1_RDB]SITE_PERSONNEL.RDB` **Note**: Oracle Rdb disk and file names are not case-sensitive.
Oracle Rdb Storage Area	To restore an Oracle Rdb storage area, click RDB Storage Area, select the RDB version, and enter the disk name and file name using the following syntax: `DISK$1:[USER1_RDB]SITE_PERSONNEL.RDB/AREA=ACCOUNTING` **Note**: Oracle Rdb disk and file names are not case-sensitive.

Table 10–2 Entering The Correct Syntax For A Restore Request

File Type	Syntax

UNIX	To restore all the directory structures and files under a directory, click UNIX Disk or Directory Tree in the left-hand column and enter the following syntax:
	`/usr/`
	To restore a specific UNIX file, click UNIX Individual File in the left-hand column and enter the complete pathname:
	`/usr/usr1/file.c`
	Note:
	UNIX file names are case-sensitive. You must enter the file name exactly as it was originally created on the UNIX system.

NT	To restore an NT directory and all the files underneath it, click NT Disk or Directory Tree and enter the following syntax:
	`c:\usr\`
	To restore an individual NT file, click NT Individual File in the left-hand column and enter the file name using the following syntax:
	`c:\usr\usr1\file.c`
	Note:
	NT file names are not case-sensitive.

10.2.1 Restore Request Restrictions

ABS imposes the following restrictions when adding file names or disk names to a restore request:

- You cannot specify more than eight file names or disk names per restore request.
- Wildcard characters are only valid for backup agents that support wildcard characters, such as the OpenVMS BACKUP Utility.

Note

If you are restoring selective data using a wildcard character, you may not get the saveset you desire. This is because ABS will locate the first saveset that matches the wildcard file specification and use that saveset in the restore. Multiple savesets will not be used, and some of the data you require may not be restored. To be sure that you will get the files you require, use ABS LOOKUP to find the files and the save dates. Then use the /BEFORE or /AFTER qualifiers to specify a date for the restore. You may need to do multiple restores to restore the required data. Another alternative is to use an INCREMENTAL restore if you have full and incremental savesets to restore.

- Wildcard characters are not valid for an Oracle Rdb database or storage area:
  - Oracle Rdb database - You cannot specify wildcard characters for an Oracle Rdb disk name, directory specification, or file name. You must enter the name of the database exactly as it was entered on the save request.

    Example:

    ```
    DISK$USER1:[USER1_RDB]SITE_PERSONNEL.RDB
    ```

  - Storage area - A storage area requires the disk name, directory name, Rdb database name, and the area. Wildcard characters are not permitted.

    Example:

    ```
    DISK_NAME:[DIRECTORY_NAME]RDB_DATABASE.RDB/AREA=AREA_NAME
    ```

- NT and UNIX files - Restoring NT or UNIX files is determined by the path name entered in What Data to Restore option. To restore a UNIX root directory and all of its subdirectories, enter the following path name:

  Example:

  ```
  /usr/
  ```

- If you need to restore a complete directory structure for an NT client system, and the save request was NT Disk or Directory Tree, then you must perform the same type of restore operation. If you do not create the same type in this situation, ABS will not restore the files located in the top-level directory.

  Example:

  ```
  $ ABS SAVE “C:\TEST1”/OBJECT_TYPE=WINDOWS_NT_FILES_GTAR/SOURCE=NTNODE/FULL
  $ ABS RESTORE “C:\TEST1”/OBJECT_TYPE=WINDOWS_NT_FILES_GTAR/SOURCE=NTNODE -
  _$ /FULL
  ```

  You can, however, select NT Individual File to restore individual NT files saved from the NT Disk or Directory Tree type of save request.

- Bound volume set - To restore a bound volume set:
  - You must enter the What Data to Restore option exactly as it was entered on the save request.
  - You must use the Restore To option to restore a bound volume set. You must specify the entire list of disk names in the bound volume set. The number of disks specified in the output location must match the number of disks in the original bound volume set. See Section 10.5 for a description of the Restore To option.

Note

You can specify a comma-separated list of file or disk names DISK$USER1:,DISK$USER2:,DISK$USER3:

  - You either must dismount these output disks or mount them /FOREIGN.

- Concealed logical names - If the What Data to Restore option contains a concealed logical name, the concealed logical name must be a system-wide logical name or physical device name. Save and restore requests translate the logical names to the first concealed logical name found. However, if this concealed logical name is in the process table, access to the logical name is not available to ABS, and the restore operation will fail.

Recommendation:
Unless you are restoring an entire OpenVMS disk, specify the full path name. Include the disk name, directory name, and file name.

Examples:

```
DISK$1:[USER1]LOGIN.COM
```

If you are restoring an entire OpenVMS disk, specify only the disk name and include the trailing colon:

```
DISK$USER1:
```

10.2.2 Pre and Post- Processing Commands

Use the pre- and post- processing commands to submit commands (platform specific) that take action either before or after each file or disk name entered for the restore request.

Enter a command string for these options:

Preprocessing command:

```
$ @SHUTDOWN_DB.COM
```

Postprocessing command:

```
$ @STARTUP_DB.COM
```

ABS generates these logical names for use within a prologue or epilogue (pre- or post- processing command) for a SAVE or RESTORE request. A set of "_n" logicals will be generated for each include specification in the request. The series will begin with 1 and continue for each include specification within the request. These may be referenced in a command procedure which is executed as a prologue or epilogue for the save or restore request.

These logical names are defined in the process JOB table. They exist during the save request. Once the save request is complete the logicals will no longer be available.

ABS_OS_OBJECT_SET_n	Include specification.
ABS_OS_OBJECT_TYPE_n	Object type for the object set.
ABS_OS_DMT_n	Data movement type for the object set. Valid values are FULL, SELECTIVE, INCREMENTAL.
ABS_OS_INCREMENTAL_LEVEL_n	If the data movement type is INCREMENTAL, the ABS_OS_INCREMENTAL_LEVEL_n value defines the incremental level of the save request. The string will be in the format "Level n Operation" where n is the incremental level. If the data movement type is FULL the incremental level will be "Full Operation". If the data movement type is a base operation the incremental level will be "Base (Full) Operation". If the data movement type is selective the incremental level will be "Selective Operation".
ABS_OS_VOLUME_SET_n	Name of the volume set used for this request. If the request is to disk the value will be the disk and directory the saveset has been written to.
ABS_OS_START_RVN_n	The starting Relative Volume Number within the volume set that this object set references. The value of this logical will be 0 if the request is going to disk.

ABS_OS_LAST_RVN_n	The relative volume number within the volume set that was last used to save the saveset created by this object set. The value of this logical will be 0 if the request is going to disk. Note that this logical will not be valid for the prologue command. For the prologue command the value will be "Not yet determined" It will be defined for the epilogue command.
ABS_OS_START_FILE_POSITION_n	The starting file position of the saveset on the relative volume number within the volume set. This logical will indicate how many tapes marks to skip to find the start of the saveset on the volume. The value of this logical will be 0 if the request is going to disk.
ABS_OS_SAVESET_NAME_n	The name of the saveset that has been generated by ABS. This saveset contains the data saved based upon the object set.
ABS_OS_SAVESET_FORMAT_n	The format of the saveset. This will determine whether the saveset is in a VMS backup format, gtar format, or RMU backup format.
ABS_OS_STATUS_n	The ABS status of the include specification request.

The logical name ABS_OS_OBJECT_NUMBER is defined in the process table of the backup sub-process to indicate the serial number of the include spec (_n) being backed up in the current sub-process.

Restrictions:

- When entering the pre- or postprocessing command, an OpenVMS string is limited to 80 characters.
- You can only enter one pre- and post- processing command per restore request.
- A postprocessing command will execute only if the restore request completes successfully.

Default:
If you do not assign a string to this option, ABS does not execute any pre- or postprocessing commands.

10.2.3 Selection Criteria

Enter the specific selection criteria for the OpenVMS disk or file name the you are adding to the restore request. You can constrain the restore request by specifying a before or since date.

- Before - Restores the version of the file saved before the specified date.
- Since - Restores the version of the file saved since the specified date.
- Latest Copy - Restores the latest copy of the file saved.

10.2.4 Agent Qualifiers

Enter any backup agent specific qualifiers for the restore request. Backup agent qualifiers are determined by the type of file or disk that you are restoring, such as OpenVMS, Oracle Rdb, UNIX, or NT.

10.3 When

To execute the restore request, follow these steps:

Step 1. Click Start Time

Step 2. Select NOW, TOMORROW, NEVER, or OpenVMS Time.

If you select OpenVMS time, click the date box and enter the date and time in standard OpenVMS format. For example, to start the save request on March 13, 2000 at 2:00 p.m, use the following format:

```
13-MAR-2000 14:00
```

10.4 Where and How

ABS enables you to specify a storage policy to use for the restore request, and enables you to specify under what conditions to execute the restore request.

The storage policy option allows you to specify the name of the storage policy that contains the data that you want to restore. Specifying a storage policy name instructs ABS to search for the data in ABS catalog assigned to that storage policy.

Note

If multiple storage policies reference the same ABS catalog, and more than one storage policy contains the same file or disk name, ABS restores the most recent version of the file or disk even though it may not be located in the specified storage policy.

Default storage policy:
If you do not specify this option, ABS searches the default storage class, SYSTEM_BACKUPS.

An environment policy defines under what conditions to execute the restore request. For example, some conditions may be in what context to run the restore request (in the context of the user or ABS), who to notify when the restore request completes, under which error conditions to notify a user, and so forth. See Chapter 8, *Creating Environment Policies* for more details.

Requirement:
A restore request must use a storage policy and an environment policy. ABS provides default policies, but you can use different policies if you so choose. Click on Storage Policy and Execution Policy and select one of the policies displayed in the list box.

10.5 Restore To

The Restore To option enables you to specify an output location for the restored data. Use the following procedure:

Step 1. Click Restore To...

Step 2. Enter the output location for the restored data. The location consists of the following items:

- Disk Name - The disk name for the location of the restored data. For a bound volume set, specify each disk in the bound volume set in a comma-separated list.
- Directory Specification - The name of the directory where you want to restore the data.

  Recommendation:
  Do not specify the file name. Specify only the disk name and directory specification. ABS restores the file to its original name. This prevents multiple files from being restored on top of one another.

Restrictions:
If you are restoring a bound volume set, you must specify the output location for each disk device in the bound volume set.

Requirement:
Include the colon as part of the disk name.

Example:
```
DISK$USER1:,DISK$USER2:,DISK$USER3:
```

If you are restoring a multiple file Oracle Rdb database, you cannot specify a different output location. You can change the output location only for a single file Oracle Rdb database.

The directory specification for the Oracle Rdb database must already exist. The Oracle RMU Backup Utility cannot create a new directory.

Do not use the option for NT restore operations. The Restore To option is not valid for NT restore operations.

10.6 Restore Request Access Controls

Access Control option enables you to authorize other users access to the restore request, and to enable those users with specific access controls.

The default access is set to the user who creates the restore request. To add other users and access controls, see the procedure in Table 10–3:

Table 10–3 Enabling Access Control To A Restore Request

Step	Action
1.	Click Add...
2.	Add the node name and the user name: `NODE01::SMITH`
3.	To enable the desired access controls, click the box next the control. When selected, the box is highlighted: • Read - Users with Read access control can show the restore request. • Write - Users with Write access control can modify the restore request. • Delete - Users with Delete access control can delete the restore request. • Set - Users with Set access control can modify any attribute of the restore request, including access control. • Show - Users with Show access control can show the restore request • Control - Users with Control access control can modify the access control for the restore request, but not any of its other attributes.

10.7 Submitting the Restore Request

Once you have entered all the information for the restore request, click OK on the main window to submit the restore request to ABS policy database.

Result:
ABS displays the Submit Restore Request window, this window contains the basic information for the restore request. Click Submit to submit the restore request, or click Cancel to cancel the submit operation. If you cancel, the main window reappears and you can modify any information, or click Cancel to cancel creating the restore request.

11

Scheduling Requests

Save and restore requests are executed by calling the command procedure ABS$SYSTEM:COORDINATOR.COM with the request's universal ID as parameter 1. Typically this is done in a detached process created either by using OpenVMS Queue Manager or through a 3rd party scheduler program. ABS supports five options for scheduling requests which are all described in this chapter.

Note

The internal queue manager scheduling option is the only scheduling option available with an ABS-OMT license.

11.1 Setting the Scheduler Interface Option

During installation the user is asked to decide on the scheduler interface to be used by ABS for scheduling requests. The scheduler interface chosen is stored in the file ABS$SYSTEM:ABS$POLICY_CONFIG.DAT:

```
! Scheduler option, must be one of the list below:
!      (written by KITINSTAL at installation)
!  NONE
!  DECSCHEDULER
!  EXT_SCHEDULER
!  INT_QUEUE_MANAGER
!  EXT_QUEUE_MANAGER
!
ABS$SCHEDULER = INT_QUEUE_MANAGER
```

This file can be edited and the scheduler setting can be changed using a text editor. ABS needs to be restarted for the change to take effect.

11.2 Changing between Scheduler Interface Option

Before changing the scheduler interface option from either EXT_SCHEDULER or DECSCHEDULER you have to make sure that ABS jobs are no longer being scheduled using the old scheduler interface. Either delete all ABS jobs from the scheduler database or set the jobs on hold.

No preliminary change is necessary when switching from either INT_QUEUE_MANAGER or EXT_QUEUE_MANAGER because ABS jobs do not remain in the Queue Manager's database.

Once ABS has been restarted and the new scheduler interface option is either EXT_SCHEDULER or DECSCHEDULER you have to set the start time for all requests which should be scheduled. At this point all save requests should become jobs in the new scheduler's database.

If switching to either INT_QUEUE_MANAGER or EXT_QUEUE_MANAGER no extra step is necessary. ABS will call the Queue Manager when necessary to create the batch job for the request.

11.3 Scheduler Interface Option INT_QUEUE_MANAGER

This option causes ABS to call the OpenVMS Queue Manager to create, delete, modify and show jobs for save and restore requests. All jobs are queued to a batch queue called ABS$<node_name> (e.g. ABS$FUDGE) where node_name is the execution node name for the request. With this option the policy engine has a scheduler function active which calls the OpenVMS Queue Manager to create jobs for requests which are due to run.

For a request which is scheduled to run on a node which is not the current node and which is not in the current OpenVMS cluster, ABS sends a message to the ABS$COORD_CLEAN process on the remote node. The batch job for the request is then created, deleted or modified on the remote node by the ABS$COORD_CLEAN process.

No further setup is necessary to use this option. On your ABS VMS Client nodes, be sure that the ABS$COORD_CLEAN process is running and the ABS$<node_name> queue is working.

Information about ABS scheduling activities is logged into the ABS$LOG:ABS$POLICY_<node_name>.LOG file. To receive more information in the log file, you may define a logical:

```
$ DEFINE/SYSTEM ABS_SCHEDULER_LOGGING TRUE
```

An OPCOM message is also sent to the TAPE operator in case of ABS scheduling failures.

This is the default scheduler interface option for ABS.

11.4 Scheduler Interface Option EXT_QUEUE_MANAGER

This option causes ABS to execute the command procedure ABS$SYSTEM: ABS$EXT_QUEUE_MANAGER.COM. The command procedure uses DCL to interface with the OpenVMS Queue Manager to either create, delete, modify or show a batch job for a request. With this option the policy engine has a scheduler function active which calls the command procedure to create jobs for requests that are due to run.

The template file ABS$SYSTEM: ABS$EXT_QUEUE_MANAGER.TEMPLATE provided during installation should be copied to ABS$SYSTEM:ABS$EXT_QUEUE_MANAGER.COM and can be used as-is or, can be modified if necessary. However, caution needs to be taken if the template command procedure is modified. A failure in the command procedure could cause save or restore requests to fail. For debugging purpose, the command procedure is run in a subprocess of the ABS$POLICY process. It creates a logfile called ABS$LOG:ABS$EXT_QUEUE_MANAGER_<request_name>.LOG. This log may be used for troubleshooting problems with the command procedure.

Note

During installations, a new template file will be provided, however, your command procedure will not be overwritten.

For a request which is scheduled to run on a node which is not the current node and which is not in the current OpenVMS cluster, ABS sends a message to the ABS$COORD_CLEAN process at the remote node. The ABS$COORD_CLEAN process then spawns a subprocess to execute the local copy of ABS$SYSTEM: ABS$EXT_QUEUE_MANAGER.COM.

No further setup is necessary to use this option. On your ABS VMS Client nodes, be sure that the ABS$COORD_CLEAN process is running and the ABS$<node_name> queue is working

This option should be used as an alternative to INT_QUEUE_MANAGER if the OpenVMS Queue Manager should be used to execute requests but modifications are necessary to allow for

a more sophisticated schedules. For example a save request should never run on the first day of a month.

See the description in the template procedure about the input and output parameters.

Information about ABS scheduling activities is logged into the ABS$LOG:ABS$POLICY_<node_name>.LOG file. To receive more information in the log file, you may define a logical:

```
$ DEFINE/SYSTEM ABS_SCHEDULER_LOGGING TRUE
```

An OPCOM message is also sent to the TAPE operator in case of ABS scheduling failures.

11.5 Scheduler Interface Option EXT_SCHEDULER

This option causes ABS to execute the command procedure ABS$SYSTEM: ABS$EXT_SCHEDULER.COM. The command procedure uses DCL to interface with a 3rd party scheduler product to either create, delete, modify or show a scheduled job for a request.

The template file ABS$SYSTEM: ABS$EXT_SCHEDULER.TEMPLATE provided during installation is an example of how to interface with POLYCENTER Scheduler. The template file should be copied to ABS$SYSTEM:ABS$EXT_SCHEDULER.COM. This command procedure needs to be carefully adjusted to work with any other scheduler product. A failure in the command procedure could cause save or restore requests to fail. The command procedure is run in a subprocess of the ABS$POLICY process. It creates a log file called ABS$LOG:ABS$EXT_SCHEDULER_<request_name>.LOG. This log may be used for troubleshooting problems with the command procedure.

Note

During installations, a new template file will be provided, however, your command procedure will not be overwritten.

In contrast to EXT_QUEUE_MANAGER, ABS calls this interface only once to create a new job for a request. ABS assumes that the external scheduler does its own rescheduling of requests and can schedule request to run on a remote node.

Information about ABS scheduling activities is logged into the ABS$LOG:ABS$POLICY_<node_name>.LOG file. To receive more information in the log file, you may define a logical:

```
$ DEFINE/SYSTEM ABS_SCHEDULER_LOGGING TRUE
```

An OPCOM message is also sent to the TAPE operator in case of ABS scheduling failures.

This option should be used to tie ABS into an existing scheduler product.

11.6 Scheduler Interface Option DECSCHEDULER

This option causes ABS to call the POLYCENTER Scheduler V2.1b (DECscheduler) programming interface to create, delete, modify or show a job for a request.

Refer to the POLYCENTER Scheduler documentation on how to setup and manage the scheduler.

This option is qualified to be used with POLYCENTER Scheduler V2.1b only. However, it may be used with different versions of DECscheduler but without any further guarantee to work correctly.

11.7 Scheduler Interface Option NONE

This option causes ABS to NOT call any scheduler to create, delete, modify or show a job for a request.

The SHOW SAVE/SYMBOLS and SHOW RESTORE/SYMBOLS commands provide a means of obtaining scheduling information for a request and a means to execute that request through the following DCL command line sequence:

```
$ ABS SHOW SAVE my_request/SYMBOLS
$ SUBMIT ABS$SYSTEM:COORDINATOR /PARAMETER="''ABS_UID'"/USER=ABS
```

This executes the request 'my_request' immediately on the current node.

This option should be used if neither of the other options can be used or there is a simple requirement for executing requests.

11.8 Scheduler Interface Internals

ABS calls the scheduler interface from process ABS$POLICY.

Option INT_QUEUE_MANAGER: ABS uses the programming interface to the OpenVMS Queue Manager. A scheduler thread is started within the ABS$POLICY process to submit due requests to the OpenVMS Queue Manager. The request will be submitted to batch queue ABS$<execution_node>. If the batch queue is available on the local node or is within the OpenVMS cluster the ABS$POLICY process calls the local Queue Manager. For remote nodes the ABS$POLICY process forwards the request via DECnet to the ABS$COORD_CLEAN process running on the execution node. The ABS$COORD_CLEAN process then submits the request to the local batch queue ABS$<execution_node>.

Failures to call the OpenVMS Queue Manager will be logged in ABS$LOG:ABS$POLICY_<node_name>.LOG or ABS$LOG:ABS$COORD_CLEANUP_<node_name>.LOG. The scheduler thread in the ABS$POLICY process sends OPCOM messages to operator TAPES if it fails to schedule a request.

Option EXT_QUEUE_MANAGER: This option uses the same method as INT_QUEUE_MANAGER to schedule jobs locally or remote. But instead of calling the programming interface of the OpenVMS Queue Manager, a subprocess is created to run the command procedure ABS$SYSTEM:ABS$EXT_QUEUE_MANAGER.COM. The command procedure is responsible to issue the DCL commands to create, delete, modify and show batch jobs. Also the command procedure has to return status about the commands and in some cases additional information. See the command procedure file for more details.

Failures to execute the command procedure will be logged in ABS$LOG:ABS$POLICY_<node_name>.LOG or ABS$LOG:ABS$COORD_CLEANUP_<node_name>.LOG. The scheduler thread in the ABS$POLICY process sends OPCOM messages to operator TAPES if it fails to schedule a request. Each activation of the command procedure creates a logfile ABS$LOG:ABS$EXT_QUEUE_MANAGER_<request_name>.LOG. The request name portion of the logfile name maybe truncated to a valid OpenVMS file specification.

Option EXT_SCHEDULER: This option uses the same method as EXT_QUEUE_MANAGER to interface with the scheduler. A subprocess is created to run the command procedure ABS$SYSTEM:ABS$EXT_SCHEDULER.COM. The command procedure is responsible to issue the DCL commands to create, delete, modify and show jobs for a 3rd party scheduler product. Also the command procedure has to return status about the commands and in some cases additional information. See the command procedure template file ABS$SYSTEM:ABS$EXT_SCHEDULER.TEMPLATE for more details. In contrast to option

EXT_QUEUE_MANAGER, ABS assumes that the 3rd party scheduler product reschedules all requests locally and remote. So ABS will not call the scheduler if a request is due to run.

Failures to execute the command procedure will be logged in ABS$LOG:ABS$POLICY_<node_name>.LOG. Each activation of the command procedure creates a logfile ABS$LOG:ABS$EXT_SCHEDULER_<request_name>.LOG. The request name portion of the logfile name maybe truncated to a valid OpenVMS file specification

Option DECSCHEDULER: ABS uses the programming interface to POLYCENTER Scheduler V2.1b. ABS calls this scheduler only once locally to create any request since this scheduler product can reschedule jobs locally and remote.

Failures to call the POLYCENTER Scheduler will be logged in ABS$LOG:ABS$POLICY_<node_name>.LOG.

12

Modifying and Deleting ABS Policies and Requests

ABS allows you to modify and delete ABS policies and requests that already exist in ABS policy database. To modify polices and requests, see the requirements listed in Table 12–1.

Table 12–1 Requirements for Modifying and Deleting Policies and Requests

To modify or delete...	You must ...
Storage Policies	Have ABS_CREATE_STORAGE_CLASS access right identifier enabled on your user process
Environment Policies	Have ABS_CREATE_EXECUTION_ENV access right identifier enabled on your user process
Any policy or request	• Have the SYSPRV OpenVMS privilege enabled on your user process • Have the CMKRNL OpenVMS privilege enabled on your user process • Be logged onto ABS server node.

Note

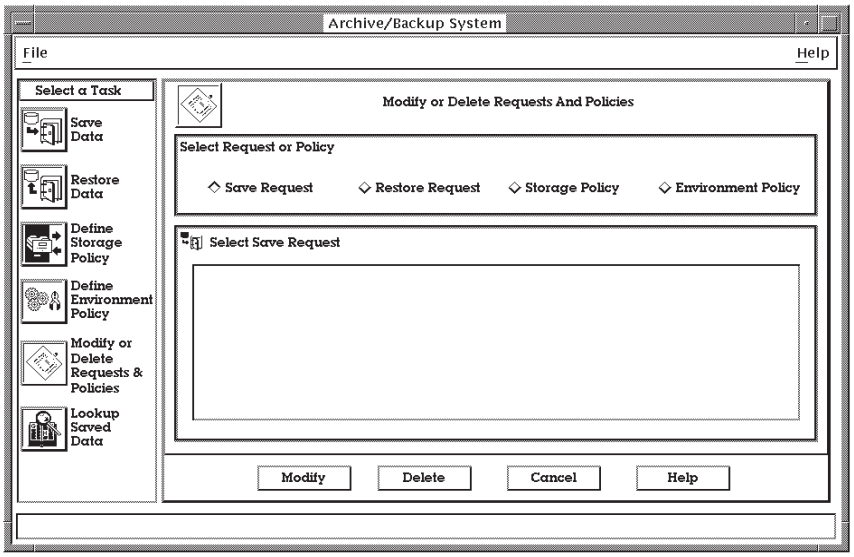
The creation of storage and environment policies are not available with an ABS-OMT license. Therefore if you delete the default policies, you can not recreate the policies with the GUI. Use the following command to recreate the storage and environment policies:

```
$ RUN ABS$SYSTEM:LITE_DB_INIT.EXE
```

To modify or delete an ABS policy or request, invoke ABS GUI as described in Chapter 6, *Displaying ABS Graphical User Interface*. Click Modify or Delete Requests & Policies.

Result:
ABS displays the Modify or Delete Requests And Policies window, illustrated in Figure 12–1.

Figure 12–1 Modify or Delete Requests And Policies Window



12.1 Select Request or Policy

Use the following procedure to modify or delete an exiting ABS request or policy.

Table 12–2 Modifying or Deleting an ABS Policy or Request

Step	Action
1.	Click the button next to the type of request or policy that you want to modify or delete in the Select Request or Policy area. **Result**: Depending upon your selection, a list of policies or requests are displayed in the lower area of the window.

Table 12–2 Modifying or Deleting an ABS Policy or Request

Step	Action	
2.	Select the request or policy from the list; click Modify or Delete.	
	IF you selected ...	**THEN ...**
	Delete	ABS displays a confirmation box. Click **Yes** to delete the request or policy. Click **Cancel** to abort the delete operation. If a storage policy or execution policy has catalog references to it, ABS will display a window and ask if you are sure you want delete the policy. If you answer yes and delete a storage policy that has catalog references, you will not be able to restore the data that was saved using that particular storage policy. ABS will display a message that warns you about this situation.
	Modify	ABS displays the modification screen for the policy or request you selected. Depending upon the type of request or policy that you are modifying, refer to one of the following Chapters: Chapter 7, *Creating Storage Policies* Chapter 8, *Creating Environment Policies* Chapter 9, *Creating Save Requests* Chapter 10, *Creating Restore Requests*
3.	When you have completed the modification or deletion, click OK. **Result**: ABS displays an informational window that contains the current settings for the request or policy.	
4.	Click Submit to modify the request or policy, click Cancel to abort the modify operation.	

13

Looking Up Saved Data

From the Lookup Saved Data option, you can find data that was previously saved using ABS or SLS. The following sections describe how to use the Lookup Saved Data option.

13.1 Data to Lookup

Click the box next to Data to Lookup and enter the disk or file name that you want to find. You can enter up to eight disk or file names as a comma-separated list.

Recommendation:
It is recommended that you do not mix file types in one lookup operation. For example, do not specify OpenVMS disk or file names along with UNIX or NT disk or file names in one lookup operation.

The file name syntax is dependent upon the file type, described in Section 13.1.1

13.1.1 File Type

This option allows you to specify the type of file to search for. The default is All, but this type of lookup is not recommended. Instead, click the box next to File Type and select the type of file, such as VMS files. See Section 13.1.2 to determine how to correctly enter the lookup syntax.

13.1.2 Entering the Correct Lookup Syntax

Table 13–1 describes how to enter the correct syntax for the lookup operation.

Table 13–1 Entering The Correct Syntax For A Lookup Operation

File Type	Syntax
All	Any of syntax definitions per file types described in this table. **Note**: The lookup operation takes longer if you select this option. Because the lookup operation is not constrained to a specific file type, the search takes much longer because it is looking through all file types before locating the data.

VMS Files	To find an OpenVMS disk name, select VMS files and enter the disk name:
	`DISK$USER1:`
	
	To find an OpenVMS individual file, enter OpenVMS file name:
	`DISK$USER1:[DIRECTORY]LOGIN.COM`
	
	Note:
	OpenVMS file names are not case-sensitive.

Table 13–1 Entering The Correct Syntax For A Lookup Operation

File Type	Syntax

UNIX	To find an individual UNIX file, enter the syntax as shown in the following example:
	`/usr/users/smith/login`
	To find a UNIX directory, enter the following syntax:
	`/usr/users/abs/`
	To use a wildcard character (*) to find UNIX files, enter the following syntax:
	`/usr/users/*`
	Result:
	This example will find all files and directories saved under the directory /usr/users. The behavior of a UNIX wildcard lookup operation is similar to the UNIX command “ls -r /user/users/*”.
	Note:
	UNIX file names are case-sensitive. You must enter the file name exactly as it was created on the UNIX system.

NT	To find an NT file, use the following syntax:
	`C:\USERS\SMITH\TEMP.TXT`
	To find an NT directory, use the following syntax:
	`C:\USERS\SMITH\`
	To find NT files using a wildcard character (*), use the following syntax:
	`C:\USERS\*`
	Result:
	This example will find all files and directories saved under the directory C:\USERS\. The behavior of an NT wildcard lookup operation is similar to the Windows NT command “DIR \USERS* /S” on the C drive.
	Note:
	NT file names are not case-sensitive.

Oracle Rdb Database	To find an Oracle Rdb database name, select RDB_Vn.n_DATABASE and enter the database file specification:
	`DISK$1:[ISER1_RDB}SITE_PERSONNEL.RDB`
	Note:
	Oracle Rdb database file names are not case-sensitive.
	Oracle Rdb Storage Area To find an Oracle Rdb storage area, select RDB_Vn.n_STORAGE_AREA and enter the storage area specification using the following syntax:
	`DISK$1:[USER1_RDB]SITE_PERSONNEL.RDB/AREA=ACCOUNTING`
	Note:
	Oracle Rdb Storage Area file names are not case-sensitive.

13.1.3 Node of Original Data

This option enables you to specify the node name where saved data originally resided. Use the following procedure to enter the node name:

Step 1. Click the box next to Node of Original Data.

Step 2. Click one of the node names displayed in the node name list, or click Other and enter the node name in the box.

13.1.4 Storage or Catalog Name

If you know the storage policy name or catalog name where the saved data resides, you can constrain the search to that particular storage policy or catalog.

Restriction:
This option is mutually exclusive. You can select either a storage policy name or a catalog name, but not both.

Note

You can assign the same catalog name to multiple storage policies. When more than one storage policy contains the same saved data, and those storage policies reference the same catalog name, ABS displays the saved data regardless of which storage policy the data references.

13.1.5 Archived Dates to Search

Archived Dates to Search option enables you to constrain the lookup operation to an exact specified date, before a specified date, or after a specified date.

See Table 13–2 to constrain the lookup operation by a specified date.

Table 13–2 Finding Saved Data By Date

To find data saved . . .	Then . . .
On an exact date	1. Click the selection box next to Date Match and select On Exact Date 2. Enter the exact date in the box next to the Date Archived option. To enter the correct date format, see Appendix B.
Before a specific date	1. Click the selection box next to Date Match and select On Or Before 2. Enter the before date in the box next to the Date Archived option. To enter the correct date format, see Appendix B.
After a specific date	1. Click the selection box next to Date Match and select On Or After 2. Enter the after date in the box next to the Date Archived option. To enter the correct date format, see Appendix B.

Restriction:
If you are looking for data that was saved using SLS, only the On Exact Date option is valid. This is because SLS does not supply any other dates.

13.2 Submitting the Lookup Operation

To submit the lookup operation once you have entered the correct data, use the following as procedure:

- Click OK to submit the lookup operation.
- Once ABS has located the saved data, ABS displays information about the saved data in the Saved Data Found area.
- Click Cancel to cancel the lookup operation.

14

Monitoring Job Status

ABS allows you to monitor the status of an active ABS job from the GUI. To view the status of an active job, use the following procedure:

Step 1. From the system prompt, define the following logical name:

```
$ DEFINE/SYSTEM ABS$MONITOR_UPDATE_INTERVAL 30
```

Result:
Causes ABS coordinator to update monitor information about any active jobs every 30 seconds.

Step 2. Display ABS GUI.

Step 3. Click Job Status from the main ABS GUI window.

Step 4. Click the job name that you want to check.

Step 5. Click Show Status.

Result:
ABS displays the status of the job.

Step 6. Click Stop if you want to stop displaying the job status. You can select another job as described in Step 4. and Step 5.

Step 7. Click Close when you are finished.

15

Creating ABS Catalogs

An ABS catalog contains historical information about ABS save operations. This historical information includes the location of data that was saved using ABS. For this purpose, ABS provides a catalog named ABS_CATALOG.

Most businesses can operate efficiently using only ABS_CATALOG provided by ABS. However, your business needs may require you to create additional catalogs. Some of those business needs may be:

- Speed of record insertion
- Speed of lookup operations
- Segregation of saved data
- Restoring data that was previously saved using Storage Library System for OpenVMS (SLS)

This chapter contains the following information:

- Creating ABS catalogs
- Creating ABS catalogs for the purpose of restoring data saved using SLS
- Creating ABS staging catalogs
- Showing catalog objects
- Improving catalog performance

15.1 Creating An ABS Catalog

To create an ABS catalog, follow these steps:

Step 1. Define the following symbol to run ABS catalog utility:

```
$ CATALOG_OBJ :== $ABS$SYSTEM:ABS$CATALOG_OBJECT
```

Step 2. Invoke the catalog utility symbol:

```
$ CATALOG_OBJ
```

After invoking the utility, the catalog utility program prompts for the following information:

```
Function (CREATE, DELETE,MODIFY,[SHOW]):
Catalog name: PRIVATE_CATALOG
Catalog type (BRIEF,SLS,FULL_RESTORE, [VOLUME_SET]):
Catalog owner [current_node::current_username]:
Use faster staging catalog operation ([YES],NO):
Catalog location [ABS$CATALOG]:
```

Default:
If you do not specify an answer to any of the options (except the catalog name), the catalog object utility selects the defaults (enclosed in square brackets ([])). If ABS$CATALOG points to a search list, the catalog files will be created in the location pointed to by the first element in the search list.

Result:

- Places a new ABS catalog object named PRIVATE_CATALOG in the catalog object database
- Allows you to select a catalog object type (either BRIEF, SLS or FULL_RESTORE). See section below if you select SLS.
- Specifies the owner as the current user on the current node unless otherwise specified.
- Allows you to enable faster staging catalog operation. See description below.
- The catalog files are placed at location ABS$CATALOG (a logical name search list).

  You can specify a different device and directory for the new catalog files. To use such a catalog you have to add a new search list element to the logical name ABS$CATALOG in ABS$SYSTEM:ABS$SYSTARTUP.COM. You either restart ABS or, if you do not want to restart ABS at this point just modify the current definition of this logical using the DCL DEFINE command.

  For example:

```
$ DEFINE/SYSTEM/EXECUTIVE/NOLOG ABS$CATALOG -

ABS$ROOT:[CATALOG],DISK$BACKUPS:[ABS_CATALOGS]
```

  This allows catalog files to be located in DISK$BACKUPS:[ABS_CATALOGS].

  Restriction:
  If you create an SLS ABS catalog type, you cannot enable staging.

- Creates the catalog file in ABS$CATALOG directory (not for SLS catalogs).

  Requirement:
  You must create a catalog on the node where the data resides (the data that you are going to back up), even if the data is going to be backed up to a drive on a remote node. This means you must run the catalog object utility on the node where the backup operation will be originated.

  Example:

IF ...	AND ...	THEN ...
You are going to back up the data on NODEA to a drive on NODEB	The storage policy uses the catalog named PRIVATE_CATALOG	PRIVATE_CATALOG must reside on NODEA.

15.1.1 Creating a BRIEF type catalog

The BRIEF catalog type stores all information about save requests performed and all files saved. It allows individual file lookups and restores. This is the default catalog type.

15.1.2 Creating a FULL_RESTORE type catalog

The FULL_RESTORE catalog type only stores information about save requests performed. No information about individual filenames are stored in the catalog. The size of a FULL_RESTORE catalog is drastically smaller than the BRIEF catalog type but you cannot restore individual files. Save requests using this catalog type must be of type FULL and only specify a disk name. Staging does not apply to these catalogs.

You can still use the backup agent outside of ABS to do a selective restore.

For example:

```
$ BACKUP MKA500:01MAY20001234567. /SELECT=[MyDir]MyFile.Dat  *
```

restores the file [MyDir]MyFile.Dat from save set 01MAY20001234567 to the current directory.

To view information about saved disks in a FULL_RESTORE catalog, use the ABS REPORT SAVE_LOG command. The report shows you the volume ID and save set name used.

When a save request uses a FULL_RESTORE type of catalog, the following message is displayed in the save request log file:

"Full_Restore catalog type, individual object names will not be logged"

Restrictions:

An ABS FULL_RESTORE catalog imposes the following restrictions:

- You may only use a FULL_RESTORE catalog to do FULL save requests and FULL restore requests.
- You may not view the saved file information using the ABS LOOKUP command.

15.1.3 Creating An SLS Type Catalog

ABS allows you to create ABS catalogs solely for the purpose of restoring data saved using SLS. These catalogs are not maintained in ABS database and are used only for restore operations and not for save operations.

ABS catalogs that are created for SLS provide the following features:

- The ability to perform a ABS selective restore operation for data that was previously saved using SLS.
- The ability to perform a ABS lookup operation to search for data objects in an existing SLS history file.
- The ability to specify wildcard characters, such as an asterisk (*) or percent sign (%), for a ABS lookup operation that searches SLS history sets. You can also specify the ellipsis within square brackets ([...]).

Restrictions:
An ABS catalog created for SLS restore operations imposes the following restrictions:

- You cannot restore Oracle Rdb databases saved using SLS.
- You cannot perform a full restore operation on an image backup that used SLS.
- You cannot perform a full restore operation with subsequent incremental backup operations that used SLS.

To create a catalog for restoring data that was saved using SLS, see the procedure in Table 15–1

Table 15–1 Creating an ABS Catalog For SLS Restores

Step	Action

1.	Create a new catalog by invoking the catalog object utility:
	`$ CATALOG_OBJ :== $ABS$SYSTEM:ABS_CATALOG_OBJECT`
	`$ CATALOG_OBJ`

2.	Specify the SLS history set name as the name of the catalog, for example: Catalog name: MY_HIST **Notes**: • The catalog name must match the SLS history set name. • For each SLS history set name from which you want to perform a lookup operation, you must create a corresponding ABS catalog. • For each SLS history set name from which you want to perform a restore operation, you must create a corresponding ABS catalog and storage policy.
3.	Enter the SLS catalog type: `Catalog type ([BRIEF],SLS,FULL_RESTORE): SLS`
4.	Do not enable staging: `Use faster staging catalog operation ([YES]/NO): NO`
5.	If you plan to perform a restore operation, create a storage policy that: • References the new catalog name (same as the SLS history set name). • Specifies a media type. • Enables READ only access control. Because this catalog is not intended for save operations, do not enable WRITE access control.

15.1.4 Creating a VOLUME_SET type catalog

TheVOLUME_SET catalog type stores all information like the BRIEF catalog type. However, the catalog file for the instance records (s. 15.6.1 Technical Details) is made up of multiple files: one file per volume set. Wildcard lookups take slightly longer on VOLUME_SET type catalogs compared to a BRIEF type catalog. But VOLUME_SET type catalogs allow for faster catalog cleanups because now a whole file is deleted instead of reading the whole file and deleting expired entries. By deleting a complete instance file of a catalog the catalog does not grow in size as much as the BRIEF catalog type does.

A VOLUME_SET catalog type cannot be used for FILES_11 archive types.

15.1.5 Creating a Catalog using Staging Operation

ABS allows you to enable staging for a ABS catalog. A catalog that provides staging improves the performance of the save operation because the catalog entry for a saved file is first written to a sequential disk file in ABS$CATALOG. Once the backup operation has completed a separate process moves the entries from the staging catalog file to the final catalog (the catalog name specified in the storage class associated with the save request).

The final catalog does not contain the information about the save operation until the staging process has completed. If you request a backup operation and immediately look in the final catalog,

the entries may not be available, yet. The backup operation and the staging process must complete before the currently saved files can be looked up in the catalog.

You can always modify the staging setting for an existing catalog using the ABS_CATALOG_OBJECTS utility. The use of this feature is highly recommended.

15.2 Showing a Catalog

To view a catalog definition, use the ABS_CATALOG_OBJECT utility and select the SHOW function.

To view the contents of a catalog use the ABS LOOKUP command or select the lookup function in the GUI. The DCL commands and GUI operations that search the catalog files require the user to have read access to those catalog files. This is because these operations are executed in the context of the user, and not in the context of the ABS account.

With the default catalog protection, a user would have to be logged into a system account or ABS account, or have elevated privileges that enables the user read access to the files (such as BYPASS, SYSPRV, or READALL).

Examples of such operations are using ABS LOOKUP command and ABS REPORT SAVE_LOG DCL command, or using the LOOKUP option from the GUI.

15.3 Modifying a Catalog

If you choose to MODIFY a catalog object, you will be prompted for the fields in the same manner as CREATE. To be sure of the values to set, you should first do a SHOW of the catalog so that you will not inadvertently change fields which you do not want to change.

Restriction:
You may not modify the CATALOG TYPE. To change the catalog type you must first delete and then recreate the catalog.

15.4 Deleting a Catalog

The DELETE option will delete the catalog object and the catalog files located in the ABS$CATALOG directory. If you do not wish to delete the actual catalog files, then copy them to another name or location prior to executing the DELETE function.

15.5 Improving Catalog Performance

ABS provides a catalog conversion command procedure that improves the target catalog update performance by doing a file-to-file conversion. A target catalog is the final catalog where ABS entries reside, while a staging catalog is a temporary catalog that increases ABS performance during save operations. By converting the target catalogs, you can improve ABS target catalog update performance. This describes how to convert ABS catalogs. For additional improvement of the catalog update performance, you can also move the target catalogs to a different disk by defining a system level search list logical for ABS$CATALOG in ABS$SYSTARTUP.COM.

15.5.1 Converting ABS Catalogs

Run the conversion command procedure for each individual catalog.

```
$ @ABS$SYSTEM:ABS$CONVERT_CATALOG MyCatalog
```

The command procedure creates a new copy of the catalog files. The new and the old files will reside in the same directory. The command procedure also allows you to move the converted files to a different disk or directory.

```
$ @ABS$SYSTEM:ABS$CONVERT_CATALOG <catalog_name>  <disk:>[<dir>]
```

15.5.2 Moving Target Catalogs to a Different Disk

You can also improve the target catalog update performance by moving the target catalogs to a different disk.

To do this, follow the procedure in Table 15–2.

Table 15–2 Moving Target Catalogs to a Different Disk

Step	Action
1.	Create a system-level search list logical for both ABS$CATALOG and logical names: `$ define/sys abs$catalog abs$root:[catalog],<disk:>[<dir>]` This search list includes the default catalog location as well as the disk (or disks) that can be used for the target catalogs. Update the logical definition in SYS$STARTUP:ABS$SYSTARTUP.COM.
2.	Manually copy the target catalogs from ABS$ROOT:[CATALOG] to <disk:>[ABS_CATALOG] or use the conversion procedure: `$ @ABS$SYSTEM:ABS$CONVERT_CATALOG <catalog_name>  <disk:>[<dir>]`

3.	Once the catalog files are copied to the new location, delete the following catalog files from their original location:
	`ABS$CATALOG:<catalog_name>_%AOE.DAT`
	`ABS$CATALOG:<catalog_name>_*AOE_INSNC.DAT`
	`ABS$CATALOG:<catalog_name>_%TLE.DAT`

However, ABS always writes the staging catalogs to the first element in the ABS$CATALOG search list, and then it writes the entries for the target catalogs to a different disk.

15.5.3 Moving Staging Catalog Entries

With staging catalog enabled ABS creates a command procedure at the end of a save request. A separate process is created which executes this command procedure to move all entries from the staging catalog to the final catalog. If all entries have been moved successfully the command procedure is deleted. If ABS failed to execute the command procedure you can run it manually. To do this, enter the following command at the system prompt on the node the save request was executed:

```
$ @ABS$CATALOG:<catalog_name>_m_n.COM
```

To determine which file to execute, search your save request log files in ABS$LOG to find the file names for the staging files.

The save request ABS$LOG:<save_request_name>.LOG file will contain the following information:

```
COORDINATOR: Staging process PID : 00006505
COORDINATOR: Staging catalog     : ABS$CATALOG:ABS_CATALOG_4.STG;1
COORDINATOR: Staging procedure   : ABS$CATALOG:ABS_CATALOG_4_1.COM;1
COORDINATOR: Staging logfile     : ABS$LOG:ABS_CATALOG_4.LOG
```

15.6 Sizes of Catalog Files

The ABS catalog files will grow as you continue to execute save requests which use those catalogs. The sizes depend on the number of files saved and the retention period used. For as long as the retention period has not expired more entries will be added to the catalog. Once the retention period is reached the daily ABS_CLEAN_CATLG_<node_name> batch job will remove expired

entries from the catalog. So the more files you save and the longer you want to keep the archived data the larger the catalog files.

Be sure that you consider this information when creating catalogs and assigning retention values to your storage classes. It may be best to create separate catalogs for each storage class, if the retention period is different. For example, you may create a storage class called MONTHLY_SAVE_SC, with a retention period of one month. Create a catalog to be used by that storage class. The catalog size will grow for one month and then maintain its size. But for your storage class, YEARLY_SAVE_SC, with a retention periods of a year, use a different catalog.

Please note that ABS catalogs are implemented as RMS indexed files, therefore the ABS catalog may continue to grow even after reaching the retention period, it is recommended to run the ABS$CONVERT_CATALOG procedure on the catalog to reclaim disk space.

If you have multiple catalogs, it will be easier for you to move catalogs to different disks if the size exceeds available space or do regular maintenance by running the ABS$SYSTEM:CONVERT_CATALOG command procedure.

15.6.1 Technical Details

ABS can have multiple catalogs. Each catalog is comprised of three RMS Indexed Sequential Files:

- <catalog_name>_%TLE.DAT - Transaction Log Entry
- <catalog_name>_%AOE.DAT - Archive Object Entry – not used for FULL_RESTORE catalog type
- <catalog_name>_*AOE_INSNC.DAT - Archive Entry Object Instance - not used for FULL_RESTORE catalog type, one file per volume set if VOLUME_SET catalog type

These files must reside in the same directory. Different catalogs can be in different directories or different disk volumes.

The **Transaction Log Entry** file contains two entries per save request executed. It contains among other data the save set name, the tape's volume ID and the expiration date of the save set. Depending on record compression the average record size on disk is about 300 bytes.

The **Archive Object Entry** file contains one entry for each file backed up. It contains among other data the device and file name. Depending on record compression and depending on actual filename sizes the average record size on disk is about 300 bytes.

The **Archive Object Entry** Instance file contains an entry for every time a file is backed up. It does not contain the filename but a back pointer to the record in the AOE. Depending on record compression the average record size on disk is about 200 bytes. For a VOLUME_SET catalog type there is one file per volume set in use. The volume set name is part of the instance file name.

15.7 What size is the ABS catalog?

TLE: This grows to the average size of how many save requests are active.

- This file does not have size problems
- Low volatility to deletes
- 300 bytes times number of active save requests times retention period in days + some record overhead.

AOE: This grows to the number of files that are actively being backed up

- Medium volatility to deletes

- 300 bytes times number of active files + some record overhead

AOE_INSNC: This can grow very large.

- Sized is based on how many files are being backup up and how long the retention time on the file is.
- High volatility to deletes.
- 200 bytes times average number of files backed up per day times the retention period in days.

Example1:

- 1 disk volume with 40,000 files
- full saves every week (40,000 files)
- incrementals 6 times a week (estimate 2,000 files/day)
- retention is 30 days for all backups
- TLE 300 X 7 X 30 = 63K bytes
- AOE: 300 X 40,000 = 12 MB
- AOE_INSNC: 200 X 7428 X 30 = 44 MB

Example 2:

- 1 disk volume, 40,000 files
- full saves every night (40,000 files)
- retention is 30 days for all backups
- TLE: small
- AOE: 300 X 40,000 = 12 MB
- AOE_INSNC: 200 X 40,000 X 30 = 240 MB

Example 3:

- 10 disk volumes, total of 400,000 files
- full saves every week (400,000 files)
- incrementals 6 times a week (20,000 files)
- retention is 30 days for all backups
- TLE: small
- AOE: 300 X 400,000 = 120 MB
- AOE_INSNC: 200 X 74285 X 30 = 445 MB

Example 4:

- 10 disk volumes, 400,000 files,
- full saves every night (400,000 files)
- retention is 365 days for all backups
- TLE: small compared to rest
- AOE: 300 X 400,000 = 120 MB
- AOE_INSNC: 200 X 400,000 X 365 = 29 GB

…and if you had 100 volumes: AOE_INSNC is 292 GB!!!

As you can see from example 4 catalogs can become quite large. Changing the backup schedule so that less files are saved and using shorter retention periods help to maintain smaller catalogs. If this cannot be achieved extra disk space should be reserved for the ABS catalogs with space for future expansion.

Part II

MDMS Operations

This part of the *Archive Backup System for OpenVMS Guide to Operations* contains information abou the Media and Device Management Services for OpenVMS (MDMS).

16

What is MDMS?

This chapter starts by describing the Media and Device Management Services software (MDMS)' management concept and its implementation. Following that is a description of the product's internal interfaces.

Note

User interfaces are described in the following chapter.

Media and Device Management Services V3.2A (MDMS), can be used to manage locations of tape volumes in your IT environment. MDMS identifies all tape volumes by their volume label or ID. Volumes can be located in different places like tape drives or onsite locations. Requests can be made to MDMS for moving volumes between locations. If automated volume movement is possible - like in a jukebox (tape loader, tape library) - MDMS moves volume/s without human intervention. MDMS sends out operator messages if human intervention is required.

MDMS allows scheduled moves of volumes between onsite and offsite locations (e.g. vaults). Multiple nodes in a network can be setup as an MDMS domain. Note that:

- all nodes in a domain access one MDMS database
- all MDMS objects like volumes and drives are described in the MDMS database

MDMS is a client/server application. At a given time only one node in an MDMS domain will be serving user requests and accessing the database. This is the *database server.* All other MDMS servers (which are not the database server) are clients to the database server. All user requests will be delegated through the local MDMS server to the database server of the domain.

In case of failure of the designated database server, MDMS' automatic failover procedures ensure that any of the other nodes in the domain, that has the MDMS server running, can take over the role of the database server.

16.1 MDMS Objects

MDMS manages all information in its database as *objects*. Table 16–1 lists and describes the MDMS objects.

Table 16–1 MDMS Object Records and What they Manage

This Object Record...	Meets the Need to...
Domain	Manage domain-wide operating parameters. MDMS creates this object record automatically.
Node	Describe a node in the MDMS domain. It defines the node's network names.You cannot operate MDMS without Node object records.
Group	Group node object records. Groups are a convenient shortcut to specify a list of nodes.

Table 16–1 MDMS Object Records and What they Manage

This Object Record...	Meets the Need to...
Location	Describe a location in your environment. A location can be the name of a building, a room or a facility.
Request	Handle all MDMS operations initiated by a user or an application.
Drive	Describe an OpenVMS drive to MDMS.
Jukebox	Describe a tape loader or tape library to MDMS.
Magazine	Describe a tape magazine to MDMS. The use of *magazine objects*, is optional even if magazines are used in reality.
Media Type	Describe the different media types represented by volumes.
Pool	Describe a group of volumes. Pools control which user has access to volumes in a group.
Volume	Describe an individual magnetic tape medium.

MDMS tries to reflect the true states of objects in the database. MDMS requests by users may cause a change in the state of objects. For some objects MDMS can only assume the state, for example: that a volume has been moved offsite. Wherever possible, MDMS tries to verify the state of the object. For example if MDMS finds a volume that should have been in a jukebox slot, in a drive, it updates the database with the current placement of the volume.

16.2 MDMS Interfaces

MDMS provides an internal callable interface to ABS and HSM software. This interfacing is transparent to the ABS or HSM user. However some MDMS objects can be selected from ABS and HSM.

MDMS communicates with the OpenVMS OPCOM facility when volumes need to be moved, loaded, unloaded, and for other situations where operator actions are required. Most MDMS commands allow control over whether or not an OPCOM message will be generated and whether or not an operator reply is necessary.

MDMS controls jukeboxes by calling specific callable interfaces. For SCSI controlled jukeboxes MDMS uses the MRD/MRU callable interface. For StorageTek jukeboxes MDMS uses DCSC. You still have access to these jukeboxes using the individual control software but doing so will make objects in the MDMS database out-of-date.

17

MDMS Configuration

The Media and Device Management Services Installation and configuration guide provides information about establishing the MDMS domain configuration. The information in this chapter goes beyond the initial configuration of MDMS, explaining concepts in more detail than the product installation and configuration guide. This chapter also includes procedures related to changing an existing MDMS configuration.

The major sections in this chapter focus on the MDMS domain and its components, and the devices that MDMS manages.

A sample configuration for MDMS is shown in Appendix E.

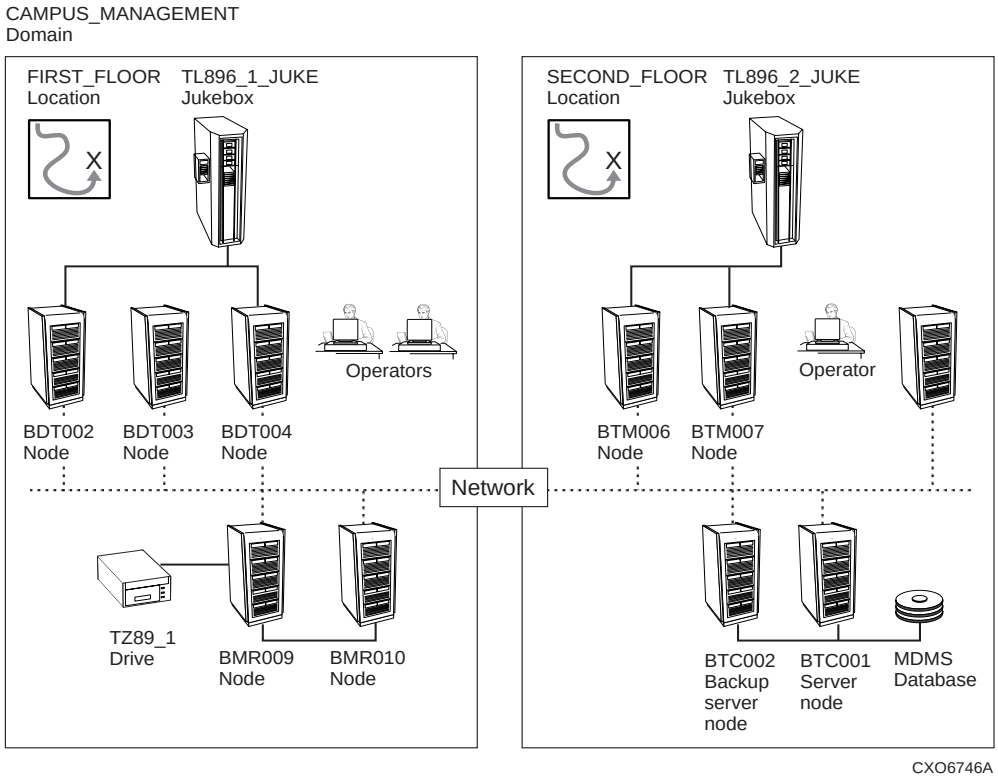
If you have MDMS/SLS V2.X installed, you can convert the symbols and database to MDMS V3. Appendix Kdescribes what has changed, how to do the conversion and how to use MDMS V2.9 clients with an MDMS V3 database server (for a rolling upgrade).

17.1 The MDMS Management Domain

To manage drives and volumes, you must first configure the scope of the MDMS management domain. This includes placing the database in the best location to assure availability, installing and configuring the MDMS process on nodes that serve ABS 3 or HSM 3, and defining node and domain object record attributes. The MDMS Domain is defined by:

- the MDMS database
- start up files on the nodes which access it
- node and domain object records

Figure 17–1 The MDMS Domain



Understanding MDMS configuration concepts is necessary to configure a reliable and available service

17.1.1 The MDMS Database

The MDMS database is a collection of OpenVMS RMS files that store the records describing the objects you manage. lists the files that make up the MDMS database.

Table 17–1 MDMS Database Files and Their Contents

Database File	Object Records
MDMS$DOMAIN_DB.DAT	The only Domain object record
MDMS$DRIVE_DB.DAT	All Drive object records
MDMS$GROUP_DB.DAT	All Group object records
MDMS$JUKEBOX_DB.DAT	All Jukebox object records
MDMS$LOCATION_DB.DAT	All Location object records
MDMS$MAGAZINE_DB.DAT	All Magazine object records
MDMS$MEDIA_DB.DAT	All Media Type object records
MDMS$NODE_DB.DAT	All Node object records
MDMS$POOL_DB.DAT	All Pool object records

Table 17–1 MDMS Database Files and Their Contents

MDMS$VOLUME_DB.DAT	All Volume object records

17.1.1.1 Database Performance

If you are familiar with the structure of OpenVMS RMS files, you can tune and maintain them over the life of the database. You can find File Definition Language (FDL) files in the MDMS$ROOT:[SYSTEM] directory for each of the database files. Refer to the OpenVMS Record Management System documentation for more information on tuning RMS files and using the supplied FDL files.

17.1.1.2 Database Safety

MDMS keeps track of all objects by recording their current state in the database. In the event of a catastrophic system failure, you would start recovery operations by rebuilding the system, and then by restoring the important data files in your enterprise. Before restoring those data files, you would have to first restore the MDMS database files.

Another scenario would be the failure of the storage system on which the MDMS files reside. In the event of a complete disk or system failure, you would have to restore the contents of the disk device containing the MDMS database.

The procedures in this section describe ways to create backup copies of the MDMS database. These procedures use MDMS$SYSTEM:MDMS$COPY_DB_FILES.COM command procedure. This command procedure copies database files with the CONVERT/SHARE command. The procedure in describes how to copy MDMS database files only. The procedure in describes how to process the MDMS database files when they are copied as part of an image backup on the disk device.

To Make Backup Copies of the MDMS Database

The procedure outlined in describes how you can make backup copies of just the MDMS database files using the OpenVMS Backup Utility. This procedure does not account for other files on the device.

Table 17–2 How to Back Up the MDMS Database Files

Step...	Action...
1.	Prepare for making back up copies by finding a disk with enough available space to temporarily hold a copy of each file in the MDMS database.
2.	Determine a time of relative inactivity by MDMS clients, ABS or HSM. For ABS, this could be a few hours after the completion of system backups. For HSM, this is more difficult to determine because a shelving policy could be activated at any time. If necessary, shut down ABS and/or HSM to make sure there are no requests of MDMS.

Table 17–2 How to Back Up the MDMS Database Files

3.	**Note**: If you cannot shut down HSM or ABS, when running MDMS$COPY_DB_FILES.COM, it is possible an update to the data base file can occur after it has been opened. This can create a possibility that the copy of the database file will be out of synchronization with other database files. At the determined time, copy the MDMS database files with the supplied command procedure MDMS$COPY_DB_FILES.COM. `$ @MDMS$ROOT:[TOOLS]MDMS$COPY_DB_FILES` After the MDMS$COPY_DB_FILES command procedure ends, copies of the database files reside on the same disk as the original files.
4.	Use the OpenVMS Backup Utility to create a back up copy of the database files. You must have at least one tape device configured to be shared with applications other than MDMS. The following shows an example BACKUP command: $BACKUP MDMS$DATABASE_LOCATION:*.DAT_COPY *tape_device_name*
5.	After the OpenVMS Backup Utility operation completes, delete the file copies from the database directory.
6.	Store the copies of the MDMS database in a safe location.

To Process the MDMS Database for an Image Backup of the Device

The procedure in shows how to process the MDMS database files for an image backup. The image backup could be part of a periodic full backup and subsequent incremental. This procedure also describes how to use the files in case you restore them.

Table 17–3 Processing MDMS Database Files for an Image Backup

Step...	Action...

1.	Create a preprocessing command procedure to execute before the image backup on the disk. The command procedure must first purge old database file copies from the directory, then creates a new set of copies.
	
	`$PURGE MDMS$DATABASE_LOCATION:*.DAT_COPY`
	`$@MDMS$SYSTEM:MDMS$COPY_DB_FILES`

2.	Plan the backup operation on the disk containing the MDMS database files, to make sure that the preprocessing command procedure executes before the actual backup procedure.
3.	Run the backup operation. Each time you create a backup copy of the disk, you will get a consistent copy of the MDMS database files.

4.	When you need to restore the data to the device, you need to use the consistent files. Rename the .DAT_COPY files to become the .DAT files, then purge the .DAT files from the directory.
	
	`$RENAME MDMS$DATABASE:*.DAT_COPY MDMS$DATABASE:*.DAT`
	`$PURGE MDMS$DATABASE`

17.1.1.3 Moving the MDMS Database

In the event the disk device on which you keep the MDMS database runs out of space, you have the option of moving the MDMS database, or moving other files off the device. The procedure described in this section explains the actions you would have to perform to move the MDMS database. Use this procedure first as a gauge to decide whether moving the MDMS database would be easier or more difficult than moving the other files. Secondarily, use this procedure to

relocate the MDMS database to another disk device. describes how to move the MDMS database to a new device location.

Table 17–4 How to Move the MDMS Database

Step...	Action...
1.	Shut down any applications using MDMS: ABS or HSM. Refer to the respective application documentation for specific commands.
2.	Shut down the MDMS process on all nodes in the domain.
3.	Using the OpenVMS Backup Utility, create a copy of the database files. Use the CRC and VERIFY options to help ensure your copy is valid.
4.	Using the OpenVMS Backup Utility, restore the copy of the database files into the new location. Use CRC and VERIFY options to ensure the restored copy is valid.
5.	In every MDMS start up file SYS$MANAGER:MDMS$SYSTARTUP.COM, define the MDMS$DATABASE_FILES logical to point to the new location.
6.	Start up MDMS on a node enabled as a database server.
7.	From the node, make sure you can access the database by entering an MDMS SHOW command to examine a record from each database file. If you get an error, first check to make sure that the logical assignment for the MDMS$DATABASE_FILES is correct. If the logical assignment is correct, then you will have to determine why the files are not accessible.
8.	Start up the remaining MDMS nodes.
9.	Keep the previous database files on-line, until you know the new database files are accessible.
10.	After you are certain the new database files are accessible, delete the original files.

17.1.2 The MDMS Process

This section describes the MDMS software process, including server availability, interprocess communication, and start up and shut down operations.

17.1.2.1 Server Availability

Each node in an MDMS domain has one MDMS server process running. Within an MDMS domain only one server will be serving the database to other MDMS servers. This node is designated as the *MDMS Database Server*, while the others become *MDMS clients*. Of the servers listed as database servers, the first one to start up tries to open the database. If that node can successfully open the database, it is established as the database server. Other MDMS servers will then forward user requests to the node that has just become the database server.

Subsequently, if the database server fails because of a hardware failure or a software induced shut down, the clients compete among themselves to become the database server. Whichever client is the first to successfully open the database, becomes the new database server. The other clients will then forward user requests to the new database server. User requests issued on the node which is the database server, will be processed on that node immediately.

17.1.2.2 The MDMS Account

During installation you create the MDMS user account as shown in . This account is used by MDMS for every operation it performs.

Example 17–1 MDMS User Account

```
Username:          MDMS$SERVER             Owner:  SYSTEM MANAGER
Account:           SYSTEM                  UIC:    [1,4] ([SYSTEM])
CLI:               DCL                     Tables:
Default:SYS$SYSROOT:[SYSMGR]
LGICMD:SYS$LOGIN:LOGIN
Flags: DisForce_Pwd_Change DisPwdHis
Primary days:   Mon Tue Wed Thu Fri Sat Sun
Secondary days:
No access restrictions
Expiration:       (none)     Pwdminimum: 14    Login Fails:        0
Pwdlifetime:    30 00:00     Pwdchange: 1-JUL-1998 12:19
Maxjobs:               0     Fillm:     500    Bytlm:         100000
Maxacctjobs:           0     Shrfillm:   0     Pbytlm:             0
Maxdetach:             0     BIOlm:   10000    JTquota:         4096
Prclm:                10     DIOlm:     300    WSdef:           5000
Prio:                  4     ASTlm:     300    WSquo:          10000
Queprio:               0     TQElm:     300    WSextent:       30000
CPU:              (none)     Enqlm:    2500    Pgflquo:       300000
Authorized Privileges:
DIAGNOSE  NETMBX PHY_IO READALL SHARE SYSNAM SYSPRV TMPMBX WORLD
Default Privileges:
DIAGNOSE  NETMBX PHY_IO READALL SHARE SYSNAM SYSPRV TMPMBX WORLD
```

17.1.3 The MDMS Start Up File

MDMS creates the SYS$STARTUP:MDMS$SYSTARTUP.COM command procedure on the initial installation. This file includes logical assignments that MDMS uses when the node starts up. The installation process also offers an opportunity to make initial assignments to the logicals.

If you install MDMS once for shared access in an OpenVMS Cluster environment, this file is shared by all members. If you install MDMS on individual nodes within an OpenVMS Cluster environment, this file is installed on each node.

In addition to creating node object records and setting domain and node attributes, you must define logicals in the MDMS start up file. These are all critical tasks to configure the MDMS domain.

provides brief descriptions of most of the logical assignments in MDMS$SYSTARTUP.COM. More detailed descriptions follow as indicated.

Table 17–5 MDMS$SYSTARTUP.COM Logical Assignments

Logical Name	Assignment
MDMS$DATABASE_SERVERS	List of all nodes that can run as the MDMS database server. See Section for more information.
MDMS$ROOT	Device and directory of MDMS files.
MDMS$LOGFILE_LOCATION	Device and directory of the MDMS log file. See Section for more information.
MDMS$DATABASE_LOCATION	Device and directory of the MDMS database files. All installations in any one domain must define this as a common location. Section identifies the MDMS database files and describes how they should be managed.

Table 17–5 MDMS$SYSTARTUP.COM Logical Assignments

MDMS$TCPIP_SENDPORTS	Range of ports for the node to use for out going connections. The default range is for privileged ports; 1 through 1023.
MDMS$SUPPORT_PRE_V3	Support for SLS/MDMS Version 2.9*x* clients. The default value is FALSE. If you need to support some systems running SLS/MDMS Version 2.9*x*, then set this value to TRUE.

17.1.3.1 MDMS$DATABASE_SERVERS - Identifies Domain Database Servers

Of all the nodes in the MDMS domain, you select those which can act as a database server. Only one node at a time can be the database server. Other nodes operating at the same time communicate with the node acting as the database server. In the event the server node fails, another node operating in the domain can become the database server if it is listed in the MDMS$DATABASE_SERVERS logical.

For instance, in an OpenVMS Cluster environment, you can identify all nodes as a potential server node. If the domain includes an OpenVMS Cluster environment and some number of nodes remote from it, you could identify a remote node as a database server if the MDMS database is on a disk served by the Distributed File System software (DECdfs). However, if you do not want remote nodes to function as a database server, do not enter their names in the list for this assignment.

The names you use must be the full network name specification for the transports used. shows example node names for each of the possible transport options. If a node uses both DECnet and TCP/IP, full network names for both should be defined in the node object

Note

When you specify the use of both DECnet and TCP/IP network transports in the configuration, you should include node names for each transport as appropriate. Specifying only one node name for a specific transport is allowable. However, when that node attempts to locate a database server on start up, only the transport for which the name applies will be used, thereby limiting reliability.

Table 17–6 Network Node Names for MDMS$DATABASE_NODES

Network Transport	Node Name Examples
DECnet	NODE_A,NODE_B
DECnet Plus	SITE:.NODE_A.SITE,SITE:.NODE_B.SITE
TCP/IP	node_a.site.inc.com,node_b.site.inc.com

17.1.3.2 MDMS$LOGFILE_LOCATION

Defines the location of the Log Files. For each server running, MDMS uses a log file in this location. The log file name includes the name of the ***cluster*** node it logs.
For example, the log file name for a node with a ***cluster*** node name NODE_A would be:

```
MDMS$LOGFILE_LOCATION:MDMS$LOGFILE_NODE_A.LOG
```

17.1.3.3 MDMS Shut Down and Start Up

How to Shut Down MDMS

To shut down MDMS on the current node enter this command:

```
$@SYS$STARTUP:MDMS$SHUTDOWN.COM
```

How to Restart MDMS

To restart MDMS (shut down and immediate restart), enter the shut down command and the parameter RESTART:

```
$@SYS$STARTUP:MDMS$SHUTDOWN RESTART
```

How to Start Up MDMS

To start up MDMS on the current node enter this command:

```
$@SYS$STARTUP:MDMS$STARTUP.COM
```

17.1.4 Managing an MDMS Node

The MDMS node object record characterizes the function of a node in the MDMS domain and describes how the node communicates with other nodes in the domain.

17.1.4.1 Defining a Node's Network Connection

To participate in an MDMS domain, a node object has to be entered into the MDMS database. This node object has 4 attributes to describe its connections in a network:

1. If the node is part of a DECnet (Phase IV) network, then the name of the node object must match exactly with the node's DECnet node name (i.e. SYS$NODE). Otherwise the name of the node object may be any character string up to 31 characters.
2. If the node is part of a DECnet-Plus (Phase V) network, the DECnet-Plus full name must be supplied as an attribute to the node object, using the /DECNET_PLUS_FULLNAME Qualifier or GUI equivalent.
3. If the node is part of an Internet or Intranet using TCP/IP, the TCP/IP full name must be supplied as an attribute to the node object, using the /TCPIP_FULLNAME Qualifier or GUI equivalent.
4. Depending on which network or networks are available or should be used, the node's transport attribute has to be set to either DECNET, TCPIP or both.

When an MDMS server starts up it only has its network node name/s to identify itself in the MDMS database. Therefore if a node has a network node name but it is not defined in the *node object records* of the database, this node will be rejected as not being fully enabled. For example, a node has a TCP/IP name and TCP/IP is running but the node object record shows the TCP/IP full name as blank.

There is one situation where an MDMS server is allowed to function even if it does not have a node object record defined or the node object record does not list all network names. This is in the case of the node being an MDMS database server. Without this exception, no node entries can be created in the database. As long as a database server is not fully enabled in the database it will not start any network listeners.

17.1.4.2 Defining How the Node Functions in the Domain

This section describes how to designate an MDMS node as a database server, enable and disable the node.

Designating Potential Database Servers

When you install MDMS, you must decide which nodes will participate as potential database servers. To be a database server, the node must be able to access the database disk device. Typically, in an OpenVMS Cluster environment, all nodes would have access to the database disk device, and would therefore be identified as potential database servers.

Set the database server attribute for each node identified as a potential database server. For nodes in the domain that are not going to act as a database server, negate the database server attribute.

Disabling and Enabling MDMS Nodes

There are several reasons for disabling an MDMS node.

- Preventing the node you are disabling from becoming the database server.
- Preventing applications and users on the node from issuing or processing MDMS requests.

Disable the node from the command line or the GUI and restart MDMS.

When you are ready to return the node to service, enable the node.

17.1.4.3 Enabling Interprocess Communication

Nodes in the MDMS domain have two network transport options: one for DECnet, the other for TCP/IP. When you configure a node into the MDMS domain, you can specify either or both these transport options by assigning them to the transport attribute. If you specify both, MDMS will attempt interprocessor communications on the first transport value listed. MDMS will then try the second transport value if communication fails on the first.

If you are using the DECnet Plus network transport, define the full DECnet Plus node name in the decnet fullname attribute. If you are using an earlier version of DECnet, leave the DECnet-Plus fullname attribute blank.

If you are using the TCP/IP network transport, enter the node's full TCP/IP name in the TCPIP fullname attribute. You can also specify the receive ports used by MDMS to listen for incoming requests. By default, MDMS uses the port range of 2501 through 2510. If you want to specify a different port or range of ports, append that specification to the TCPIP fullname. For example:

```
node_a.site.inc.com:2511-2521
```

17.1.4.4 Describing the Node

Describe the function, purpose of the node with the description attribute. Use the location attribute to identify the MDMS location where the node resides.

17.1.4.5 Communicating with Operators

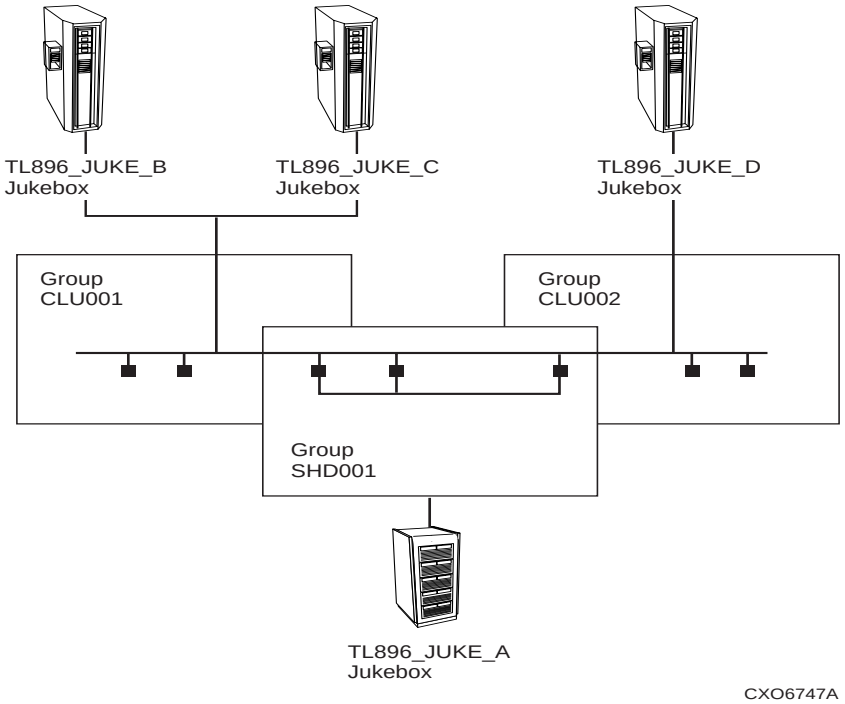
List the OPCOM classes of operators with terminals connected to this node who will receive OPCOM messages. Operators who enable those classes will receive OPCOM messages pertaining to devices connected to the node.

For more information about operator communication, see Section .

17.1.5 Managing Groups of MDMS Nodes

MDMS provides the *group* object record to define a group of nodes that share common drives or jukeboxes. Typically, the group object record represents all nodes in an OpenVMS Cluster environment, when drives in the environment are accessible from all nodes.

Figure 17–2 Groups in the MDMS Domain



Some configurations involve sharing a device between nodes of different OpenVMS Cluster environments. You could create a group that includes all nodes that have access to the device.

When you create a group to identify shared access to a drive or jukebox assign the group name as an attribute of the drive or jukebox. When you set the group attribute of the drive or jukebox object record, MDMS clears the node attribute.

The following command examples create a functionally equivalent drive object records.

```
$!These commands create a drive connected to a Group object
$MDMS CREATE GROUP CLUSTER_A /NODES=(NODE_1,NODE_2,NODE_3)
$MDMS CREATE DRIVE NODE$MUA501/GROUPS=CLUSTER_A
$!
$!This command creates a drive connected to NODE_1, NODE_2, and NODE_3
$MDMS CREATE DRIVE NODE$MUA501/NODES=(NODE_1,NODE_2,NODE_3)
```

Figure 17–2 is a model of organizing clusters of nodes in groups and how devices are shared between groups.

17.1.6 Managing the MDMS Domain

The domain object record describes global attributes for the domain and includes the description attribute where you can enter an open text description of the MDMS domain. Additional domain object attributes define configuration parameters, access rights options, and default volume management parameters. See Figure 17–1.

17.1.6.1 Domain Configuration Parameters

Operator Communications for the Domain

Include all operator classes to which OPCOM messages should go as a comma separated list value of the OPCOM classes attribute. MDMS uses the domain OPCOM classes when nodes do not have their classes defined.

For more information about operator communication, see Section, Managing Operations.

Resetting the Request Identifier Sequence

If you want to change the request identifier for the next request, use the request id attribute.

17.1.6.2 Domain Options for Controlling Rights to Use MDMS

This section briefly describes the attributes of the domain object record that implement rights controls for MDMS users. Refer to Appendix on MDMS Rights and Privileges for the description of the MDMS rights implementation.

ABS Users

If you use MDMS to support ABS, you can set the ABS rights attribute to allow any user with any ABS right to perform certain actions with MDMS. This feature provides a short cut to managing rights by enabling ABS users and managers access to just the features they need. Negating this attribute means users with any ABS rights have no additional MDMS rights.

MDMS Client Applications

MDMS defines default low level rights for the application rights attribute according to what ABS and HSM minimally require to use MDMS.

Caution

The ABS or HSM processes include the MDMS_APPLICATION_RIGHTS identifier which assumes the low level rights associated with it. Do not modify the low level rights for the domain application rights attribute. Changing the values to this attribute can cause your application to fail.

Default Rights for Various System Users

If you want to grant all users certain MDMS rights without having to modify their UAF records, you can assign those low level rights to the default rights attribute. Any user without specific MDMS rights in their UAF file will have the rights assigned to the default rights identifier.

Use the operator rights attribute to identify all low level rights granted to any operator who has been granted the MDMS_OPERATOR right in their UAF.

Use the SYSPRV attribute to allow any process with SYSPRV enabled the rights to perform any and all operations with MDMS.

Use the user rights attribute to identify all low level rights granted to any user who has been granted the MDMS_USER right in their UAF.

17.1.6.3 Domain Default Volume Management Parameters

The MDMS domain includes attributes used as the foundation for volume management. Some of these attributes provide defaults for volume management and movement activities, others define particular behavior for all volume management operations. The values you assign to these attributes will, in part, dictate how your volume service will function. lists brief descriptions of these attributes.

Table 17–7 Default Volume Management Parameters

Attribute	Meaning
Offsite Location	MDMS uses this location for the volume and magazine offsite location unless another location is specified.
Onsite Location	MDMS uses this location for the volume and magazine onsite location unless another location is specified.

Table 17–7 Default Volume Management Parameters

Maximum Scratch Time	This is the maximum amount of time that can be set as the scratch time on any volume in the domain.
Mail Users	A list of e-mail address for users or accounts to be notified when volumes are deallocated. Any email address on this list must be in syntax that the OpenVMS Mail Utility can process.
Deallocate State	Specifies whether a volume is immediately freed upon reaching the deallocation date, or if the volume is put into a transition state for temporary protection before being set free.
Transition Time	The amount of time a volume stays in the transition state.
Scratch Time	MDMS uses the time span specified here to set the default scratch date when MDMS allocates a volume.
Protection	The default protection for volumes allocated to ABS and MDMS. The format is the standard OpenVMS file protection specification format.

17.1.7 MDMS Domain Configuration Issues

This section addresses issues that involve installing additional MDMS nodes into an existing domain, or removing nodes from an operational MDMS domain.

17.1.7.1 Adding a Node to an Existing Configuration

Once you configure the MDMS domain, you might have the opportunity to add a node to the existing configuration. describes the procedure for adding a node to an existing MDMS domain.

Table 17–8 Adding a Node to an Existing Configuration

Step...	Action...
1.	Create a node object record with either the CLI or GUI. Set the transport and network name attributes in accordance with available net work options. For more information, see Section .
2.	Decide if the node will be a database server or will only function as an MDMS client. • If the node is to be a database server, set the database server attribute (default) • If the node is not to be a database server, negate the database server attribute.
3.	Set the remaining node object attributes, then complete the creation of the node.
4.	If the node will not share an existing startup file and database server image, then install the MDMS software with the VMSINSTAL utility.
5.	6. If the new node is a database server, then add the node by its network transport names to the MDMS$DATABASE_SERVERS list in all start up files in the MDMS domain.

17.1.7.2 Removing a node from an existing configuration

When you remove a node from the MDMS domain, there are several additional activities you must perform after deleting the node object record.

- If the node was a database server, remove its node names from all MDMS start up files in the MDMS$DATABASE_SERVERS logical assignment.
- Remove any references to the node that might exist in remaining MDMS object records.

- Remove any references to the node that might exist in DCL command procedures.

17.2 Configuring MDMS Drives, Jukeboxes and Locations

MDMS manages the use of drives for the benefit of its clients, ABS and HSM. You must configure MDMS to recognize the drives and the locations that contain them. You must also configure MDMS to recognize any jukebox that contains managed drives.

You will create drive, location, and possibly jukebox object records in the MDMS database. The attribute values you give them will determine how MDMS manages them. The meanings of some object record attributes are straightforward. This section describes others because they are more important for configuring operations.

17.2.1 Configuring MDMS Drives

Before you begin configuring drives for operations, you need to determine the following aspects of drive management:

- How to describe the drive
- Which systems need access to the drive
- How the drive fits into your operations

17.2.1.1 How to Describe an MDMS Drive

You must give each drive a name that is unique within the MDMS domain. The drive object record can be named with the OpenVMS device name, if desired, just as long as the name is not duplicated elsewhere.

Use the description attribute to store a free text description of anything useful to your management of the drive. MDMS stores this information, but takes no action with it.

The device attribute must contain the OpenVMS allocation class and device name for the drive. If the drive is accessed from nodes other than the one from which the command was entered, you must specify nodes or groups in the /NODE or /GROUP attributes in the drive record. Do not specify nodes or groups in the drive name or the device attribute.

If the drive resides in a jukebox, you must specify the name of the jukebox with the jukebox attribute. Identify the position of the drive in the jukebox by setting the drive number attribute. Drives start at position 0.

Additionally, the jukebox that contains the drives must also be managed by MDMS.

17.2.1.2 How to Control Access to an MDMS Drive

MDMS allows you to dedicate a drive solely to MDMS operations, or share the drive with other users and applications. Specify your preference with the shared attribute.

You need to decide which systems in your data center are going to access the drives you manage.

Use the groups attribute if you created group object records to represent nodes in an OpenVMS Cluster environment or nodes that share a common device.

Use the nodes attribute if you have no reason to refer to any collection of nodes as a single entity, and you plan to manage nodes, and the objects that refer to them, individually.

The last decision is whether the drive serves locally connected systems, or remote systems using the RDF software. The access attribute allows you to specify local, remote (RDF) or both.

17.2.1.3 How to Configure an MDMS Drive for Operations

Specify the kinds of volumes that can be used in the drive by listing the associated media type name in the media types attribute. You can force the drive to not write volumes of particular media types. Identify those media types in the read only attribute.

If the drive has a mechanism for holding multiple volumes, and can feed the volumes sequentially to the drive, but does not allow for random access or you choose not to use the random access feature, then you can designate the drive as a stacker by setting the stacker attribute.

Set the disabled attribute when you have to exclude the managed drive from operations by MDMS. If the drive is the only one of its kind (for example if it accepts volumes of a particular media type that no other drives accept), make sure you have another drive that can take load requests. Return the drive to operation by setting the enabled attribute.

17.2.1.4 Determining Drive State

Caution

Changing the value of the state attribute could cause MDMS or the applications using it to fail.

The drive object record state attribute shows the state of managed MDMS drives. MDMS sets one of four values for this attribute: Empty, Full, Loading, or Unloading.

17.2.1.5 Adding and Removing Managed Drives

The procedure described in describes how to add a drive to the MDMS domain.

The procedure described in describes how to remove a drive from the MDMS domain.

17.2.2 Configuring MDMS Jukeboxes

MDMS manages Media Robot Driver (MRD) controlled jukeboxes and DCSC controlled jukeboxes. MRD is a software that controls SCSI-2 compliant medium changers. DCSC is software that controls large jukeboxes manufactured by StorageTek, Inc. This section first describes the MDMS attributes used for describing all jukeboxes by function. Subsequent descriptions explain attributes that characterize MRD jukeboxes and DCSC jukeboxes respectively.

17.2.2.1 How to Describe an MDMS Jukebox

Assign unique names to jukeboxes you manage in the MDMS domain. When you create the jukebox object record, supply a name that describes the jukebox.

Set the control attribute to MRD if the jukebox operates under MRD control. Otherwise, set the control to DCSC.

Use the description attribute to store a free text description of the drive. You can describe its role in the data center operation or other useful information. MDMS stores this information for you, but takes no actions with it.

17.2.2.2 How to Control Access to an MDMS Jukebox

You can dedicate a jukebox solely to MDMS operations, or you can allow other applications and users access to the jukebox device. Specify your preference with the shared attribute.

You need to decide which systems in the data center are going to access the jukebox.

Use the groups attribute if you created group object records to represent nodes in an OpenVMS Cluster environment or nodes that share a common device.

Use the nodes attribute if you have no reason to refer to any collection of nodes as a single entity, and you plan to manage nodes, and the objects that refer to them, individually.

17.2.2.3 How to Configure an MDMS Jukebox for Operations.

Disable the jukebox to exclude it from operations. Make sure that applications using MDMS will either use other managed jukeboxes, or make no request of a jukebox you disable. Enable the jukebox after you complete any configuration changes. Drives within a disabled jukebox cannot be allocated.

17.2.2.4 Attribute for DCSC Jukeboxes

Set the library attribute to the library identifier of the particular silo the jukebox objects represents. MDMS supplies 1 as the default value. You will have to set this value according the number silos in the configuration and the sequence in which they are configured.

17.2.2.5 Attributes for MRD Jukeboxes

Specify the number of slots for the jukebox. Alternatively, if the jukebox supports magazines, specify the topology for the jukebox (see Section 17.2.2.7 Magazines and Jukebox Topology).

The robot attribute must contain the OpenVMS device name of the jukebox medium changer (also known as the robotic device).

If the jukebox is accessed from nodes other than the one from which the command was entered, you must specify nodes or groups in the /NODE or /GROUP attributes in the jukebox record. Do not specify nodes or groups in the jukebox name or the robot attribute.

17.2.2.6 Determining Jukebox State

Caution

Changing the value of the state attribute could cause MDMS or the applications using it to fail.

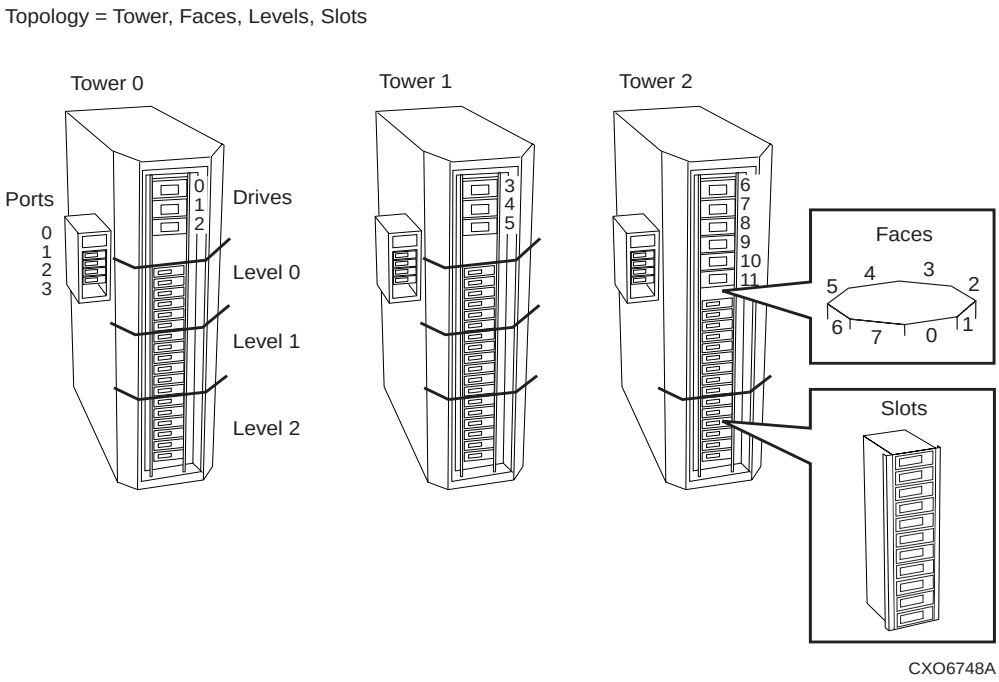
The jukebox object record state attribute shows the state of managed MDMS jukeboxes. MDMS sets one of three values for this attribute: Available, In use, and Unavailable.

17.2.2.7 Magazines and Jukebox Topology

If you decide that your operations benefit from the management of magazines (groups of volumes moved through your operation with a single name) must set the jukebox object record to enable it. Set the usage attribute to magazine and define the jukebox topology with the topology attribute. See Figure 17–4 for a sample overview of how the 11 and 7 slot bin packs can be used as a magazine.

Setting the usage attribute to nomagazine means that you will move volumes into and out of the jukebox independently (using separate commands for each volume, regardless if they are placed into a physical magazine or not).

Figure 17–3 Jukebox Topology



The following paragraphs explain jukebox topology.

Towers, Faces, Levels, and Slots

Some jukeboxes have their slot range subdivided into towers, faces, and levels. See for an overview of how the configuration of Towers, Faces, Levels and Slots constitute Topology. Note that the topology in comprises 3 towers. In the list of topology characteristics, you should identify every tower in the configuration. For each tower in the configuration, you must inturn identify:

- the tower by number (starting at zero)
- the number of faces in the tower (starting at one)
- the number of levels per face (starting at one)
- the number of slots per magazine (starting at one)

Restrictions for Using Magazines

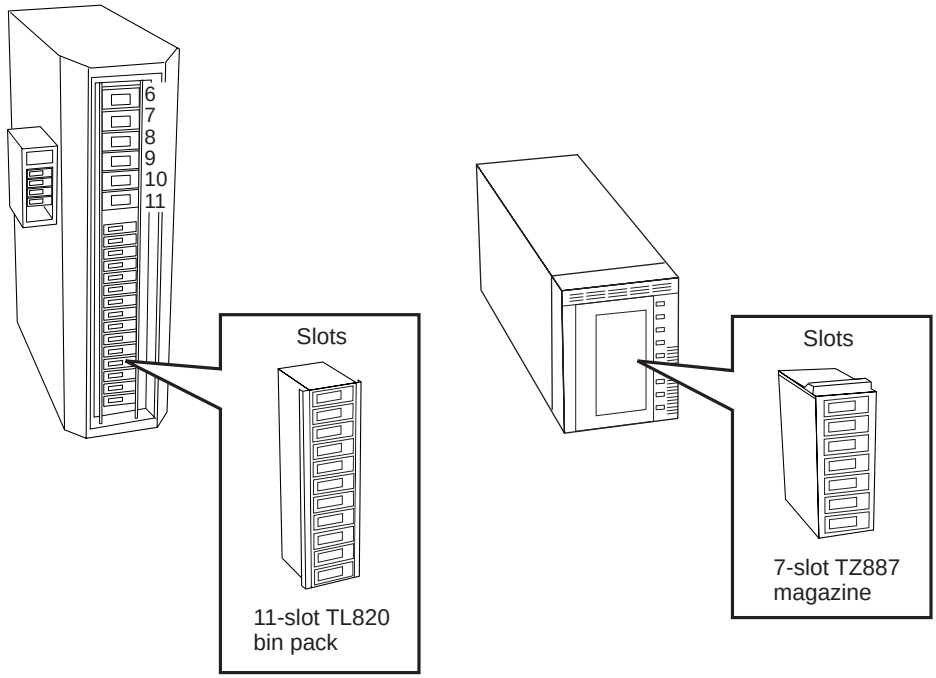
You must manually open the jukebox when moving magazines into and out of the jukebox. Once in the jukebox, volumes can only be loaded and unloaded relative to the slot in the magazine it occupies.

TL896 Example

While using multiple TL896 jukebox towers you can treat the 11 slot bin packs as magazines. The following command configures the topology of the TL896 jukebox as shown in for use with magazines:

```
$ MDMS CREATE JUKEBOX JUKE_1/ -
$_ /TOPOLOGY=(TOWERS=(0,1,2), FACES=(8,8,8), -
$_ LEVELS=(3,3,2), SLOTS=(11,11,11))
```

Figure 17–4 Magazines



17.2.3 Summary of Drive and Jukebox Issues

This section describes some of the management issues that involve both drives and jukeboxes.

17.2.3.1 10.2.3.1Enabling MDMS to Automatically Respond to Drive and Jukebox Requests

Drive and jukebox object records both use the automatic load reply attribute to provide an additional level of automation.

When you set the automatic reply attribute to the affirmative, MDMS will poll the drive or jukebox for successful completion of an operator-assisted operation for those operations where polling is possible. For example, MDMS can poll a drive, determine that a volume is in the drive, and cancel the associated OPCOM request to acknowledge a load. Under these circumstances, an operator need not reply to the OPCOM message after completing the load. To use this feature, set the automatic reply attribute to the affirmative. When this attribute is set to the negative, which is the default, an operator must acknowledge each OPCOM request for the drive or jukebox before the request is completed.

17.2.3.2 Creating a Remote Drive and Jukebox Connection

If you need to make backup copies to a drive in a remote location, using the network, then you must install the Remote Device Facility software (RDF). The RDF software must then be configured to work with MDMS.

See for a description of the actions you need to take to configure RDF software.

Table 17–9 Actions for Configuring Remote Drives

Stage	Action
1.	Install the appropriate RDF component on the node. • Install the RDF Server software on all nodes that are connected to the tape drives used for remote operations. • Install the RDF Client software on all nodes that initiate remote operations to those tape drives.
2.	For each tape drive served with RDF Server software, make sure there is a drive object record in the MDMS that describes it. Take note of each node connected to the drive, even if the drive object record includes a group definition instead of a node.
3.	On each node connected to the tape drive, edit the file TTI_RDEV:CONFIG_*node*.DAT so that all tape drives are represented. The syntax for representing tape drives is given in the file.

17.2.3.3 How to Add a Drive to a Managed Jukebox

When you add another drive to a managed jukebox, just specify the name of the jukebox in which the drive resides, in the drive object record.

17.2.3.4 Temporarily Taking a Managed Device From Service

You can temporarily remove a drive or jukebox from service. MDMS allows you to disable and enable drive and jukebox devices. This feature supports maintenance or other operations where you want to maintain MDMS support for ABS or HSM, and temporarily remove a drive or jukebox from service.

Note

If you remove a jukebox from service, you cannot access any of its volumes. Make sure you empty the jukebox, or make sure your operations will continue, without the use of the volumes in any jukebox you disable.

17.2.3.5 Changing the Names of Managed Devices

During the course of management, you might encounter a requirement to change the device names of drives or jukeboxes under MDMS management, to avoid confusion in naming. When you have to change the device names, follow the procedure in Table 17–10.

Table 17–10 Changing the Names of Managed Devices

Step...	Action...
1.	Either find a time when ABS or HSM is not using the drive or jukebox device or disable the device with MDMS.
2.	Change the device names at the operating system. Verify the devices respond using operating system commands or MRU commands for a jukebox device.
3.	Change the MDMS drive device name, and/or the jukebox robot name as needed to reflect the new system device names.

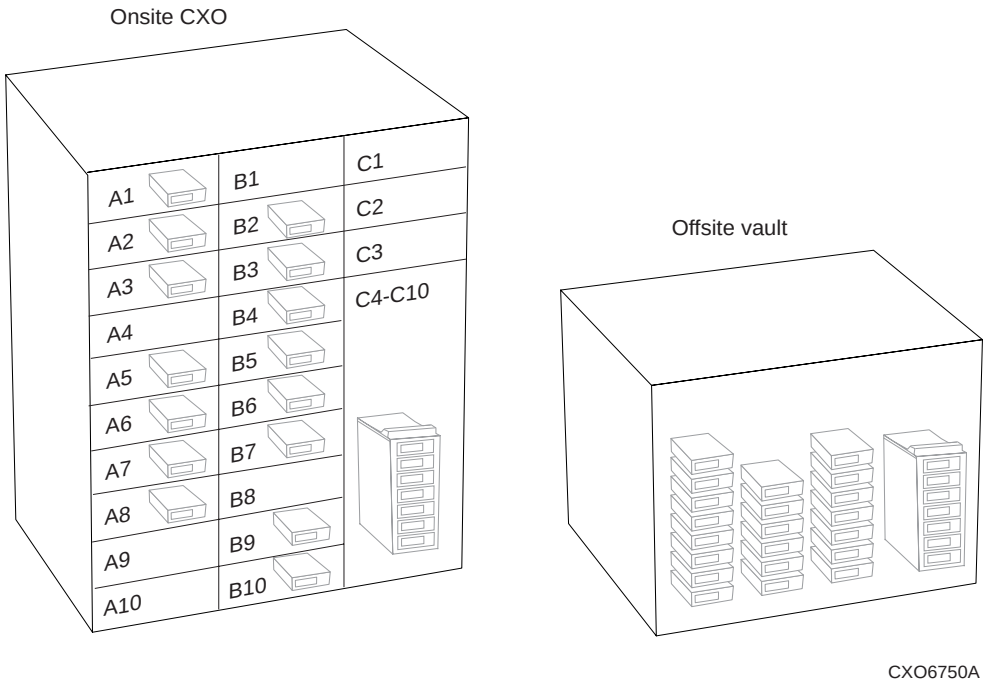
Table 17–10 Changing the Names of Managed Devices

4.	If your drive and/or jukebox object records are named according to the operating system device name, then you should create new object records. If you want to create new object records, use the inherit feature and specify the previous object record. For GUI operation.
5.	If you created new object records, then delete the old object records, and check and modify any references to the old object records. For more information.
6.	Enable the new drive and/or jukebox with MDMS.

17.2.4 Locations for Volume Storage

MDMS allows you to identify locations in which you store volumes. Create a location object record for each place the operations staff uses to store volumes. These locations are referenced during move operations, load to, or unload from stand-alone drives.

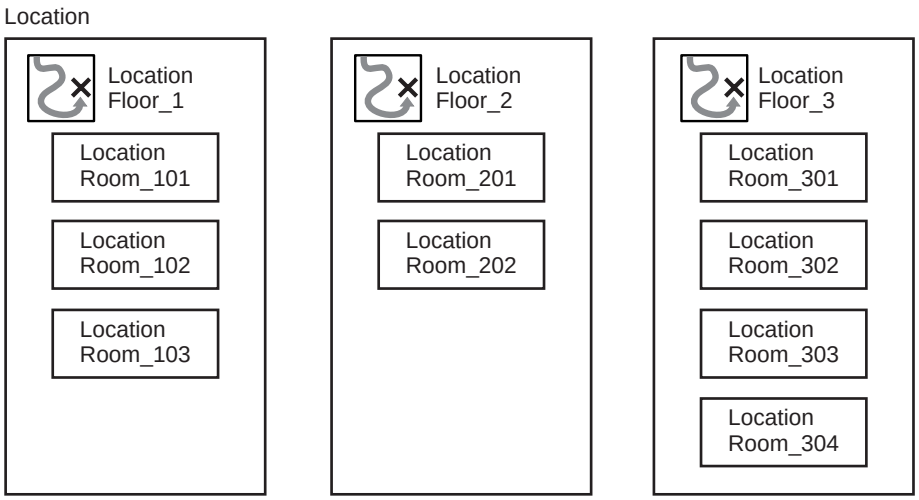
Figure 17–5 Volume Locations



If you need to divide your location space into smaller, named locations, define locations hierarchically. The location attribute of the location object record allows you to name a higher level location. For example, you can create location object records to describe separate rooms in a data center by first creating a location object record for the data center. After that, create object records for each room, specifying the data center name as the value of the location attribute for the room locations.

When allocating volumes or drives by location, the volumes and drives do not have to be in the exact location specified; rather they should be in a compatible location. A location is considered compatible with another if both have a common root higher in the location hierarchy. For example, in Figure 17–6, locations Room_304 and Floor_2 are considered compatible, as they both have location Building_1 as a common root.

Figure 17–6 Named Locations



Your operations staff must be informed about the names of these locations as they will appear in OPCOM messages. Use the description attribute of the location object record to describe the location it represents as accurately as possible. Your operations staff can refer to the information in the event they become confused about a location mentioned in an OPCOM message.

You can divide a location into separate spaces to identify locations of specific volumes. Use the spaces attribute to specify the range of spaces in which volumes can be stored. If you do not need that level of detail in the placement of volumes at the location, negate the attribute.

17.3 Sample MDMS Configurations

MDMS provides a configuration procedure to guide you through the configuration process. Please refer to Appendix "Sample Configuration of MDMS" for how to use this procedure.

18

Basic MDMS Operations

This chapter describes basic MDMS operations and functions that apply to many MDMS actions.

18.1 MDMS User Interfaces

MDMS includes two interfaces: a command line interface (CLI) and a graphic user interface (GUI). This section describes how these interfaces allow you to interact with MDMS.

18.1.1 Command Line Interface

The CLI is based on the MDMS command. The CLI includes several features that offer flexibility and control in the way in which you use it. This interface provides for interactive operations and allows you to run DCL command procedures for customized operations.

Understanding these features help you become a more effective command line interface user and DCL programmer.

18.1.1.1 Command Structure

The command structure includes the MDMS keyword, the operational verb and an object class name at a minimum. Optionally the command can include a specific object name and command qualifiers.

The following example shows the MDMS command structure for most commands:

```
$MDMS verb object_class [object_name] [/qualifier [,...]]
```

The Move and Report commands support multiple parameters, as documented in the Archive/Backup System for OpenVMS (ABS) or Hierarchical Storage Management for OpenVMS (HSM) Command Reference Guide.

18.1.1.2 Process Symbols and Logical Names for DCL Programming

Some MDMS commands include features for capturing text that can be used to support DCL programming.

The MDMS SHOW VOLUME command includes a /SYMBOLS qualifier to define a set of symbols that store the specified volume's attributes.

Several MDMS commands can involve operator interaction if necessary. These commands includes a /REPLY qualifier to capturing the operator's reply to the OPCOM message created to satisfy the request.

The allocate commands can return an allocated object name. You can assign a process logical name to pick up this object name by using the /DEFINE=logical name qualifier.

18.1.1.3 Creating, Changing, and Deleting Object Records With the CLI

The interactions between the MDMS process and object records in the database form the basis of MDMS operations. Most command line interface actions involve the object record verbs MDMS

CREATE, MDMS SET, MDMS SHOW, and MDMS DELETE. Use the MDMS CREATE verb to create object records that represent the objects you manage. Use MDMS SHOW and MDMS SET to view and change object attributes. The MDMS DELETE command removes object records from the MDMS database.

You do not create all object records with the MDMS CREATE command or with the GUI creation options. MDMS creates some records automatically. During installation MDMS creates the Domain object record, and volume object records can be created in response to an inventory operation.

18.1.1.4 Add and Remove Attribute List Values With the CLI

This section describes the how to add, remove, and change attribute list values.

Command Features

The MDMS CREATE and MDMS SET commands for every object class that has one or more attributes with list values include /ADD and /REMOVE qualifiers. These qualifiers allow you to manipulate the attribute lists.

Use the /ADD qualifier to add a new value to the attribute value list with both the MDMS CREATE/INHERIT and MDMS SET commands.

Use the /REMOVE qualifier to remove an identified value from the attribute value list with both the MDMS CREATE/INHERIT and MDMS SET commands.

To change an entire attribute value list, specify a list of new values with the attribute qualifier.

Command Examples

The following example shows how these qualifiers work.

This command creates a new drive object record, taking attribute values from an existing drive object record. In it, the user adds a new media type name to the /MEDIA_TYPE value list.

```
$MDMS CREATE DRIVE TL8_4 /INHERIT=TL89X_1 /MEDIA_TYPE=(TK9N) /ADD
```

After being created, the data center management plan requires the jukebox containing drive TL8_4 to service requests from a different group of nodes. To change the group list values, but nothing else, the user issues the following SET command.

```
$MDMS SET DRIVE TL8_4 /GROUPS=(FINGRP,DOCGRP)
```

Later, the nodes belonging to DOCGRP no longer need drive TL8_4. The following command removes DOCGRP from the /GROUPS attribute list.

```
$MDMS SET DRIVE TL8_4 /GROUPS=DOCGRP /REMOVE
```

18.1.1.5 Operational CLI Commands

The MDMS command line interface includes commands for operations in the MDMS domain. These commands initiate actions with managed objects. Qualifiers to these commands tailor the command actions to suit your needs. The following examples show how these qualifiers work:

Table 18–1 Operational CLI Commands

Command	Operation
MDMS ALLOCATE DRIVE	Allocate a drive for exclusive use of an MDMS client process, such as ABS or HSM.
MDMS ALLOCATE VOLUME	Allocate a volume for exclusive use of an MDMS user.

Table 18–1 Operational CLI Commands

Command	Operation
MDMS BIND VOLUME	Bind a volume to a volume set.
MDMS CANCEL REQUEST	Cancel an outstanding request before it is completed by the MDMS system
MDMS DEALLOCATE DRIVE	Free a drive that has been allocated for the exclusive use of an MDMS client process.
MDMS DEALLOCATE VOLUME	Deallocate a volume into either the Free or Transition states, making it available for use by other users (optionally after a transition interval).
MDMS INITIALIZE VOLUME	Initialize a volume, making it ready for writing and reading.
MDMS INVENTORY JUKEBOX	Compare the contents of a jukebox with the MDMS database and take corrective action as specified.
MDMS LOAD DRIVE	Load a known drive with any compatible volume.
MDMS LOAD VOLUME	Load a known volume into any compatible drive.
MDMS MOVE MAGAZINE	Move a magazine from one location or jukebox to another location or jukebox.
MDMS MOVE VOLUME	Move a volume from any location, jukebox, or magazine, to another location, jukebox, or magazine.
MDMS REPORT VOLUME	Generate a report of volumes sharing specified attributes.
MDMS UNBIND VOLUME	Remove a volume from a volume set.
MDMS UNLOAD DRIVE	Unload any volume from the specified drive.
MDMS UNLOAD VOLUME	Unload the specified volume from a drive.

18.1.1.6 Asynchronous Requests

Many MDMS commands include the /NOWAIT qualifier. These commands start actions that require some time to complete. Commands entered with /NOWAIT are internally queued by MDMS as an asynchronous request. The request remains in the queue until the action succeeds or fails.

To show currently outstanding requests, use the MDMS SHOW REQUESTS command. To cancel a request, use the MDMS CANCEL REQUEST command.

18.1.2 Graphic User Interface

MDMS includes a GUI based on Java technology. Through the GUI, you can manage MDMS from any Java enabled system on your network that is connected to an OpenVMS system running MDMS.

18.1.2.1 Object Oriented Tasks

Most MDMS operations involve single actions on one or more objects. The basic concept of the GUI supports this management perspective. The interface allows you to select one or more objects and enables management actions through point-and-click operations.

Viewing Object Records with the GUI

To view object records with the GUI, select the class from the icon bar at the top of the screen. Use the next screen to select the particular records you want to view, then press the Modify or Delete option. The GUI then displays the object record.

Operational Actions With the GUI

In addition to creating, modifying, and deleting object records, the GUI enables management actions. Table 18–2 shows the objects and the actions associated with them.

Table 18–2 Operational Actions With the GUI

Object	Action	Operation
Drive	Load	Load a known drive with any compatible volume.
	Unload	Unload any volume from the specified drive.
Jukebox	Inventory	Compare the contents of jukebox with the MDMS database; take corrective action as specified.
Volume	Allocate	Allocate a volume for the exclusive use of an MDMS user.
	Bind	Bind a volume to a volume set.
	Deallocate	Deallocate a volume into either FREE or TRANSITION states, making it available for use by other users (optionally after a transition interval).
	Initialize	Initialize a volume, making it ready for writing and reading.
	Load	Load a known volume into any compatible drive.
	Move	Move a volume from any location, jukebox, or magazine, to another location, jukebox, or magazine.
	Report	Generate a report of volumes sharing specified attributes.
	Unbind	Remove a volume from a volume set.
	Unload	Unload the specified volume from a drive.
Magazine	Move	Move a magazine from one location or jukebox to another location or jukebox.
Request	Cancel	Cancel an outstanding request before it is completed by the MDMS system.

18.1.2.2 Combined Tasks

The graphic user interface also provides guides for combined tasks. These guides take you through tasks that involve multiple steps on multiple objects.

Add Devices and Volumes

This task interface first takes you through the procedures to add a new jukebox and drive to the MDMS domain. The second part of the procedure takes you through all the steps to add volumes to the MDMS domain. You can use just the second part to add volumes at any time.

Delete Devices and Volumes

Use this task interface to remove a jukebox or drive, and volumes from MDMS management. This procedure provides you with the necessary decisions to make sure the MDMS database is

kept current after all necessary object records have been deleted. Without this procedure, you could likely delete object records, but leave references to them in the attribute fields of remaining records.

Site to Site Rotation

This procedure facilitates moving volumes to an offsite vault location for safe storage. It takes you through the steps to bring volumes from an offsite location, then gather volumes for movement to the offsite location.

Service a Jukebox

Use this procedure when backup operations use volumes in a jukebox and you need to supply free volumes for future backup requests. This procedure allows you to gather allocated volumes from the jukebox, then replace them with free volumes. The procedure also allows you to use the jukebox vision system.

18.2 Access Rights for MDMS Operations

This section describes access rights for MDMS operations. MDMS works with the OpenVMS User Authorization File (UAF), so you need to understand the Authorize Utility and OpenVMS security before changing the default MDMS rights assignments.

MDMS rights control access to operations, not to object records in the database.

Knowing the security implementation will allow you to set up MDMS operation as openly or securely as required.

18.2.1 Description of MDMS Rights

MDMS controls user action with process rights granted to the user or application through low and high level rights.

18.2.1.1 Low Level Rights

The low level rights are named to indicate an action and the object the action targets. For instance, the MDMS_MOVE_OWN right allows the user to conduct a move operation on a volume allocated to that user. The MDMS_LOAD_ALL right allows the user to load any managed volume.

For detailed descriptions of the MDMS low level rights, refer to the ABS or HSM Command Reference Guide.

18.2.1.2 High Level Rights

MDMS associates high level rights with the kind of user that would typically need them. Refer to the ABS or HSM Command Reference Guide for a detailed list of the low level rights associated with each high level right. The remainder of this section describes the high level rights.

MDMS User

The default MDMS_USER right is for any user who wants to use MDMS to manage their own tape volumes. A user with the MDMS_USER right can manage only their own volumes. The default MDMS_USER right does not allow for creating or deleting MDMS object records, or changing the current MDMS configuration.

Use this right for users who perform non-system operations with ABS or HSM.

MDMS Application

The default MDMS_APPLICATION right is for the ABS and HSM applications. As MDMS clients using managed volumes and drives, these applications require specific rights.

The ABS or HSM processes include the MDMS_APPLICATION rights identifier which assumes the low level rights associated with it. Do not modify the low level rights values for the Domain application rights attribute. Changing the values to this attribute can cause your application to fail.

MDMS Operator

The default MDMS_OPERATOR right supports data center operators. The associated low level rights allow operators to service MDMS requests for managing volumes, loading and unloading drives.

The Default Right

The low level rights associated with the MDMS_DEFAULT right apply to any OpenVMS user who does not have any specific MDMS right granted in their user authorization (SYSUAF.DAT) file. Use the default right when all users can be trusted with an equivalent level of MDMS rights.

18.2.2 Granting MDMS Rights

The high level rights are defined by domain object record attributes with lists of low level rights. The high level rights are convenient names for sets of low level rights.

For MDMS users, grant high and/or low level rights as needed with the Authorize Utility. You can take either of these approaches to granting MDMS rights.

You can ensure that all appropriate low level rights necessary for a class of user are assigned to the corresponding high level right, then grant the high level rights to users.

You can grant any combination of high level and low level rights to any user.

Use the procedure outlined in Table 18–3 to review and set rights that enable or disable access to MDMS operations. CLI command examples appear in this process description but can use the GUI to accomplish this procedure as well.

Table 18–3 Reviewing and Setting MDMS Rights

Step...	Action...
1.	Show the domain object record values for each high level right. • For all system users, examine the default rights attribute. • For MDMS operators, examine the operator rights attribute. • For MDMS users, examine the user rights attribute. Review the low level rights associated with each high level right. If you have questions about actions view the list of low level rights and the actions they enable. Example `$MDMS SHOW DOMAIN /FULL`

Table 18–3 Reviewing and Setting MDMS Rights

Step...	Action...

2.	If the low level rights associated with the high level right are not adequate for a class of user, then add appropriate rights.
	
	If the low level rights associated with the high level right enable inappropriate options for a class of user, then remove the inappropriate rights.
	
	Example:
	
	`$MDMS SET DOMAIN /OPERATOR_RIGHTS=MDMS_SET_PROTECTED/ADD`
	
	`or`
	
	`$MDMS SET DOMAIN /USER_RIGHTS=MDMS_ASSIST/REMOVE`

3.	If you do not want all system users to have implicit access to MDMS operations, then negate the domain object record default rights attribute. $MDMS SET DOMAIN /NODEFAULT_RIGHTS By default, a user with the OpenVMS SYSPRV privilege is granted all MDMS rights. If you wish to disable this feature, disable the SYSPRV privilege in the domain record:. $MDMS SET DOMAIN /NOSYSPRV
4.	If you want any user with ABS privileges to have access to appropriate MDMS rights to support just ABS operations, set the domain object record ABS rights attribute. $MDMS SET DOMAIN /ABS_RIGHTS For all system user accounts that need access to MDMS, grant the appropriate rights. If a user needs only the rights associated with a class of user, grant that user the high level right associated with that class only. UAF> GRANT/IDENTIFIER MDMS_USER DEVUSER

Table 18–3 Reviewing and Setting MDMS Rights

Step...	Action...
5.	If a user needs a combination of rights, then grant that user the high and/or low level rights needed to enable the user to do their job with MDMS. You must issue a separate command for each right granted. UAF> GRANT/IDENTIFIER MDMS_OPERATOR DCOPER %UAF-I-GRANTMSG, identifier MDMS_OPERATOR granted to DCOPER UAF> GRANT/IDENTIFIER MDMS_LOAD_SCRATCH DCOPER %UAF-I-GRANTMSG, identifier MDMS_LOAD_SCRATCH granted to DCOPER If you do not want a particular user to acquire the default rights, then disable the user's ability to operate MDMS with the default rights. UAF> GRANT/IDENTIFIER MDMS_NO_DEFAULT APPUSER

18.3 Creating, Modifying, and Deleting Object Records

This section describes the basic concepts that relate to creating, modifying, and deleting object records.

18.3.1 Creating Object Records

Both the CLI and GUI provide the ability to create object records. MDMS imposes rules on the names you give object records. When creating object records, define as many attribute values as you can, or inherit attributes from object records that describe similar objects.

18.3.1.1 Naming Objects

When you create an object record, you give it a name that will be used as long as it exists in the MDMS database. MDMS also accesses the object record when it is an attribute of another object record; for instance a media type object record named as a volume attribute.

MDMS object names may include any digit (0 through 9), any upper case letter (A through Z), and any lower case letter (a through z). Additionally, you can include $ (dollar sign) and _ (underscore).

18.3.1.2 Differences Between the CLI and GUI for Naming Object Records

The MDMS CLI accepts all these characters. However, lower case letters are automatically converted to upper case, unless the string containing them is surrounded by the "(double quote) characters. The CLI also allows you to embed spaces in object names if the object name is surrounded by the " characters.

The MDMS GUI accepts all the allowable characters, but will not allow you to create objects that use lower case names, or embed spaces. The GUI will display names that include spaces and lower case characters if they were created with the CLI.

Compaq recommends that you create all object records with names that include no lower case letters or spaces. If you create an object name with lower case letters, and refer to it as an attribute value which includes upper case letters, MDMS may fail an operation.

Naming Examples

The following examples illustrate the concepts for creating object names with the CLI.

These commands show the default CLI behavior for naming objects:

```
$!Volume created with upper case locked
$MDMS CREATE VOLUME CPQ231 /INHERIT=CPQ000 !Standard upper case DCL
$MDMS SHOW VOLUME CPQ231
$!
$!Volume created with lower case letters
$MDMS CREATE VOLUME cpq232 /INHERIT=CPQ000 !Standard lower case DCL
$MDMS SHOW VOLUME CPQ232
$!
$!Volume created with quote-delimited lower case, forcing lower case naming
$MDMS CREATE VOLUME ìcpq233î /INHERIT=CPQ000 !Forced lower case DCL
$!
$!This command fails because the default behavior translates to upper case
$MDMS SHOW VOLUME CPQ233
$!
$!Use quote-delimited lower case to examine the object record
$MDMS SHOW VOLUME ìcpq233î
```

18.3.2 Inheritance on Creation

This feature allows you to copy the attributes of any specified object record when creating or changing another object record. For instance, if you create drive object records for four drives in a new jukebox, you fill out all the attributes for the first drive object record. Then, use the inherit option to copy the attribute values from the first drive object record when creating the subsequent three drive object records.

If you use the inherit feature, you do not have to accept all the attribute values of the selected object record. You can override any particular attribute value by including the attribute assignment in the command or GUI operation. For CLI users, use the attribute's qualifier with the MDMS CREATE command. For GUI users, set the attribute values you want.

Not all attributes can be inherited. Some object record attributes are protected and contain values that apply only to the specific object the record represents. Check the command reference information to identify object record attributes that can be inherited.

18.3.3 Referring to Non-Existent Objects

MDMS allows you to specify object record names as attribute values before you create the records. For example, the drive object record has a media types attribute. You can enter media type object record names into that attribute when you create the drive object before you create the media type object records.

18.3.4 Rights for Creating Objects

The low level rights that enable a user to create objects are MDMS_CREATE_ALL (create any MDMS object record) and MDMS_CREATE_POOL (create volumes in a pool authorized to the user).

18.3.5 Modifying Object Records

Whenever your configuration changes you will modify object records in the MDMS database. When you identify an object that needs to be changed you must specify the object record as it is named. If you know an object record exists, but it does not display in response to an operation to change it, you could be entering the name incorrectly. Section 18.3.1.1 describes the conventions for naming object records.

18.3.6 Protected Attributes

Do not change protected attributes if you do not understand the implications of making the particular changes. If you change a protected attribute, you could cause an operation to fail or prevent the recovery of data recorded on managed volumes.

MDMS uses some attributes to store information it needs to manage certain objects. The GUI default behavior prevents you from inadvertently changing these attributes. By pressing the Enable Protected button on the GUI, you can change these attributes. The CLI makes no distinction in how it presents protected attributes when you modify object records. Ultimately, the ability to change protected attributes is allowed by the MDMS_SET_PROTECTED right and implicitly through the MDMS_SET_RIGHTS right.

The command reference guide identifies protected attributes

18.3.7 Rights for Modifying Objects

The low level rights that allow you to modify an object by changing its attribute values are shown below:.

Table 18–4 Low Level Rights

This right	Enables you to modify
MDMS_SET_ALL	Any MDMS database object record.
MDMS_SET_PROTECTED	Protected attributes used internally by MDMS.
MDMS_SET_OWN	Attributes of volumes allocated to the user.
MDMS_SET_POOL	Attributes of volumes in pools authorized to the user.
MDMS_SET_RIGHTS	The MDMS domain high level rights definition

18.3.8 Deleting Object Records

When managed objects, such as drives or volumes, become obsolete or fail, you may want to remove them from management. When you remove these objects, you must also delete the object records that describe them to MDMS.

When you remove object records, there are two reviews you must make to ensure the database accurately reflects the management domain: review the remaining object records and change any attributes that reference the deleted object records, review any DCL command procedures and change any command qualifiers that reference deleted object records.

18.3.9 Reviewing Managed Objects for References to Deleted Objects

When you delete an object record, review object records in the database for references to those objects. Table 18–5 shows which object records to check when you delete a given object record. Use this table also to check command procedures that include the MDMS SET command for the remaining objects.

Change references to deleted object records from the MDMS database. If you leave a reference to a deleted object record in the MDMS database, an operation with MDMS could fail.

Table 18–5 Reviewing Managed Objects for References to Deleted Objects

When you delete...	Review these object records...
Group	Drive
	Jukebox
Pool (Authorized, Default Users)	

Table 18–5 Reviewing Managed Objects for References to Deleted Objects

When you delete...	Review these object records...
Jukebox	Drive
Jukebox	
Magazine (MDMS sets the attribute)	
Volume (MDMS sets the attribute)	
Location	Domain (Offsite, Onsite Location)
Location	
Magazine (Offsite, Onsite Location)	
Node	
Volume (Offsite, Onsite Location)	
Media Type	Domain
Drive	
Volume	
Node	Drive
Group	
Jukebox	
Pool (Authorized, Default Users)	
Pool	Volume

18.3.10 Reviewing DCL Command Procedures for References to Deleted Objects

When you delete an object record, review any DCL command procedures for commands that reference those objects. Other than the MDMS CREATE, SET, SHOW, and DELETE commands for a given object record, Table 18–6 shows which commands to check. These commands could have references to the deleted object record.

Change references to deleted object records from DCL commands. If you leave a reference to a deleted object record in a DCL command, an operation with MDMS could fail.

Table 18–6 Reviewing DCL Commands for References to Deleted Objects

When you delete...	Review these DCL commands...
Drive	MDMS ALLOCATE DRIVE
	MDMS DEALLOCATE DRIVE
	MDMS LOAD DRIVE
	MDMS LOAD VOLUME
	MDMS UNLOAD DRIVE
Group	MDMS ALLOCATE DRIVE

Table 18–6 Reviewing DCL Commands for References to Deleted Objects

When you delete...	Review these DCL commands...
	MDMS CREATE DRIVE
	MDMS CREATE JUKEBOX
	MDMS SET DRIVE
	MDMS SET JUKEBOX
Jukebox	MDMS ALLOCATE DRIVE
	MDMS ALLOCATE VOLUME
	MDMS CREATE MAGAZINE
	MDMS CREATE VOLUME
	MDMS INITIALIZE VOLUME
	MDMS INVENTORY JUKEBOX
	MDMS SET MAGAZINE
	MDMS SET VOLUME
	MDMS REPORT VOLUME
Location	MDMS ALLOCATE DRIVE
	MDMS ALLOCATE VOLUME
	MDMS CREATE LOCATION (Location attribute)
	MDMS CREATE JUKEBOX
	MDMS CREATE MAGAZINE (Onsite, Offsite Location)
	MDMS CREATE NODE
	MDMS CREATE VOLUME (Onsite, Offsite Location)
	MDMS MOVE VOLUME
	MDMS REPORT VOLUME (Onsite, Offsite Location Fields)
	MDMS SET DOMAIN (Onsite, Offsite Location)
	MDMS SET JUKEBOX
	MDMS SET LOCATION (Location attribute)
	MDMS SET MAGAZINE (Onsite, Offsite Location)
	MDMS SET NODE
	MDMS SET VOLUME (Onsite, Offsite Location)
Media Type	MDMS ALLOCATE DRIVE
	MDMS ALLOCATE VOLUME
	MDMS CREATE DRIVE
	MDMS CREATE VOLUME

Table 18–6 Reviewing DCL Commands for References to Deleted Objects

When you delete...	Review these DCL commands...
	MDMS INITITALIZE VOLUME
	MDMS INVENTORY JUKEBOX
	MDMS LOAD DRIVE
	MDMS REPORT VOLUME
	MDMS SET DOMAIN
	MDMS SET VOLUME
Node	MDMS ALLOCATE DRIVE
	MDMS CREATE DRIVE
	MDMS CREATE GROUP
	MDMS CREATE JUKEBOX
	MDMS CREATE POOL (Authorized, Default Users)
	MDMS SET DRIVE
	MDMS SET GROUP
	MDMS SET JUKEBOX
	MDMS SET POOL (Authorized, Default Users)
Pool	MDMS ALLOCATE VOLUME
	MDMS LOAD DRIVE
	MDMS REPORT VOLUME
	MDMS SET VOLUME
Volume	MDMS ALLOCATE DRIVE
	MDMS ALLOCATE VOLUME/LIKE_VOLUME
Volume Set	MDMS BIND VOLUME/TO_SET

18.3.11Rights for Deleting Objects

The low level rights that enable a user to delete objects are MDMS_DELETE_ALL (delete any MDMS object record) and MDMS_DELETE_POOL (delete volumes in a pool authorized to the user).

19

Connecting and Managing Remote Devices

This chapter explains how to configure and manage remote devices Remote Device Facility (RDF). Media and Device Management Services (MDMS) and RDF allow you to manage remote devices.

19.1 The RDF Installation

When you install Media and Device Management Services (MDMS) you are asked whether you want to install the RDF software.

During the installation you place the RDF client software on the nodes with disks you want to backup. You place the RDF server software on the systems to which the tape backup devices are connected. This means that when using RDF, you serve the tape backup device to the systems with the client disks.

All of the files for RDF are placed in TTI_RDF: for your system. There will be separate locations for VAX or Alpha.

Note

RDF is not available if you are running ABS/MDMS with the ABS-OMT license.

19.2 Configuring RDF

After installing RDF you should check the TTI_RDEV:CONFIG_nodename.DAT file to make sure it has correct entries.

This file:

- is located on the RDF server node with the tape device
- is created initially during installation
- is a text file
- includes the definition of each device accessible by the RDF software. This definition consists of a physical device name and an RDF characteristic name.

Example:

```
Device $1$MIA0 MIAO
```

Verify:

Check this file to make sure that all RDF characteristic names are unique to this node.

19.3 Using RDF with MDMS

The following sections describe how to use RDF with MDMS.

19.3.1 Starting Up and Shutting Down RDF Software

Starting up RDF software:

RDF software is automatically started up along with then MDMS software when you enter the following command:

```
$ @SYS$STARTUP:MDMS$STARTUP
```

Shutting down RDF software:

To shut down the RDF software, enter the following command:

```
$ @SYS$STARTUP:MDMS$SHUTDOWN
```

19.3.2 The RDSHOW Procedure

Required privileges:

The following privileges are required to execute the RDSHOW procedure: NETMBX, TMPMBX.

In addition, the following privileges are required to show information on remote devices allocated by other processes: SYSPRV,WORLD.

19.3.3 Command Overview

You can run the RDSHOW procedure any time after the MDMS software has been started. RDF software is automatically started at this time.

Use the following procedures:

```
$ @TTI_RDEV:RDSHOW CLIENT
$ @TTI_RDEV:RDSHOW SERVER node_name
$ @TTI_RDEV:RDSHOW DEVICES
```

node_name is the node name of any node on which the RDF server software is running.

19.3.4 Showing Your Allocated Remote Devices

To show remote devices that you have allocated, enter the following command from the RDF client node:

```
$ @TTI_RDEV:RDSHOW CLIENT
```

Result:

```
RDALLOCATED devices for pid 20200294, user DJ, on node OMAHA::
Local logical        Rmt node  Remote device
TAPE01               MIAMI::    MIAMI$MUC0
```

DJ is the user name and OMAHA is the current RDF client node.

19.3.5 Showing Available Remote Devices on the Server Node

The RDSHOW SERVER procedure shows the available devices on a specific SERVER node. To execute this procedure, enter the following command from any RDF client or RDF server node:

```
$ @TTI_RDEV:RDSHOW SERVER MIAMI
```

MIAMI is the name of the server node whose devices you want shown.

Result:

```
Available devices on node MIAMI::
Name        Status  Characteristics/Comments
MIAMI$MSA0   in use  msa0
...by pid 20200246, user CATHY (local)
MIAMI$MUA0   in use  mua0
...by pid 202001B6, user CATHY, on node OMAHA::
MIAMI$MUB0   -free-  mub0
MIAMI$MUC0   in use  muc0
...by pid 2020014C, user DJ, on node OMAHA::
```

This RDSHOW SERVER command shows any available devices on the server node MIAMI, including any device characteristics. In addition, each allocated device shows the process PID, username, and RDF client node name.

The text (local) is shown if the device is locally allocated.

19.3.6 Showing All Remote Devices Allocated on the RDF Client Node

To show all allocated remote devices on an RDF client node, enter the following command from the RDF client node:

```
$ @TTI_RDEV:RDSHOW DEVICES
```

Result:

```
Devices RDALLOCATED on node OMAHA::
RDdevice  Rmt node  Remote device  User name     PID
RDEVA0:     MIAMI::    MIAMI$MUC0      DJ            2020014C
RDEVB0:     MIAMI::    MIAMI$MUA0      CATHY         202001B6
```

This command shows all allocated devices on the RDF client node OMAHA. Use this command to determine which devices are allocated on which nodes.

19.4 Monitoring and Tuning Network Performance

This section describes network issues that are especially important when working with remote devices.

19.4.1 DECnet Phase IV

The Network Control Program (NCP) is used to change various network parameters. RDF (and the rest of your network as a whole) benefits from changing two NCP parameters on all nodes in your network. These parameters are:

- PIPELINE QUOTA
- LINE RECEIVE BUFFERS

Pipeline quota

The pipeline quota is used to send data packets at an even rate. It can be tuned for specific network configurations. For example, in an Ethernet network, the number of packet buffers represented by the pipeline quota can be calculated as approximately:

```
buffers = pipeline_quota / 1498
```

Default:

The default pipeline quota is 10000. At this value, only six packets can be sent before acknowledgment of a packet from the receiving node is required. The sending node stops after the sixth packet is sent if an acknowledgment is not received.

Recommendation:

The PIPELINE QUOTA can be increased to 45,000 allowing 30 packets to be sent before a packet is acknowledged (in an Ethernet network). However, performance improvements have not been verified for values higher than 23,000. It is important to know that increasing the value of PIPELINE QUOTA improves the performance of RDF, but may negatively impact performance of other applications running concurrently with RDF.

Line receive buffers

Similar to the pipeline quota, line receive buffers are used to receive data at a constant rate.

Default:

The default setting for the number of line receive buffers is 6.

Recommendation:

The number of line receive buffers can be increased to 30 allowing 30 packets to be received at a time. However, performance improvements have not been verified for values greater than 15 and as stated above, tuning changes may improve RDF performance while negatively impacting other applications running on the system.

19.4.2 DECnet-Plus (Phase V)

As stated in DECnet-Plus(Phase V), (DECnet/OSI V6.1) Release Notes, a pipeline quota is not used directly. Users may influence packet transmission rates by adjusting the values for the transport's characteristics MAXIMUM TRANSPORT CONNECTIONS, MAXIMUM RECEIVE BUFFERS, and MAXIMUM WINDOW. The value for the transmit quota is determined by MAXIMUM RECEIVE BUFFERS divided by Actual TRANSPORT CONNECTIONS.
This will be used for the transmit window, unless MAXIMUM WINDOW is less than this quota. In that case, MAXIMUM WINDOW will be used for the transmitter window.

The DECnet-Plus defaults (MAXIMUM TRANSPORT CONNECTIONS = 200 and MAXIMUM RECEIVE BUFFERS = 4000) produce a MAXIMUM WINDOW of 20. Decreasing MAXIMUM TRANSPORT CONNECTIONS with a corresponding increase of MAXIMUM WINDO may improve RDF performance, but also may negatively impact other applications running on the system.

19.4.3 Changing Network Parameters

This section describes how to change the network parameters for DECnet Phase IV and DECnet-PLUS.

19.4.4 Changing Network Parameters for DECnet (Phase IV)

The pipeline quota is an NCP executor parameter. The line receive buffers setting is an NCP line parameter.

The following procedure shows how to display and change these parameters in the permanent DECnet database. These changes should be made on each node of the network.

Table 19–1 How to Change Network Parameters

Step	Action

1	Enter:
	`$ run sys$system:NCP`
	`NCP>show executor characteristics`
	
	Result:
	`Node Permanent Characteristics as of 24-MAY-1991 10:10:58`
	`Executor node = 20.1 (DENVER)`
	`Management version = V4.0.0`
	`.`
	`.`
	`.`
	`Pipeline quota = 10000`

2	Enter:
	`NCP>define executor pipeline quota 45000`
	`NCP>show known lines`
	
	Result:
	`Known line Volatile Summary as of 24-MAY-1991 10:11:13`
	`Line  State`
	`SVA-0  on`

3	Enter:
	`NCP>show line sva-0 characteristics`
	
	Result:
	`Line Permanent Characteristics as of 24-MAY-1991 10:11:31`
	`Line = SVA-0`
	`Receive buffers  = 6  <-- value to change`
	`Controller  = normal`
	`Protocol  = Ethernet`
	`Service timer  = 4000`
	`Hardware address  = 08-00-2B-0D-D0-5F`
	`Device buffer size  = 1498`

4	Enter:
	`NCP>define line sva-0 receive buffers 30`
	`NCP>exit`

Requirement:

For the changed parameters to take effect, the node must be rebooted or DECnet must be shut down.

19.4.5 Changing Network Parameters for DECnet-Plus(Phase V)

The Network Control Language (NCL) is used to change DECnet-Plus network parameters. The transport parameters MAXIMUM RECEIVE BUFFERS, MAXIMUM TRANSPORT CONNECTIONS and MAXIMUM WINDOW can be adjusted by using NCL's SET OSI TRANSPORT command. For example:

```
NCL> SET OSI TRANSPORT MAXIMUM RECEIVE BUFFERS = 4000       !default value
NCL> SET OSI TRANSPORT MAXIMUM TRANSPORT CONNECTIONS = 200  !default value
NCL> SET OSI TRANSPORT MAXIMUM WINDOWS = 20                 !default value
```

To make the parameter change permanent, add the NCL command(s) to the SYS$MANAGER:NET$OSI_TRANSPORT_STARTUP.NCL file. Refer to the DENET-Plus (DECnet/OSI) Network Management manual for detailed information.

19.4.6 Resource Considerations

Changing the default values of line receive buffers and the pipeline quota to the values of 30 and 45000 consumes less than 140 pages of nonpaged dynamic memory.

In addition, you may need to increase the number of large request packets (LRPs) and raise the default value of NETACP BYTLM.

Large request packets

LRPs are used by DECnet to send and receive messages. The number of LRPs is governed by the SYSGEN parameters LRPCOUNT and LRPCOUNTV.

Recommendation:

A minimum of 30 free LRPs is recommended during peak times. Show these parameters and the number of free LRPs by entering the following DCL command:

```
$ SHOW MEMORY/POOL/FULL
```

Result:

```
System Memory Resources on 24-JUN-1991 08:13:57.66
Large Packet (LRP) Lookaside List Packets Bytes
Current Total Size                  36     59328
Initial Size (LRPCOUNT)             25     41200
Maximum Size (LRPCOUNTV)            200    329600
Free Space                          20     32960
```

In the LRP lookaside list, this system has:

- Current Total Size of 36

The SYSGEN parameter LRPCOUNT (LRP Count) has been set to 25. The Current Size is not the same as the Initial Size. This means that OpenVMS software has to allocate more LRPs. This causes system performance degradation while OpenVMS is expanding the LRP lookaside list.

The LRPCOUNT should have been raised to at least 36 so OpenVMS does not have to allocate more LRPs.

Recommendation:

Raise the LRPCOUNT parameter to a minimum of 50. Because the LRPCOUNT parameter is set to only 25, the LRPCOUNT parameter is raised on this system even if the current size was also 25.

- Free Space is 20

This is below the recommended free space amount of 30. This also indicates that LRPCOUNT should be raised. Raising LRPCOUNT to 50 (when there are currently 36 LRPs) has the effect of adding 14 LRPs. Fourteen plus the 20 free space equals over 30. This means that the recommended value of 30 free space LRPs is met after LRPCOUNT is set to 50.

- The SYSGEN parameter LRPCOUNTV (LRP count virtual) has been set to 200.

The LRPCOUNTV parameter should be at least four times LRPCOUNT. Raising LRPCOUNT may mean that LRPCOUNTV has to be raised. In this case, LRPCOUNTV does not have to be raised because 200 is exactly four times 50 (the new LRPCOUNT value).

Make changes to LRPCOUNT or LRPCOUNTV in both:

- SYSGEN (using CURRENT)
- SYS$SYSTEM:MODPARAMS.DAT file (for when AUTOGEN is run with REBOOT)

Example: Changing LRPCOUNT to 50 in SYSGEN

```
Username:  SYSTEM
Password:  (the system password)
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
SYSGEN>  USE CURRENT
SYSGEN>  SH LRPCOUNT
Parameter Name             Current   Default   Minimum   Maximum
LRPCOUNT                        25         4         0      4096
SYSGEN>  SET LRPCOUNT 50
SYSGEN>  WRITE CURRENT
SYSGEN>  SH LRPCOUNT
Parameter Name             Current   Default   Minimum   Maximum
LRPCOUNT                        50         4         0      4096
```

Requirement:

After making changes to SYSGEN, reboot your system so the changes take effect.

Example: Changing the LRPCOUNT for AUTOGEN

Add the following line to MODPARAMS.DAT:

```
$ MIN_LRPCOUNT = 50      ! ADDED {the date} {your initials}
```

Result:

This ensures that when AUTOGEN runs, LRPCOUNT is not set below 50.

NETACP BYTLM

The default value of NETACP is a BYTLM setting of 65,535. Including overhead, this is enough for only 25 to 30 line receive buffers. This default BYTLM may not be enough.

Recommendation:

Increase the value of NETACP BYTLM to 110,000.

How to increase NETACP BYTLM:

Before starting DECnet, define the logical NETACP$BUFFER_ LIMIT by entering:

```
$ DEFINE/SYSTEM/NOLOG NETACP$BUFFER_LIMIT 110000
$ @SYS$MANAGER:STARTNET.COM
```

19.4.7 Controlling RDF's Effect on the Network

By default, RDF tries to perform I/O requests as fast as possible. In some cases, this can cause the network to slow down. Reducing the network bandwidth used by RDF allows more of the network to become available to other processes.

The RDF logical names that control this are:

```
RDEV_WRITE_GROUP_SIZE
RDEV_WRITE_GROUP_DELAY
```

Default:

The default values for these logical names is zero. The following example shows how to define these logical names on the RDF client node:

```
$ DEFINE/SYSTEM RDEV_WRITE_GROUP_SIZE  30
$ DEFINE/SYSTEM RDEV_WRITE_GROUP_DELAY  1
```

Further reduction:

To further reduce bandwidth, the RDEV_WRITE_GROUP_DELAY logical can be increased to two (2) or three (3).

Note

Reducing the bandwidth used by RDF causes slower transfers of RDF's data across the network.

19.4.8 Surviving Network Failures

Remote Device Facility (RDF) can survive network failures of up to 15 minutes long. If the network comes back within the 15 minutes allotted time, the RDCLIENT continues processing WITHOUT ANY INTERRUPTION OR DATA LOSS. When a network link drops while RDF is active, after 10 seconds, RDF creates a new network link, synchronizes I/Os between the RDCLIENT and RDSERVER, and continues processing.

The following example shows how you can test the RDF's ability to survive a network failure. (This example assumes that you have both the RDSERVER and RDCLIENT processes running.)

```
$ @tti_rdev:rdallocate tti::mua0:
RDF - Remote Device Facility (Version 4.1) - RDALLOCATE Procedure
Copyright (c) 1990, 1996 Touch Technologies, Inc.
Device TTI::TTI$MUA0 ALLOCATED, use TAPE01 to reference it
$ backup/rewind/log/ignore=label sys$library:*.* tape01:test
```

from a second session:

```
$ run sys$system:NCP
NCP> show known links

Known Link Volatile Summary as of 13-MAR-1996 14:07:38
Link       Node           PID     Process     Remote link  Remote user
24593  20.4 (JR)         2040111C  MARI_11C_5         8244  CTERM
16790  20.3 (FAST)       20400C3A  -rdclient-      16791  tti_rdevSRV
24579  20.6 (CHEERS)     20400113  REMACP             8223  SAMMY
24585  20.6 (CHEERS)     20400113  REMACP             8224  ANDERSON
NCP> disconnect link 16790
.
.
.
```

Backup pauses momentarily before resuming. Sensing the network disconnect, RDF creates a new -rdclient- link. Verify this by entering the following command:

```
NCP> show known links
Known Link Volatile Summary as of 13-MAR-1996 16:07:00

Link       Node           PID     Process     Remote link  Remote user
24593  20.4 (JR)         2040111C  MARI_11C_5         8244  CTERM
24579  20.6 (CHEERS)     20400113  REMACP             8223  SAMMY
24585  20.6 (CHEERS)     20400113  REMACP             8224  ANDERSON
24600  20.3 (FAST)       20400C3A  -rdclient-      24601  tti_rdevSRV
```

```
NCP> exit
```

19.5 Controlling Access to RDF Resources

The RDF Security Access feature allows storage administrators to control which remote devices are allowed to be accessed by RDF client nodes.

19.5.1 Allow Specific RDF Clients Access to All Remote Devices

You can allow specific RDF client nodes access to all remote devices.

Example:

For example, if the server node is MIAMI and access to all remote devices is granted only to RDF client nodes OMAHA and DENVER, then do the following:

1. Edit TTI_RDEV:CONFIG_MIAMI.DAT
2. Before the first device designation line, insert the /ALLOW qualifier

```
Edit TTI_RDEV:CONFIG_MIAMI.DAT
CLIENT/ALLOW=(OMAHA,DENVER)
DEVICE $1$MUA0:  MUA0, TK50
DEVICE MSA0:     TU80, 1600bpi
```

OMAHA and DENVER (the specific RDF CLIENT nodes) are allowed access to all remote devices (MUA0, TU80) on the server node MIAMI.

Requirements:

If there is more than one RDF client node being allowed access, separate the node names by commas.

19.5.2 Allow Specific RDF Clients Access to a Specific Remote Device

You can allow specific RDF client nodes access to a *specific* remote device.

Example:

If the server node is MIAMI and access to MUA0 is allowed by RDF client nodes OMAHA and DENVER, then do the following:

1. Edit TTI_RDEV:CONFIG_MIAMI.DAT
2. Find the device designation line (for example, DEVICE 1MUA0:)
3. At the end of the device designation line, add the /ALLOW qualifier:

```
$ Edit TTI_RDEV:CONFIG_MIAMI.DAT
DEVICE $1$MUA0:  MUA0, TK50/ALLOW=(OMAHA,DENVER)
DEVICE MSA0:     TU80, 1600bpi
```

OMAHA and DENVER (the specific RDF client nodes) are allowed access only to device MUA0. In this situation, OMAHA is not allowed to access device TU80.

19.5.3 Deny Specific RDF Clients Access to All Remote Devices

You can deny access from specific RDF client nodes to all remote devices. For example, if the server node is MIAMI and you want to deny access to all remote devices from RDF client nodes OMAHA and DENVER, do the following:

1. Edit TTI_RDEV:CONFIG_MIAMI.DAT
2. Before the first device designation line, insert the /DENY qualifier:

```
$ Edit TTI_RDEV:CONFIG_MIAMI.DAT
CLIENT/DENY=(OMAHA,DENVER)
DEVICE $1$MUA0:  MUA0, TK50
DEVICE MSA0:     TU80, 16700bpi
```

OMAHA and DENVER are the specific RDF client nodes denied access to all the remote devices (MUA0, TU80) on the server node MIAMI.

19.5.4 Deny Specific RDF Clients Access to a Specific Remote Device

You can deny specific client nodes access to a specific remote device.

Example:

If the server node is MIAMI and you want to deny access to MUA0 from RDF client nodes OMAHA and DENVER, do the following:

1. Edit TTI_RDEV:CONFIG_MIAMI.DAT
2. Find the device designation line (for example, DEVICE 1MUA0:)
3. At the end of the device designation line, add the /DENY qualifier:

```
$ Edit TTI_RDEV:CONFIG_MIAMI.DAT
DEVICE $1$MUA0:  MUA0, TK50/DENY=(OMAHA,DENVER)
DEVICE MSA0:     TU80, 16700bpi
```

OMAHA and DENVER RDF client nodes are denied access to device MUA0 on the server node MIAMI.

19.6 RDserver Inactivity Timer

One of the features of RDF is the RDserver Inactivity Timer. This feature gives system managers more control over rdallocated devices.

The purpose of the RDserver Inactivity Timer is to rddeallocate any rdallocated device if NO I/O activity to the rdallocated device has occurred within a predetermined length of time. When the RDserver Inactivity Timer expires, the server process drops the link to the client node and deallocates the physical device on the server node. On the client side, the client process deallocates the RDEVn0 device.

The default value for the RDserver Inactivity Timer is 3 hours.

The RDserver Inactivity Timer default value can be manually set by defining a system wide logical on the RDserver node prior to rdallocating on the rdclient node. The logical name is RDEV_SERVER_INACTIVITY_TIMEOUT.

To manually set the timeout value:

```
$ DEFINE/SYSTEM RDEV_SERVER_INACTIVITY_TIMEOUT seconds
```

For example, to set the RDserver Inactivity Timer to 10 hours, you would execute the following command on the RDserver node:

```
$ DEFINE/SYSTEM RDEV_SERVER_INACTIVITY_TIMEOUT 36000
```

19.7 RDF Error Messages

CLIDENY	Access from this CLIENT to the SERVER is not allowed. Check for "CLIENT/ALLOW" in the RDserver's configuration file.
CLIENTSBUSY	All 16 pesudo-devices are already in use.

CLIDENY	Access from this CLIENT to the SERVER is not allowed. Check for "CLIENT/ALLOW" in the RDserver's configuration file.
DEVDENY	Client is not allowed to the Device or to the Node. This error message is dependent on the "CLIENT/ALLOW", "/ALLOW" or "CLIENT/DENY", "/DENY" qualifiers in the configuration file. Verify that the configuration file qualifier is used appropriately.
EMPTYCFG	The RDserver's configuration file has no valid devices or they are all commented out.
LINKABORT	The connection to the device was aborted. For some reason the connection was interrupted and the remote device could not be found. Check the configuration file as well as the remote device.
NOCLIENT	The RDdriver was not loaded. Most commonly the RDCLIENT_STARTUP.COM file was not executed for this node.
NOREMOTE	This is a RDF status message. The remote device could not be found. Verify the configuration file as well as the status of the remote device.
SERVERTMO	The RDserver did not respond to the request. Most commonly the RDSERVER_ STARTUP.COM file was not executed on the server node. Or, the server has too many connections already to reply in time to your request.

20

MDMS Management Operations

20.1 Managing Volumes

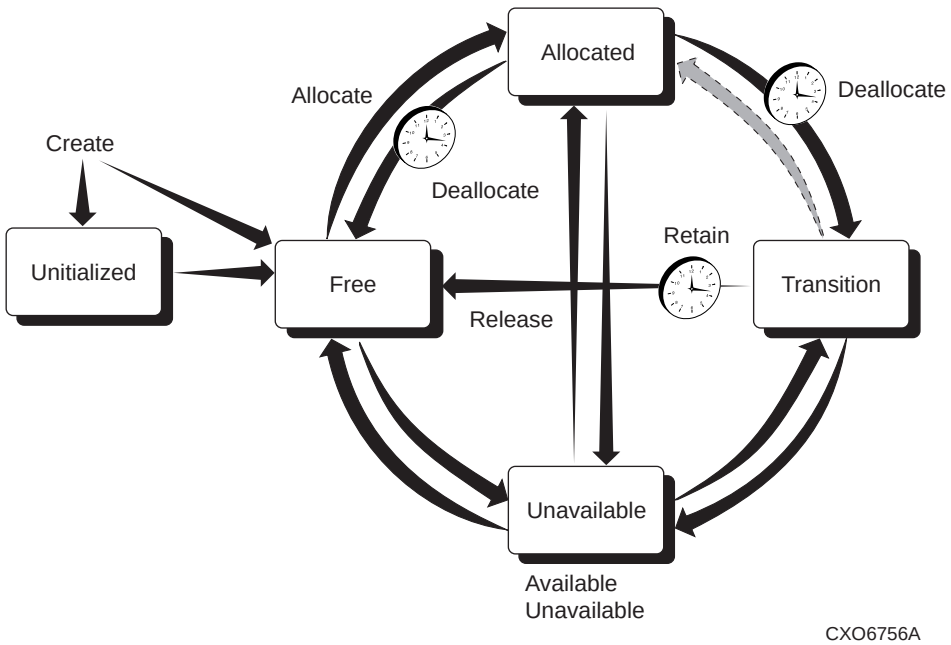
MDMS manages volume availability with the concept of a life cycle. The primary purpose of the life cycle is to ensure that volumes are only written when appropriate, and by authorized users. By setting a variety of attributes across multiple objects, you control how long a volume, once written, remains safe. You also set the time and interval for a volume to stay at an offsite location for safe keeping, then return for re-use once the interval passes.

This section describes the volume life cycle, relating object attributes, commands and life cycle states. This section also describes how to match volumes with drives by creating media type object records.

20.1.1 Volume Life Cycle

The volume life cycle determines when volumes can be written, and controls how long they remain safe from being overwritten. Table 20–1 describes operations on volumes within the life cycle.

Figure 20–1 Volume States



Each row describes an operation with current and new volume states, commands and GUI actions that cause volumes to change states, and if applicable, the volume attributes that MDMS uses to cause volumes to change states. Descriptions following the table explain important aspects of each operation.

Table 20–1 MDMS Volume State Transitions

Current State	Transition to New State	New State
Blank	MDMS CREATE VOLUME Volume Create	UNINTIALIZED
Blank	MDMS CREATE VOLUME/PREINIT	FREE
UNINITIALIZED	MDMS INITIALIZE VOLUME Volume Initialize	FREE
FREE	MDMS INITIALIZE VOLUME Volume Initialize	FREE
FREE	MDMS ALLOCATE VOLUME Volume Allocate	ALLOCATED
ALLOCATED	MDMS DEALLOCATE VOLUME Volume Deallocate or automatically on the volume scratch date	TRANSITION
ALLOCATED	MDMS DEALLOCATE VOLUME Volume Deallocate or automatically on the volume scratch date	FREE
TRANSITION	MDMS SET VOLUME /RELEASE Volume Release or automatically on the volume transition time	FREE
Any State	MDMS SET VOLUME /UNAVAILABLE Volume Unavailable	UNINITIALIZED
UNINITIALIZED	MDMS SET VOLUME /AVAILABLE Volume Available	Previous State
UNINITIALIZED	MDMS DELETE VOLUME Volume Delete	BLANK
FREE	MDMS DELETE VOLUME Volume Delete	BLANK

20.1.2 Volume States by Manual and Automatic Operations

This section describes the transitions between volume states. These processes enable you to secure volumes from unauthorized use by MDMS client applications, or make them available to meet continuing needs. Additionally, in some circumstances, you might have to manually force a volume transition to meet an operational need.

Understanding how these volume transitions occur automatically under MDMS control, or take place manually will help you manage your volumes effectively.

20.1.2.1 Creating Volume Object Records

You have more than one option for creating volume object records. You can create them explicitly with the MDMS CREATE VOLUME command: individually, or for a range of volume identifiers.

You can create the volumes implicitly as the result of an inventory operation on a jukebox. If an inventory operation finds a volume that is not currently managed, a possible response (as you determine) is to create a volume object record to represent it.

You can also create volume object records for large numbers of volumes by opening the jukebox, loading the volumes into the jukebox slots, then running an inventory operation.

Finally, it is possible to perform scratch loads on standalone or stacker drives using the MDMS LOAD DRIVE /CREATE command. If the volume that is loaded is does not exist in the database, MDMS will create it.

You must create volumes explicitly through the MDMS CREATE VOLUME command, or implicitly through the inventory or load operations.

20.1.2.2 Initializing a Volume

Caution

MDMS expects the internally initialized volume label on the physical medium will match the printed label. Always initialize volumes so the recorded volume labels match the printed labels. If the recorded volume label on the tape does not match the printed label on the cartridge, MDMS operations will fail.

Use the MDMS initialize feature to make sure that MDMS recognizes volumes as initialized. Unless you acquire preinitialized volumes, you must explicitly initialize them MDMS before you can use them. If your operations require, you can initialize volumes that have just been released from allocation.

When you initialize a volume or create a volume object record for a preinitialized volume, MDMS records the date in the initialized date attribute of the volume object record.

20.1.2.3 Allocating a Volume

Typically, applications request the allocation of volumes. Only in rare circumstances will you have to allocate a volume to a user other than ABS or HSM. However, if you use command procedures for customized operations that require the use of managed media, you should be familiar with the options for volume allocation. Refer to the ABS or HSM Command Reference Guide for more information on the MDMS ALLOCATE command.

Once an application allocates a volume, MDMS allows read and write access to that volume only by that application. MDMS sets volume object record attributes to control transitions between volume states. Those attributes include:

- the allocated date attribute contains the date and time MDMS allocates the volume.
- the scratch date attribute contains the date and time MDMS will deallocate the volume.

The application requesting the volume can direct MDMS to set additional attributes for controlling how long it keeps the volume and how it releases it. These attributes include:

- the scratch date attributes indicates the date when MDMS automatically sets the volume to a non-allocated state. A volume reaching the scratch date may be either free for use, or may be placed in a transition state.
- the transition time attribute contains the time interval a volume remains in the transition state. The transition state allows you to buffer, or stage, the release of volumes between their allocation (for keeping data safe) and their subsequent re-use (overwriting data). To release volumes directly to a free state, negate the attribute.

20.1.2.4 Holding a Volume

MDMS allows no other user or application to load or unload a volume with the state attribute value set to ALLOCATED, unless the user has MDMS_LOAD_ALL rights. This volume state allows you to protect your data. Set the amount of time a volume remains allocated according to your data retention requirements.

During this time, you can choose to move the volume to an offsite location.

20.1.2.5 Freeing a Volume

When a volume's scratch date passes, MDMS automatically frees the volume from allocation.

If the application or user negates the volume object record scratch date attribute, the volume remains allocated permanently.

Use this feature when you need to retain the data on the volume indefinitely.

After the data retention time has passed, you have the option of making the volume immediately available, or you can elect to hold the volume in a TRANSITION state. To force a volume through the TRANSITION state, negate the volume object record transition time attribute.

You can release a volume from transition with the DCL command MDMS SET VOLUME /RELEASE. Conversely, you can re-allocate a volume from either the FREE or TRANSITION states with the DCL command MDMS SET VOLUME /RETAIN.

Once MDMS sets a volume's state to FREE, it can be allocated for use by an application once again.

20.1.2.6 Making a Volume Unavailable

You can make a volume unavailable if you need to prevent ongoing processing of the volume by MDMS. MDMS retains the state from which you set the UNAVAILABLE state. When you decide to return the volume for processing, the volume state attribute returns to its previous value.

The ability to make a volume unavailable is a manual feature of MDMS.

20.1.3 Matching Volumes with Drives

MDMS matches volumes with drives capable of loading them by providing the logical media type object. The media type object record includes attributes whose values describe the attributes of a type of volume.

The domain object record names the default media types that any volume object record will take if none is specified.

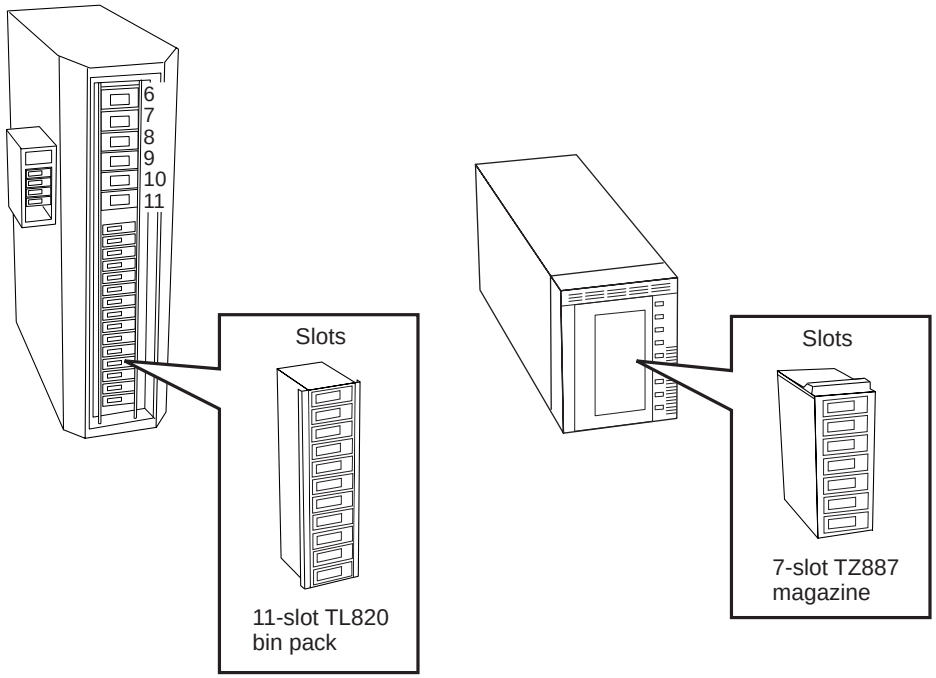
Create a media type object record to describe each type of volume. Drive object records include an attribute list of media types the drive can load, read, and write.

Volume object records for uninitialized volumes include a list of candidate media types. Volume object records for initialized volumes include a single attribute value that names a media type. To allocate a drive for a volume, the volume's media type must be listed in the drive object record's media type field, or its read-only media-type field for read-only operations.

20.1.4 Magazines for Volumes

Use magazines when your operations allow you to move and manage groups of volumes for single users. Create a magazine object record, then move volumes into the magazine (or similar carrier) with MDMS. All the volumes can now be moved between locations and jukeboxes by moving the magazine to which they belong.

Figure 20–2 Magazines



The jukeboxes must support the use of magazines; that is, they must use carriers that can hold multiple volumes at once. If you choose to manage the physical movement of volumes with magazines, then you may set the usage attribute to MAGAZINE for jukebox object records of jukeboxes that use them. You may also define the topology attribute for any jukebox used for magazine based operations.

If your jukebox does not have ports, and requires you to use physical magazines, you do not have to use the MDMS magazine object record. The jukebox can still access volumes by slot number. Single volume operations can still be conducted by using the move operation on individual volumes, or on a range of volumes.

20.1.5 Symbols for Volume Attributes

MDMS provides a feature that allows you to define a series of OpenVMS DCL symbols that describe the attributes of a given volume. By using the /SYMBOLS qualifier with the MDMS SHOW VOLUME command, you can define symbols for all the volume object record attribute values. Use this feature interactively, or in DCL command procedures, when you need to gather information about volumes for subsequent processing.

Refer to the ABS or HSM Command Reference Guide description of the MDMS SHOW VOLUME command.

20.2 Managing Operations

MDMS manages volumes and devices as autonomously as possible. However, it is sometimes necessary - and perhaps required - that your operations staff be involved with moving volumes or loading volumes in drives. When MDMS cannot conduct an automatic operation, it sends a message through the OpenVMS OPCOM system to an operator terminal to request assistance.

Understanding this information will help you set up effective and efficient operations with MDMS.

20.2.1 Setting Up Operator Communication

This section describes how to set up operator communication between MDMS and the OpenVMS OPCOM facility. Follow the steps in Table 20–2 to set up operator communication.

Table 20–2 Setting Up Operator Communication

Step...	Action...
1.	Check or set OPCOM classes for each MDMS node.
2.	Identify the operator terminals nearest to MDMS locations, drives, and jukeboxes.
3.	Enable the operator terminals to receive communication through the OPCOM classes set.

20.2.1.1 Set OPCOM Classes by Node

Set the domain object record OPCOM attribute with the default OPCOM classes for any node in the MDMS management domain.

Each MDMS node has a corresponding node object record. An attribute of the node object record is a list of OPCOM classes through which operator communication takes place. Choose one or more OPCOM classes for operator communication to support operations with this node.

20.2.1.2 Identify Operator Terminals

Identify the operator terminals closest to MDMS locations, drives and jukeboxes. In that way, you can direct the operational communication between the nodes and terminals whose operators can respond to it.

20.2.1.3 Enable Terminals for Communication

Make sure that the terminals are configured to receive OPCOM messages from those classes. Use the OpenVMS REPLY/ENABLE command to set the OPCOM class that corresponds to those set for the node or domain.

```
$REPLY/ENABLE=(opcom_class,[...])
```

Where *opcom_class* specifications are those chosen for MDMS communication.

20.2.2 Activities Requiring Operator Support

Several commands include an assist feature where you can either require or forego operator involvement. Other MDMS features allow you to communicate with particular OPCOM classes, making sure that specific operators get messages. You can configure jukebox drives for automatic loading, and stand alone drives for operator supported loading. See Table 20–3 for a list of operator communication features and your options for using them.

Table 20–3 Operator Management Features

Use These Features...	To Manage These Operations...
Domain and node object records, OPCOM classes attribute	Use this attribute of the node and domain object records to identify the operator terminals to receive OPCOM messages. The domain OPCOM classes apply if none are specified for any node.

Table 20–3 Operator Management Features

Use These Features...	To Manage These Operations...
Drive and jukebox object records, automatic reply attribute	Use this attribute to control whether operator acknowledgments are required for certain drive and jukebox operations. The default (negated) value requires operator acknowledgment for all operations. Setting the attribute to the affirmative will result in MDMS polling the devices for most operations, and completing the request without specific operator acknowledgment. The operator should observe the OPCOM message and look for one of two phrases: • "and reply when completed" - this means that the OPCOM message must be acknowledged before the request will continue • "(auto-reply enabled)" - this means that the OPCOM message will be automatically cancelled and the request will continue after the requested action has been performed
Assist or noassist options and the reply option for these commands or actions: – Allocate drive – Initialize volume – Load drive – Load volume – Move magazine – Move volume – Unload drive – Unload volume	For all listed commands, you can either request or forego operator assistance. When you use the assist option, MDMS will communicate with the operators specified by the OPCOM classes set in the domain object record. Using the noassist option directs MDMS not to send operator messages. You must be granted the MDMS_ASSIST right to use the assist option. The reply option allows you to capture the operator reply to the command. This feature facilitates the use of DCL command procedures to manage interaction with operators.
The message option for these commands: – Load drive – Load volume	For load operations, use the message option to pass additional information to the operator identified to respond to the load request.

20.3 Serving Clients of Managed Media

Once configured, MDMS serves ABS and HSM with uninterrupted access to devices and volumes for writing data. Once allocated, MDMS catalogs volumes to keep them safe, and makes them available when needed to restore data.

To service ABS and HSM, you must supply volumes for MDMS to make available, enable MDMS to manage the allocation of devices and volumes, and meet client needs for volume retention and rotation.

20.3.1 Maintaining a Supply of Volumes

To create and maintain a supply of volumes, you must regularly add volumes to MDMS management, and set volume object record attributes to allow MDMS to meet ABS and HSM needs.

20.3.1.1 Preparing Managed Volumes

To prepare volumes for use by MDMS, you must create volume object records for them and initialize them if needed. MDMS provides different mechanisms for creating volume object records: the create, load, and inventory operations. When you create volume object records, you should consider these factors:

- The situational demands under which you create the volume object records.
- The application needs of the volumes for which you create object records.
- Those additional aspects of the volume for which you will have little, if any, need to change later on.

The following sections provide more detailed information.

Meeting Situational Demands

If you create volume object records with the use of a vision equipped jukebox, you must command MDMS to use the jukebox vision system and identify the slots in which the new volumes reside. These two operational parameters must be supplied to either the create or inventory operation.

For command driven operations, these two commands are functionally equivalent.

```
$MDMS INVENTORY JUKEBOX jukebox_name /VISION/SLOTS=slot_range /CREATE
$MDMS CREATE VOLUME /JUKEBOX=jukebox_name /VISION/SLOTS=slot_range
```

If you create volume object records with the use of a jukebox that does not have a vision system, **you must supply the range of volume names as they are labelled and as they occupy the slot range**.

If you create volume object records for volumes that reside in a location other than the default location (as defined in the domain object record), you must identify the placement of the volumes and the location in the onsite or offsite attribute. Additionally, you must specify the volume name or range of volume names.

If you create volume object records for volumes that reside in the default onsite location, you need not specify the placement or onsite location. However, you must specify the volume name or range of volume names.

Meeting Application Needs

If you acquire preinitialized volumes for MDMS management, and you want to bypass the MDMS initialization feature, you must specify a single media type attribute value for the volume.

Select the format to meet the needs of your MDMS client application. For HSM, use the BACKUP format. For ABS, use BACKUP or RMUBACKUP.

Use a record length that best satisfies your performance requirements. Set the volume protection using standard OpenVMS file protection syntax. Assign the volume to a pool you might use to manage the consumption of volumes between multiple users.

Static Volume Attributes

Static volume attributes rarely, if ever, need to be changed. MDMS provides them to store information that you can use to better manage your volumes.

The description attribute stores up to 255 characters for you to describe the volume, its use, history, or any other information you need.

The brand attribute identifies the volume manufacturer.

Use the record length attribute to store the length or records written to the volume, when that information is needed.

20.3.2 Servicing a Stand Alone Drive

If you use a stand alone drive, enable MDMS operator communication on a terminal near the operator who services the drive. MDMS signals the operator to load and unload the drive as needed.

You must have a ready supply of volumes to satisfy load requests. If your application requires specific volumes, they must be available, and the operator must load the specific volumes requested.

To enable an operator to service a stand alone drive during MDMS operation, perform the actions listed in Table 20–4.

Table 20–4 Configuring MDMS to Service a Stand Alone Drive

Stage...	Action...
1.	Enable operator communication between nodes and terminals.
2.	Stock the location where the drive resides with free volumes.
3.	For all subsequent MDMS actions involving the drive, use the assist feature.

20.3.3 Servicing Jukeboxes

MDMS incorporates many features that take advantage of the mechanical features of automated tape libraries and other medium changers. Use these features to support lights-out operation, and effectively manage the use of volumes.

Jukeboxes that use built-in vision systems to scan volume labels provide the greatest advantage. If the jukebox does not have a vision system, MDMS has to get volume names by other means. For some operations, the operator provides volume names individually or by range. For other operations, MDMS mounts the volume and reads the recorded label.

20.3.3.1 Inventory Operations

The inventory operation registers the contents of a jukebox correctly in the MDMS database. You can use this operation to update the contents of a jukebox whenever you know, or have reason to suspect the contents of a jukebox have changed without MDMS involvement.

Note

Changing the contents of a jukebox without using MDMS move or inventory features, and not updating the MDMS database, will cause subsequent operations to fail. Always use the MDMS INVENTORY operation to make sure the MDMS database accurately reflects the contents of the jukebox whenever you know, or have reason to suspect the contents of a jukebox has changed.

Inventory for Update

When you need to update the database in response to unknown changes in the contents of the jukebox, use the inventory operation against the entire jukebox. If you know the range of slots subject to change, then constrain the inventory operation to just those slots.

If you inventory a jukebox that does not have a vision system, MDMS loads and mounts each volume, to read the volume's recorded label.

Note

Running an inventory on a large number of slots without a vision system can take from tens of minutes to several hours.

When you inventory a subset of slots in the jukebox, use the option to ignore missing volumes.

If you need to manually adjust the MDMS database to reflect the contents of jukebox, use the nophysical option for the MDMS move operation. This allows you to perform a logical move for to update the MDMS database.

Inventory to Create Volume Object Records

If you manage a jukebox, you can use the inventory operation to add volumes to MDMS management. The inventory operation includes the create, preinitialized, media types, and inherit qualifiers to support such operations.

Take the steps in Table 20–5 to use a vision jukebox to create volume object records.

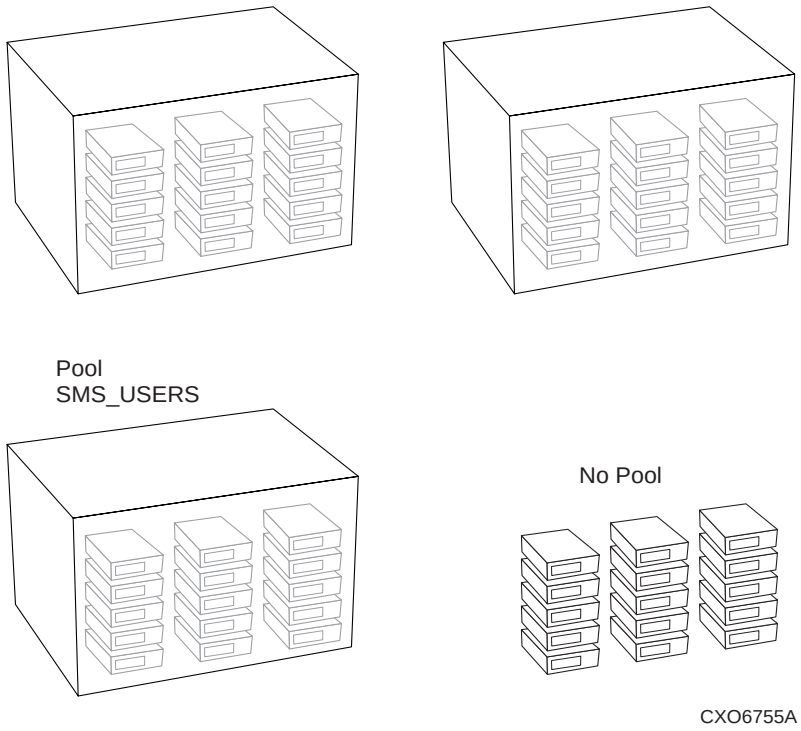
Table 20–5 How to Create Volume Object Records with INVENTORY

Step...	Action...
1.	If you plan to open the jukebox for this operation, disable the jukebox and all drives inside it.
2.	Empty as many slots as necessary to accommodate the volumes. If you cannot open the jukebox, use the MDMS MOVE command to keep the MDMS database synchronized with the actual location of volumes removed. If you open the jukebox and manually remove managed volumes, place the volumes in the location specified by the volumes' onsite location.
3.	Place labelled volumes in the open jukebox slots. If you cannot open the jukebox to expose the slots, use the Media Robot Utility software or front panel controls to move volumes to the slots.
4.	Perform the MDMS inventory operation. Use the create option to signal MDMS to create volume object records. If volumes are initialized specify the preinitialized option and a single media type name for the media types attribute, otherwise, just specify all possible media types to which the volume could relate. Use the inherit option to identify a volume object record from which to inherit other volume attribute values. Use the slots option to specify the range of slots occupied by the volumes to be managed. If the jukebox does not have a vision system, use the volume range and novision options.

20.3.4 Managing Volume Pools

To assist with accounting for volume use by data center clients, MDMS provides features that allow you to divide the volumes you manage by creating volume pools and assigning volumes to them.

Figure 20–3 Pools and Volumes



Use MDMS to specify volume pools. Set the volume pool options in ABS or HSM to specify that volumes be allocated from those pools for users as needed. Figure 20–3 identifies the pools respective to a designated group of users. Note that 'No Pool' is for use by all users.

20.3.4.1 Volume Pool Authorization

The pool object record includes two attributes to assign pools to users: authorized users, and default users.

Set the authorized users list to include all users, by node or group name, who are allowed to allocate volumes from the pool.

Set the default users list to include all users, by node or group name, for whom the pool will be the default pool. Unless another pool is specified during allocation, volumes will be allocated from the default pool for users in the default users list.

Because volume pools are characterized in part by node or group names, anytime you add or remove nodes or groups, you must review and adjust the volume pool attributes as necessary.

20.3.4.2 Adding Volumes to a Volume Pool

After you create a volume pool object record, you can associate managed volumes with it. Select the range of volumes you want to associate with the pool and set the pool attribute of the volumes to the name of the pool.

This can be done during creation or at any time the volume is under MDMS management.

20.3.4.3 Removing Volumes from a Volume Pool

There are three ways to remove volumes from a volume pool.

- You can delete the volume object records.

- You can set the pool attribute of selected volume object records to a different volume pool name.
- You can negate the pool attribute of selected volume object records.

20.3.4.4 Changing User Access to a Volume Pool

To change access to volume pools, modify the membership of the authorized users list attribute.

If you are using the command line to change user access to volume pools, use the /ADD and /REMOVE command qualifiers to modify the current list contents. Use the /NOAUTHORIZED_USERS qualifier to erase the entire user list for the volume pool.

If you are using the GUI to change user access to volume pools, just edit the contents of the authorized users field.

You can also authorize users with the /DEFAULT_USERS attribute, which means that the users are authorized, and that this pool is the pool for which allocation requests for volumes are applied if no pool is specified in the allocation request. You should ensure that any particular user has a default users entry in only one pool.

20.3.4.5 Deleting Volume Pools

You can delete volume pools. However, deleting a volume pool may require some additional clean up to maintain the MDMS database integrity. Some volume records could still have a pool attribute that names the pool to be deleted, and some DCL command procedures could still reference the pool.

If volume records naming the pool exist after deleting the pool object record, find them and change the value of the pool attribute.

The MDMS CREATE VOLUME and MDMS LOAD DRIVE commands in DCL command procedures can specify the deleted pool. Change references to the delete pool object record, if they exist, to prevent the command procedures from failing.

20.3.5 Taking Volumes Out of Service

You might want to remove volumes from management for a variety of reasons:

- You need to retain the information recorded on a volume, and remove any MDMS management access to it.
- The volume cartridge has broken.
- The volume has become unreliable.

20.3.5.1 Temporary Volume Removal

To temporarily remove a volume from management, set the volume state attribute to UNAVAILABLE. Any volume object record with the state set to UNAVAILABLE remains under MDMS management, but is not processed though the life cycle. These volumes will not be set to the TRANSITION or FREE state. However, these volumes can be moved and their location maintained.

20.3.5.2 Permanent Volume Removal

Caution

Before you remove a volume from the MDMS database, MAKE SURE the volume is not storing information for ABS or HSM. If you remove a volume from MDMS management that is referenced from ABS or HSM, you will not be able to restore the data stored on it.

To permanently remove a volume from management, delete the volume object record describing it.

20.4 Rotating Volumes from Site to Site

Volume rotation involves moving volumes to an off-site location for safekeeping with a schedule that meets your needs for data retention and retrieval. After a period of time, you can retrieve volumes for re-use, if you need them. You can rotate volumes individually, or you can rotate groups of volumes that belong to magazines.

20.4.1 Required Preparations for Volume Rotation

The first thing you have to do for a volume rotation plan is create location object records for the on-site and off-site locations. Make sure these location object records include a suitable description of the actual locations. You can optionally specify hierarchical locations and/or a range of spaces, if you want to manage volumes by actual space locations.

You can define as many different locations as your management plan requires.

Once you have object records that describe the locations, choose those that will be the domain defaults (defined as attributes of the domain object record). The default locations will be used when you create volumes or magazines and do not specify onsite and/or offsite location names. You can define only one onsite location and one offsite location as the domain default at any one time.

20.4.2 Sequence of Volume Rotation Events

Manage the volume rotation schedule with the values of the offsite and onsite attributes of the volumes or magazines you manage. You set these values. In addition to setting these attribute values, you must check the schedule periodically to select and move the volumes or magazines.

Table 20–6 shows the sequence of volume rotation events and identifies the commands and GUI actions you issue.

Table 20–6 Sequence of Volume Rotation Events

Stage...	Action...
1.	Set the volume object record onsite and offsite attributes. • Typically, once ABS has allocated a volume you will remove it until it is about to reach the scratch date. Set the onsite location and date based on when it will be freed. Set the offsite location and date based on when it will be ready to be moved offsite. However, make sure that the volume is not part of an ABS continuation set and still needed for subsequent ABS operation. • For HSM, identify volumes to go offsite based on the last access date. If a volume has not been accessed for a long period of time, there has been no need to unshelve the files stored on it. Set the offsite date based for any time after the last access. If multiple archive classes are used, the secondary archive class(es) can be removed off site as soon as a volume is filled. Set the onsite date for any time you might want to archive or delete the files on the volume.

Table 20–6 Sequence of Volume Rotation Events

Stage...	Action...

2.	Identify the volumes or magazines to be moved offsite by selecting the offsite schedule option. You can use the MDMS report or show volume features, or the show magazine feature. The following CLI examples illustrate this:
	`$MDMS SHOW VOLUME/SCHEDULE=OFFSITE`
	`$MDMS SHOW MAGAZINE/SCHEDULE=OFFSITE`

3.	Move the volumes offsite. With the GUI, you can move the volumes selected from the display. With the CLI, (interactive or command procedure) use the MDMS MOVE command with the /SCHEDULE qualifier. For example: `$MDMS MOVE VOLUME /SCHEDULE=OFFSITE` *`[location_name]`* `$MDMS MOVE MAGAZINE /SCHEDULE=OFFSITE` *`[location_name]`* MDMS communicates with operators through OPCOM, providing a list of volume identifiers for the volumes to be gathered and moved.
4.	If you need to retrieve volumes or magazines to service a restore or unshelve request, you must physically move them back to the onsite location. Use the MDMS GUI move feature for the selected volumes or magazines or use the CLI MOVE command. For example: `$MDMS MOVE VOLUME` *`volume_id location_name`* `$MDMS MOVE MAGAZINE` *`magazine_id location_name`*
5.	To return volumes to the onsite location based on their scheduled return date, use the GUI to select and move volumes and magazines based on their onsite schedule. With the GUI, you can move the volumes selected from the display. With the CLI, (interactive or command procedure) use the MDMS MOVE command with the /SCHEDULE qualifier. For example: `$MDMS MOVE VOLUME /SCHEDULE=ONSITE` *`volume_id location_name`* `$MDMS MOVE MAGAZINE /SCHEDULE=ONSITE -` `$_` *`magazine_name location_name`*
6.	Once the volumes and magazines arrive at the onsite location, negate the offsite and onsite schedules. This prevents the volumes from showing up in subsequent reports. With the GUI, remove the location date values associated with the offsite and onsite attributes. With the CLI, use the /NOONSITE and /NOOFFSITE qualifiers. For example: `SET VOLUME` *`volume_id`* `/NOONSITE /NOOFFSITE`

20.5 Scheduled Activities

MDMS starts three scheduled activities at 1AM, by default, to do the following:

- Deallocate all volumes in the database that have exceeded their scratch date.
- Release all volumes in the database that have exceeded their transition time.
- Schedule all volumes that have exceeded their onsite or offsite date.
- Schedule all magazines that have exceeded their onsite or offsite date.

These three activities are controlled by a logical, are separate jobs with names, generate log files, and notify users when volumes are deallocated. These things are described in the sections below.

20.5.1 Logical Controlling Scheduled Activities

The start time for *scheduled activities* is controlled by the logical:

```
MDMS$SCHEDULED_ACTIVITIES_START_HOUR
```

By default, the scheduled activities start a 1AM which is defined as:

```
$ DEFINE/SYSTEM/NOLOG MDMS$SCHEDULED_ACTIVITIES_START_HOUR 1
```

You can change when the scheduled activities start by changing this logical in SYS$STARTUP:MDMS$SYSTARTUP.COM. The hour must be an integer between 0 and 23.

20.5.2 Job Names of Scheduled Activities

When these scheduled activities jobs start up, they have the following names:

- MDMS$DEALVOL - deallocates and releases volumes
- MDMS$MOVVOL - moves scheduled volumes
- MDMS$MOVMAG - moves scheduled magazines

If any volumes are deallocated, the users in the Mail attribute of the Domain object will receive notification by VMS mail.

Operators will receive Opcom requests to move the volumes or magazines.

20.5.3 Log Files for Scheduled Activities

These scheduled activities generate log files. These log files are located in MDMS$LOGFILE_LOCATION and are named:

- MDMS$DEALVOL.LOG - for deallocating and releasing volumes
- MDMS$MOVVOL - for moving of scheduled volumes
- MDMS$MOVMAG - for moving of scheduled magazines

These log files do not show which volumes or magazines were acted upon. They show the command that was executed and whether it was successful or not.

If the Opcom message is not replied to by the time the next scheduled activities is started, the activity is cancel and a new activity is scheduled. For example, nobody replied to the message from Saturday at 1AM, so on Sunday MDMS canceled the request and generated a new request. The log file for Saturday night would look like this:

```
$ SET VERIFY
$ SET ON
$ MDMS MOVE VOL */SCHEDULE
%MDMS-E-CANCELED, request canceled by user
MDMS$SERVER job terminated at 25-APR-1999 01:01:30.48
```

Nothing is lost because the database did not change, but this new request could require more volumes or magazines to be moved.

The following shows an example that completed successfully after deallocating and releasing the volumes:

```
$ SET VERIFY
$ SET ON
$ MDMS DEALLOCATE VOLUME /SCHEDULE/VOLSET
MDMS$SERVER job terminated at 25-APR-1999 01:03:31.66
```

Note

The number of these log files could grow to a large number. You may want to set the version on these scheduled activities to 10 or so.

20.5.4 Notify Users When Volumes are Deallocated

To notify users when the volumes are deallocated, place the user names in the Mail attribute of the Domain object. For example:

```
$ MDMS show domain
Description: Smith's Special Domain
Mail: SYSTEM,OPERATOR1,SMITH
Offsite Location: JOHNNY_OFFSITE_TAPE_STORAGE
Onsite Location: OFFICE_65
Def. Media Type: TLZ09M
Deallocate State: TRANSITION
Opcom Class: TAPES
Request ID: 496778
Protection: S:RW,O:RW,G:R,W
DB Server Node: DEBBY
DB Server Date: 26-APR-1999 14:20:08
Max Scratch Time: NONE
Scratch Time: 365 00:00:00
Transition Time:    1 00:00:00
Network Timeout: NONE
$
```

In the above example, users SYSTEM, OPERATOR1, and SMITH will receive VMS mail when any volumes are deallocated during scheduled activities or when some one issues the following command:

```
$ MDMS DEALLOCATE VOLUME /SCHEDULE/VOLSET
```

If you delete all users in the Mail attribute, nobody will receive mail when volumes are deallocated by the scheduled activities or the DEALLOCATE VOLUME /SCHEDULE command.

21

MDMS High Level Tasks

MDMS GUI users have access to features that guide them through complex tasks. These features conduct a dialog with users, asking them about their particular configuration and needs, and then provide the appropriate object screens with information about setting specific attribute values.

The features support tasks that accomplish the following:

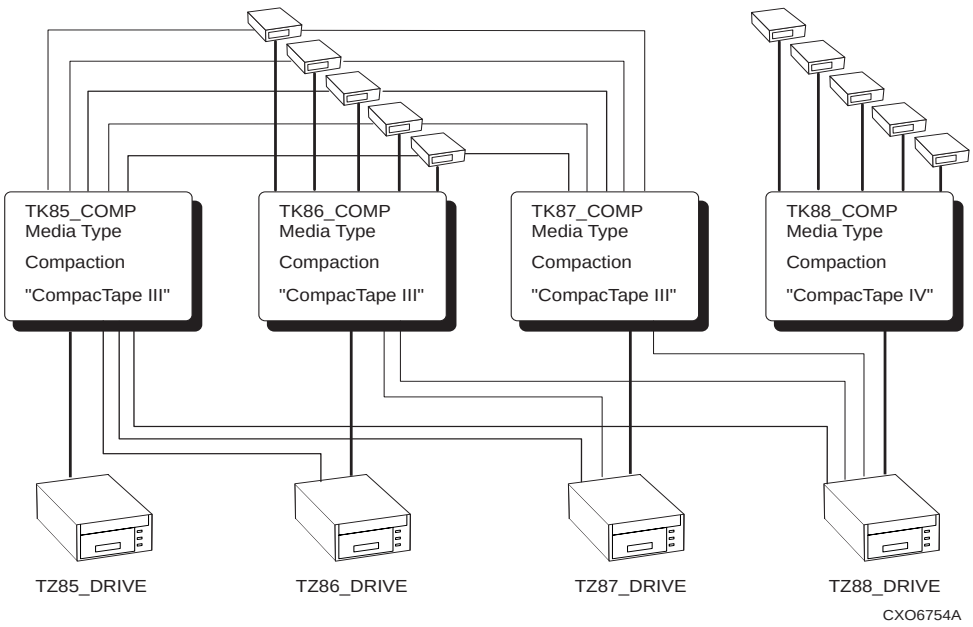
- Configuring a new drive or jukebox and/or add new volumes for management.
- Removing drives or jukeboxes and/or deleting volumes from management.
- Servicing a jukebox when it is necessary to remove allocated volumes and replace them with scratch volumes.
- Rotating volumes from the onsite location to an offsite location, and back.

The procedures outlined in this section include command examples with recommended qualifier settings shown. If you choose to perform these tasks with the command line interface, use the MDMS command reference for complete command details.

21.1 Creating Jukeboxes, Drives, and Volumes

This task offers the complete set of steps for configuring a drive or jukebox to an MDMS domain and adding new volumes used by those drives. This task can be performed to configure a new drive or jukebox that can use managed volumes.

Figure 21–1 Configuring Volumes and Drives



This task can also be performed to add new volumes into management that can use managed drives and jukeboxes.

Table 21–1 Creating Devices and Volumes

Step	Action
Create Jukebox and/or Drive	
1.	Verify that the drive is on-line and available. `$SHOW DEVICE` *`device_name`* `/FULL` Verify that the jukebox is online and available. `$SHOW DEVICE` *`device_name`* `/FULL`
2.	If you are connecting the jukebox or drive to a set of nodes which do not already share access to a common device, then create a group object record. `$MDMS CREATE GROUP` *`group_name`* `/NODES=(node_1,...)`
3.	If you are configuring a new jukebox into management, then create a jukebox object record. `$MDMS CREATE JUKEBOX` *`jukebox_name`* `/DISABLED`
4.	If the drive you are configuring uses a new type of volume, then create a media type object record. `$MDMS CREATE MEDIA_TYPE` *`media_type`*
5.	If you need to identify a new place for volume storage near the drive, then create a location object record. `$MDMS CREATE LOCATION` *`location_name`*
6.	Create the drive object record for the drive you are configuring into MDMS management. `$MDMS CREATE DRIVE` *`drive_name`* `/DISABLED`

7.	Enable the drive (and if you just added a jukebox, enable it too).
	
	`$MDMS SET DRIVE` *`drive_name`* `/ENABLED`
	`$MDMS SET JUKEBOX` *`jukebox_name`* `/ENABLED`

8.	If you are adding new volumes into MDMS management, then continue with Step 10.
9.	If you have added a new media type to complement a new type of drive, and you plan to use managed volumes, set the volumes to use the new media type. `$MDMS SET VOLUME /MEDIA_TYPE=`*`media_type_name`*
Process New Volumes	
10.	Make sure all new volumes have labels.

Table 21–1 Creating Devices and Volumes

Step	Action
11.	If the volumes you are processing are of a type you do not presently manage, complete the actions in this step. Otherwise, continue with Step 12. Create a media type object record. `$MDMS CREATE MEDIA_TYPE` *`media_type`* If the drives you manage do not accept the new media type, then set the drives to accept volumes of the new media type. `$MDMS SET DRIVE /MEDIA_TYPE=`*`media_type`*
12.	If you are using a jukebox with a vision system to create volume object records, then continue with Step 13. Otherwise, continue with Step 16 to create volume records.
Jukebox Inventory to Create Volume Object Records	

13.	If you use magazines in your operation, then continue with this step. Otherwise, continue with Step 14.
	If you do not have a managed magazine that is compatible with the jukebox, then create a magazine object record.
	`$MDMS CREATE MAGAZINE` *`magazine_name`*
	Place the volumes in the magazine.
	Move the magazine into the jukebox.
	`$MDMS MOVE MAGAZINE magazine_name jukebox_name /START_SLOT=n`
	`or`
	`$MDMS MOVE MAGAZINE magazine_name jukebox_name/START_SLOT=(n,n,n)`

14.	Place the volumes in the jukebox. If you are not using all the slots in the jukebox, note the slots you are using for this operation. Inventory the jukebox, or just the slots that contain the new volumes. If you are processing pre-initialized volumes, use the /PREINITIALIZED qualifier, then your volumes are ready for use. `$MDMS INVENTORY JUKEBOX` *`jukebox_name`* `/CREATE /VOLUME_RANGE=`*`range`*
15.	Initialize the volumes in the jukebox if they were not created as preinitialized. `$MDMS INITIALIZE VOLUME /JUKEBOX=`*`jukebox_name`* `/SLOTS=`*`range`* After you initialize volumes, you are done with this procedure.
Create Volume Object Records Explicitly	
16.	Create volume object records for the volumes you are going to manage. If you are processing preinitialized volumes, use the /PREINITIALIZED qualifier, then your volumes are ready for use. `$MDMS CREATE VOLUME` *`volume_id`*
17.	Initialize the volumes. This operation will direct the operator when to load and unload the volumes from the drive. `$MDMS INITIALIZE VOLUME` *`volume_range`* `/ASSIST`

21.2 Deleting Jukeboxes, Drives, and Volumes

This task describes the complete set of decisions and actions you could take in the case of removing a drive from management. That is, when you have to remove the last drives of a particular kind, and take with it all associated volumes, then update any remaining MDMS object records that reference the object records you delete. Any other task of removing just a drive (one of many to remain) or removing and discarding volumes involves a subset of the activities described in this procedure.

Table 21–2 Deleting Devices and Volumes

Step	Action
1.	If there is a volume in the drive you are about to remove from management, then unload the volume from the drive. `$MDMS UNLOAD DRIVE` *`drive_name`*
2.	Delete the drive from management. `$MDMS DELETE DRIVE` *`drive_name`*

3.	If you have media type object records to service only the drive you just deleted, then complete the actions in this step. Otherwise, continue with Step 4.
	
	Delete the media type object record.
	
	`$MDMS DELETE MEDIA TYPE` *`media_type`*
	
	If volumes remaining in management reference the media type, then set the volume attribute value for those volumes to reference a different media type value. Use the following command for uninitialized volumes:
	
	`$MDMS SET VOLUME /MEDIA_TYPE=`*`media_type`* `/REMOVE`
	
	Use the following command for initialized volumes:
	
	`$MDMS SET VOLUME /MEDIA TYPE=media_type`

4.	If the drives you have deleted belonged to a jukebox, then complete the actions in this step. Otherwise, continue with Step 5. If the jukebox still contains volumes, move the volumes (or magazines, if you manage the jukebox with magazines) from the jukebox to a location that you plan to keep under MDMS management. `$MDMS MOVE VOLUME` *`volume_id location`* or `$MDMS MOVE MAGAZINE` *`magazine_name location`*
5.	If a particular location served the drives or jukebox, and you no longer have a need to manage it, then delete the location. `$MDMS DELETE LOCATION` *`location_name`*
6.	Move all volumes, the records of which you are going to delete, to a managed location. `$MDMS MOVE VOLUME` *`volume_id location`*

Table 21–2 Deleting Devices and Volumes

Step	Action
7.	If the volumes to be deleted exclusively use a particular media type, and that media type has a record in the MDMS database, then take the actions in this step. Otherwise, continue with Step 8. Delete the media type object record. `$MDMS DELETE MEDIA_TYPE` *`media_type`* If drives remaining under MDMS management reference the media type you just deleted, then update the drives' media type list accordingly. `$MDMS SET DRIVE /MEDIA_TYPE` *`media_type`* `/REMOVE`
8.	If the volumes to be deleted are the only volumes to belong to a volume pool, and there is no longer a need for the pool, then delete the volume pool. `$MDMS DELETE POOL` *`pool_name`*
9.	If the volumes to be deleted exclusively used certain managed magazines, then delete the magazines. `$MDMS DELETE MAGAZINE` *`magazine_name`*
10.	Delete the volumes. `$MDMS DELETE VOLUME` *`volume_id`*

21.3 Rotating Volumes Between Sites

This procedure describes how to gather and rotate volumes from the onsite location to an offsite location. Use this procedure in accordance with your data center site rotation schedule to move backup copies of data (or data destined for archival) to an offsite location. Additionally, this procedure processes volumes from the offsite location into the onsite location.

Figure 21–2 Volume Rotation

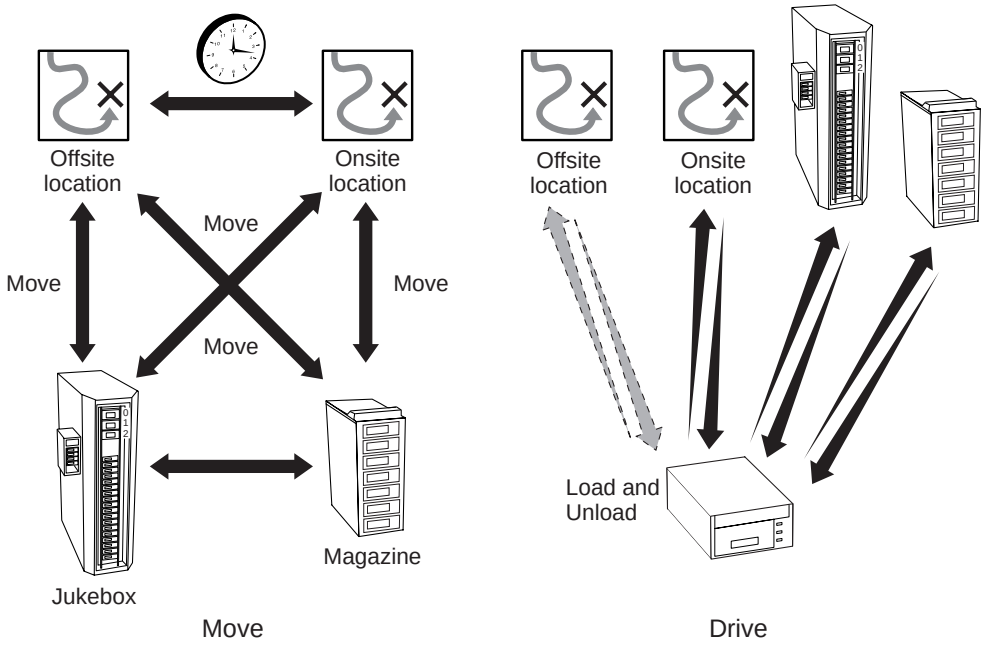


Table 21–3 Rotating Volumes Between Sites

Step	Action

1.	Prepare a report listing the offsite volumes or magazines due for rotation to your onsite location.
	`$MDMS REPORT VOLUME/SCHEDULE=ONSITE`
	or,
	`$MDMS SHOW MAGAZINE/SCHEDULE=ONSITE`
	Provide this information to the people responsible for shuttling volumes and magazines.

2.	Identify the volumes and/or magazines to move offsite.
	`$MDMS SHOW VOLUME /SCHEDULE=OFFSITE`
	or,
	`$MDMS SHOW MAGAZINE /SCHEDULE=OFFSITE`

3.	Gather the volumes into your location. If you have to retrieve magazines and/or volumes from a jukebox, then move those volumes and/or magazines out of the jukebox. Move them to an onsite location from which they will be shipped offsite. `$MDMS MOVE VOLUME /SCHEDULE=OFFSITE` *`location`* or, `$MDMS MOVE MAGAZINE /SCHEDULE=OFFSITE` *`location`*

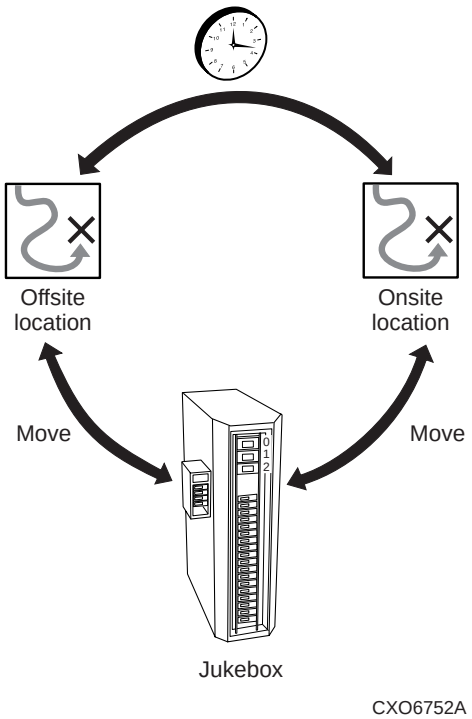
Table 21–3 Rotating Volumes Between Sites

Step	Action
4.	As the volumes are picked up for transportation, or when otherwise convenient, update the volume and/or magazine records in the database. Specify the offsite location name in this command. `$MDMS MOVE VOLUME /SCHEDULE=OFFSITE` *location* or, `$MDMS MOVE MAGAZINE /SCHEDULE=OFFSITE` *location*
5.	With MDMS, move the volumes and/or magazines to the onsite location. `$MDMS MOVE VOLUME /SCHEDULE=ONSITE` *location* or, `$MDMS MOVE MAGAZINE /SCHEDULE=ONSITE` *location*
6.	Prepare spaces for the incoming volumes and magazines. This can be accomplished by moving volumes and magazines into jukeboxes, or placing them in other locations to support operations.

21.4 Servicing Jukeboxes Used for Backup Operations

This procedure describes the steps you take to move allocated volumes from a jukebox and replace them with scratch volumes. This procedure is aimed at supporting backup operations, not operations that involve the use of managed media for hierarchical storage management.

Figure 21–3 Magazine Placement



Note

This procedure supports backup operations. Do not remove volumes allocated to HSM unless a response to a load request can be tolerated when moving the volume to the jukebox.

Table 21–4 Servicing Jukeboxes

Step	Action
1.	Report on the volumes to remove from the jukebox. `$MDMS REPORT VOLUME ALLOCATED /USER=ABS`
2.	If you manage the jukebox on a volume basis, perform this step with each volume, otherwise proceed with Step 3 with instructions for magazine management. `$MDMS MOVE VOLUME` *`volume_id location`*
3.	Identify the magazines to which the volumes belong, then move the magazines from the jukebox. `$MDMS SHOW VOLUME /MAGAZINE` *`volume_id`* then `$MDMS MOVE MAGAZINE` *`magazine_name location_name`*
4.	If you manage the jukebox on a volume basis, perform this step, otherwise proceed with Step 5 for magazine management. `$MDMS MOVE MAGAZINE` *`magazine_name location`*
5.	Move free volumes to the magazine, and move the magazine to the jukebox. `$MDMS MOVE VOLUME` *`volume_id magazine_name`* then `$MDMS MOVE MAGAZINE` *`magazine_name jukebox_name`*

A

Preparing For Disaster Recovery

In the event of a disaster, it is essential to know how to get your system up and running as quickly as possible. So that you are prepared for a disaster situation, this section contains the following information:

- The most efficient way to configure your system to ease the recovery process
- The best method of backing up those systems to ensure quick recovery
- The steps involved to actually recover the devices that have been affected by the disaster

Note

The information about this recovery process is intended only for OpenVMS VAX or Alpha systems. If you are recovering other types of systems, see the platform specific documentation for recovery information.

A.1 Efficiently Configuring Your System

To ease the recovery process, you should configure your system as recommended in *Archive Backup System for OpenVMS Installation Guide.*

The information in Section A.2 describes the preparations you need to make to be prepared for a disaster situation.

A.2 Preparing for Disaster Recovery

Table A–1 describes the tasks you must perform to make sure you are prepared in the event of a disaster situation.

Table A–1 Disaster Recovery Tasks

Step	Action

1.	Create a save request named DISASTER_SAVE for the nodes and disks (other than the system disks or the disks that contain ABS software) that contain any ABS catalogs. This is necessary only if you have moved any ABS catalogs to other disks.
	
	This save request must:
	
	• Be a full save request
	• Be done on a daily basis
	• Use the storage policy named DISASTER_RECOVERY (this storage policy must use the catalog named DISASTER_RECOVERY_CATALOG)
	
	Example:
	`$ ABS SAVE/NAME=DISASTER_SAVE`
	`_$ ABS$DISK2:ABS$CATALOG:*.*;*,ABS$DISK3:ABS$CATALOG:*.*;*`
	`_$ /FULL/INTERVAL=DAILY/START=00:00:00/STORAGE_CLASS=DISASTER_RECOVERY`
	
	Requirement:
	For each node that has ABS installed, be sure that node has a catalog named DISASTER_RECOVERY_CATALOG located in ABS$CATALOG directory.
	
	Note:
	The distinction between a typical daily system save request and this save request is that this save request only saves ABS catalogs on each disk. This save request is intended for disaster recovery only.

Table A–1 Disaster Recovery Tasks

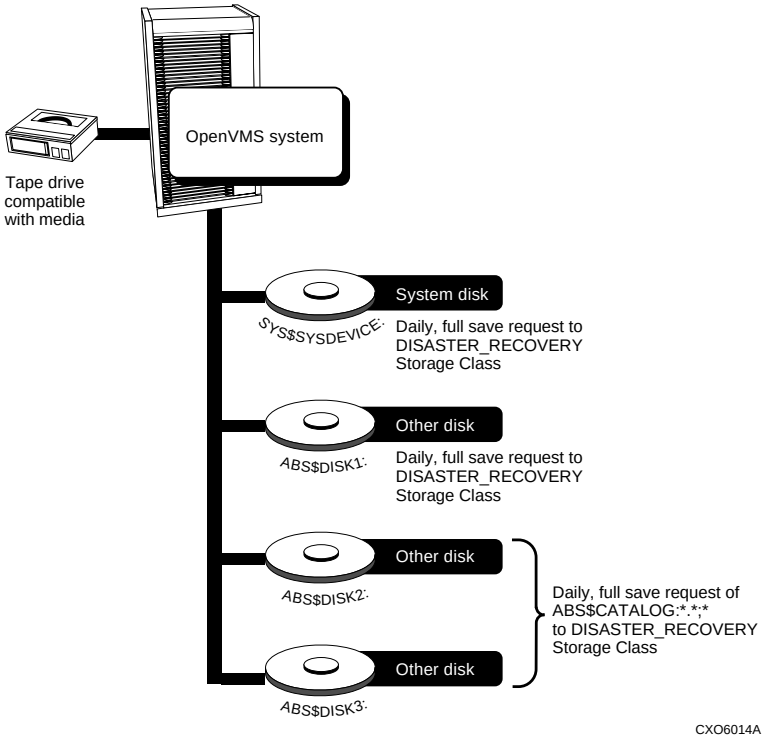
Step	Action

2.	Create a save request named SYSTEM_DISASTER_SAVE that specifies the following disks:
	• The system disk
	• The disk that contains ABS software
	• The disk that contains MDMS software
	• The disk that contains other products required by ABS, such as your 3rd party scheduler product and Motif.
	Note:
	It is recommend that all of these components reside on one disk in the system.
	This save request must:
	• Be one save request that specifies multiple disks (each of the previously described disks)
	• Be a full save request
	• Be done on a daily basis
	• Use the storage policy named DISASTER_RECOVERY (this storage policy must use the catalog named DISASTER_RECOVERY_CATALOG)
	Example:
	`$ ABS SAVE/NAME=SYSTEM_DISASTER_SAVE SYS$SYSDEVICE:,ABS$DISK1: -`
	`_$/FULL/INTERVAL=DAILY/START=00:00:00/STORAGE_CLASS=DISASTER_RECOVERY`
	Result:
	This save request backs up the system disk and the disk that contains ABS software, MDMS software, and any dependent products.

3.	Produce a report using ABS REPORT SAVE_LOG command, using the /FULL qualifier: `$ ABS REPORT SAVE_LOG/REQUEST=SYSTEM_DISASTER_SAVE/FULL` Make a note of the following fields: • Save set name • Save set position • Volume name list • Backup agent (such as VMS or RMU) • Disk or file names **Note**: Be sure to manually enter the date you generated the report on the report itself. This enables you to quickly locate the most recent report.
4.	Place the reports in a safe location. Choose a location that is protected against fire, floods, and so forth. You will need to retrieve the reports in the case of a disaster situation.

Figure A–1 shows an illustrated view of the special save requests that you need to create to ensure quick recovery from a disaster situation.

Figure A–1 Special Save Request



A.3 Recovering ABS From A Disaster Situation

To recover ABS, follow the procedure in Table A–2.

Table A–2 Recovering ABS

Step	Action
1.	Get the report stored in safekeeping.
2.	Determine which save set is the most current.
3.	Mount the volume that contains the save set. `$ MOUNT tape_device: volume_name`
4.	Boot standalone backup. Refer to Open VMS System Manager's Manual for detailed information about performing a standalone backup operation.

Table A–2 Recovering ABS

Step	Action

5.	Issue the restore command:
	`$ BACKUP tape_device: input_specifier -`
	`_$ output_specifier/LABEL=(volume_name,,,)`
	Where:
	• *tape_device* is the name of the drive that holds the volume that contains the save set.
	• *input_specifier* is the name of the save set.
	• *output_specifier* is the name of the system disk that you are restoring.
	• /*LABEL=(volume_name,...)* is the name of the volume or volumes that contain the save set. Specify the volumes in the order listed in the report.

6.	Perform a minimum boot backup operation on the system disk. See *OpenVMS System Manager's Manual* for detailed information about performing a minimum boot backup operation. • If you have additional disks to restore, go to Step 7. • If you have any incrementals that you need to apply, go to Step 7a. • Otherwise, go to Step 8.
7.	Repeat Steps 1 through 6 (except Step 4) for ABS disk, MDMS disk, and any other disks that contain dependent layered products. Instead of Step 4, mount the output device by issuing the following command: `$ MOUNT/FOREIGN output_device`

7a.	If you need to restore any incremental backup operations (you should not have to do this step if you created the save requests as previously described), follow these steps:
	Step Action
	1. Mount the volumes from the most recent to the least recent, determined by the report.
	2. Dismount the restore disk:
	`$ DISMOUNT output_device`
	3. Mount the restore disk Files-11:
	`$ MOUNT output_device volume_name`
	4. Mount the tape drive foreign:
	`$ MOUNT/FOREIGN tape_device:`
	5. Skip to the position on the save set where the incremental backup resides (supplied in report):
	`$ SET MAGTAPE/SKIP=files:n tape_device`
	6. Issue the backup command using the /INCREMENTAL qualifier:
	`$ BACKUP tape_device: save_set_name output_device/INCREMENTAL`

Table A–2 Recovering ABS

Step	Action
8.	Perform a system shutdown and startup on the disk using OpenVMS shutdown and startup command procedures. See *OpenVMS System Manager's Manual* for information about shutdown and startup procedures.
9.	Use ABS to restore other ABS catalogs that have been affected.
10.	Restore any remaining disks using ABS.

A.4 Recovering ABS Client Nodes

To recover any OpenVMS client nodes, all ABS client nodes must have access to a tape device or disk device that is compatible with the volume that contains the save set for that node. The recovery procedure is same as described in Section A.3.

B

ABS Time Formats

For scheduling purposes, Archive Backup System for OpenVMS (ABS) uses OpenVMS date/time formats. The following sections describe the date/time formats used by ABS.

B.1 Start Time Format

When creating a save or restore request, you can specify the exact time that you want the request to begin. Table B–1 lists the valid formats that you can use and defines each format listed.

Table B–1 Start Time Formats

Valid Entry Format	Definition

dd-mmm-yyyy hh:mmm:ss	Specifies the absolute starting time. If you omit the year (yyyy), the default is the current calendar year. If you omit all or part of the time (hh:mmm:ss), the omitted portion is set to 0 (zero) by default.
	
	Restriction:
	You can enter the date or time without quotation marks. If you want to enter the date and time, the syntax must be enclosed within quotation marks.
	
	Examples:
	`1. 01-JAN-1997`
	
	`2. 12:00:00`
	
	`3. ”01-JAN-1997 12:00:00”`

+dddd hh:mmm:ss	Specifies a delta time, based on the current date and time. You must specify the days (+dddd) even if you use 0 (zero). **Example**: `”+3 4:45”` **Result**: Starts the request 3 days, 4 hours, and 45 minutes from the present time.
TOMORROW+hh:mm:ss	Specifies to start the request tomorrow plus the number of hours and minutes. **Example**: `”TOMORROW+4:45”` **Result**: Starts the request tomorrow at 04:45 a.m.

B.2 Explicit Interval

When creating an ABS save request, you can specify an explicit interval at which to repeat the save request. Modifying the explicit interval option does not change the next scheduled start time for the save request. If you are using the INT_QUEUE_MANAGER or EXT_QUEUE_MANAGER scheduler interface option this interval is ignored. For EXT_SCHEDULER and DECSCHEDULER please refer to the description of interval specification for the 3rd party scheduler product being used.

The explicit interval is passed as a string to the scheduler interface being used. It is not used for the default scheduler interface option INT_QUEUE_MANAGER.

C

ABS Cleanup Utilities

Archive Backup System for OpenVMS (ABS) provides the following cleanup utilities:

- A Database Cleanup Utility - This cleanup utility removes one-time-only save and restore requests from ABS policy database that are no longer necessary.
- A Catalog Cleanup Utility - This cleanup utility periodically searches ABS catalogs and removes any expired entries from those catalogs.

This information presented in this appendix describes how to start, stop, and change the default behavior of ABS cleanup utilities.

C.1 Database Cleanup Utility

ABS provides the Database Cleanup Utility so that one-time-only save or restore requests that have completed and are not scheduled to run again will be removed from ABS policy database. Review the following descriptions to understand the criteria that the Database Cleanup Utility uses:

- **Removing save requests**

  The Database Cleanup Utility removes any save request records from the ABS database after 72 hours that were scheduled to run one time only and have completed. It also removes the save request's corresponding DECscheduler entry from the scheduler database.

- **Removing restore requests**

  ABS immediately executes a restore request upon creation. If the restore request executed successfully, the database cleanup utility removes the restore request record from ABS database after 72 hours.

C.1.1 Starting Up the Database Cleanup Utility

The Database Cleanup Utility is started when you run the file ABS$STARTUP.COM. It runs as an OpenVMS batch job and resubmits itself every night at midnight. You can also start the database cleanup utility by executing command procedure ABS$SYSTEM:ABS$START_DB_CLEANUP.COM anytime.

C.1.2 Changing the Database Cleanup Utility Default Behavior

If you wish to change the start time, modify the start_time symbol in command procedure ABS$SYSTEM:ABS$START_DB_CLEANUP.COM.

The default behavior of the Database Cleanup Utility is to remove any successfully executed one-time-only job records from ABS policy database after 72 hours. However, if you wish to change the cleanup delay, modify the symbol CleanupDelay in the command procedure ABS$SYSTEM:ABS$START_DB_CLEANUP.COM.

C.1.3 Database Cleanup Utility Log File

ABS database cleanup utility creates a new log file named ABS$LOG:ABS_CLEAN_DB_UTIL.LOG each time the cleanup utility is started. This log file contains the information about the records that are removed and any associated error messages.

Recommendation:
For maintenance purposes, periodically check this log file and purge the older versions.

C.1.4 Shutting Down the Database Cleanup Utility

To shut down the database cleanup utility job and to manually prevent the future scheduling, use one of the methods in Table C–1.

Table C–1 Shutting Down the Database Cleanup Utility

Method	Action
Delete the Queue Manager job entry	`$ SHOW ENTRY ABS_CLEAN_DB_UTIL /USER=ABS` `$ DELETE/ENTRY=<number>`

Note

When you shut down ABS software using the shutdown command procedure SYS$MANAGER:ABS$SHUTDOWN.COM,database cleanup utility is automatically deleted.

C.2 Catalog Cleanup Utility

ABS provides a Catalog Cleanup Utility that removes the references to expired data objects from ABS catalog. This feature helps maintain the size of the catalogs. The Catalog Cleanup Utility searches ABS catalogs for any entries that have expired on or before yesterday's date, and then it removes those entries from the catalogs.

C.2.1 Starting Up the Catalog Cleanup Utility

ABS creates an OpenVMS batch job for the Catalog Cleanup Utility on each node that is running ABS. If multiple nodes access and use the same catalogs, you may want to change the default behavior of the Catalog Cleanup Utility. Multiple nodes may access the same catalog if the following conditions are present:

- ABS is installed in an OpenVMS Cluster with a common disk.
- ABS is set up so that all catalogs from different installations reside in the same directory location.

The catalog cleanup utility is started automatically in SYS$STARTUP:ABS$STARTUP.COM. Command procedure ABS$SYSTEM:ABS$START_CATALOG_CLEANUP.COM is called in the startup procedure to create an OpenVMS batch job which is scheduled to run at noon. The job resubmitts itself for noon every day.

C.2.2 Changing the Catalog Cleanup Utility Default Behavior

If you wish to change the start time, modify the SubmitTime symbol in command procedure ABS$SYSTEM: ABS$START_CATALOG_CLEANUP.COM. The next time the catalog cleanup utility will resubmit itself with the new start time.

C.2.3 Catalog Cleanup Utility Log File

The Catalog Cleanup Utility creates the following log files:

- Each time the Catalog Cleanup Utility batch job starts, a new log file is created. This log file is called ABS$LOG:ABS_CATALOG_ CLEANUP.LOG. This file contains all informational and debug level information.
- ABS also creates a process log file named ABS$LOG:ABS_CLEAN_CATLG_ node.LOG. This file contains process information.

  To record all expired data object entries found in the ABS catalog, define the system logical name ABS_CATALOG_CLEANUP_DEBUG as described in Table C–2. Note that log file may become large if there is a lot of expired entries.

Table C–2 Defining the Catalog Cleanup Utility Logical Names

Logical Name	When to Define the Logical Name
ABS_CATALOG_CLEANUP	ABS automatically defines this logical name in the file ABS$STARTUP.COM. If you have performed a shutdown of the Catalog Cleanup Utility (described in Section C.2.5), you must redefine this logical name to be able to restart the Catalog Cleanup Utility: `$ DEFINE/SYSTEM ABS_CATALOG_CLEANUP 1`
ABS_CATALOG_CLEANUP_DEBUG	Define this logical name if it is important to have this information for debugging or historical purposes. Enter the following command from the system prompt: `$ DEFINE/SYSTEM ABS_CATALOG_CLEANUP_DEBUG 1` Do not define this logical name if this information is not important, or if disk space is an issue.

The ABS catalog cleanup utility creates a new log file named ABS$LOG:ABS_CLEAN_CATLG_<node_name>.LOG each time the cleanup utility is started. This log file contains the information about the records that are removed and any associated error messages.

Recommendation:
For maintenance purposes, periodically check this log file and purge the older versions

C.2.4 Shutting Down the Catalog Cleanup Utility

The file SYS$MANAGER:ABS$SHUTDOWN.COM will shutdown the Catalog Cleanup Utility. This is the recommended method of shutting down ABS and any of the utilities that ABS invokes.

If you need to shut down the catalog cleanup utility without shutting down the rest of the ABS software, deassign ABS_CATALOG_CLEANUP logical name:

```
$ SHOW LOG/FULL ABS_CATALOG_CLEANUP
“ABS_CATALOG_CLEANUP” [exec] = “1” (LNM$SYSTEM_TABLE)

$ DEASSIGN/SYSTEM/EXECUTIVE ABS_CATALOG_CLEANUP

$ SHOW LOG/FULL ABS_CATALOG_CLEANUP
%SHOW-S-NOTRAN, no translation for logical name ABS_CATALOG_CLEANUP
```

Once this logical name is deassigned, ABS performs an orderly shut down of the Catalog Cleanup Utility. Running SYS$MANAGER:ABS$SHUTDOWN.COM deletes the Catalog Cleanup Utility batch job.

Note

Do not use any other method to shut down the Catalog Cleanup Utility. Using another method could leave a catalog in an inconsistent or unusable state.

C.2.4.1 Restarting the Catalog Cleanup Utility

If you have previously shutdown the Catalog Cleanup Utility by deassigning the logical name ABS_CATALOG_CLEANUP, you must redefine the logical in order for the Catalog Cleanup Utility to perform a cleanup operation. Failing to do so would allow the Catalog Cleanup Utility to continue to run, but the cleanup operation would not take place.

```
$ DEFINE/SYSTEM ABS_CATALOG_CLEANUP 1
```

C.2.5 ABS Catalog Cleanup Utility Process

There are two pieces that are required for the Catalog Cleanup Utility to run correctly:

1. A OpenVMS batch job. This job runs each day to perform catalog cleanups and is invoked by SYS$STARTUP:ABS$STARTUP.COM. This is started either by the system's automatic startup files or by a user if the Catalog Cleanup Utility has been manually shut down.
2. The logical name ABS_CATALOG_CLEANUP. How this logical is defined determines whether the currently running Catalog Cleanup Utility job will continue or will shut down.

   If a Catalog Cleanup Utility batch executes, the first thing it does is check if ABS_CATALOG_CLEANUP logical is defined.

   If this logical is not defined (the logical has been deassigned as described in Section C.2.4), the Catalog Cleanup Utility batch job resubmits itself without performing any cleanup.

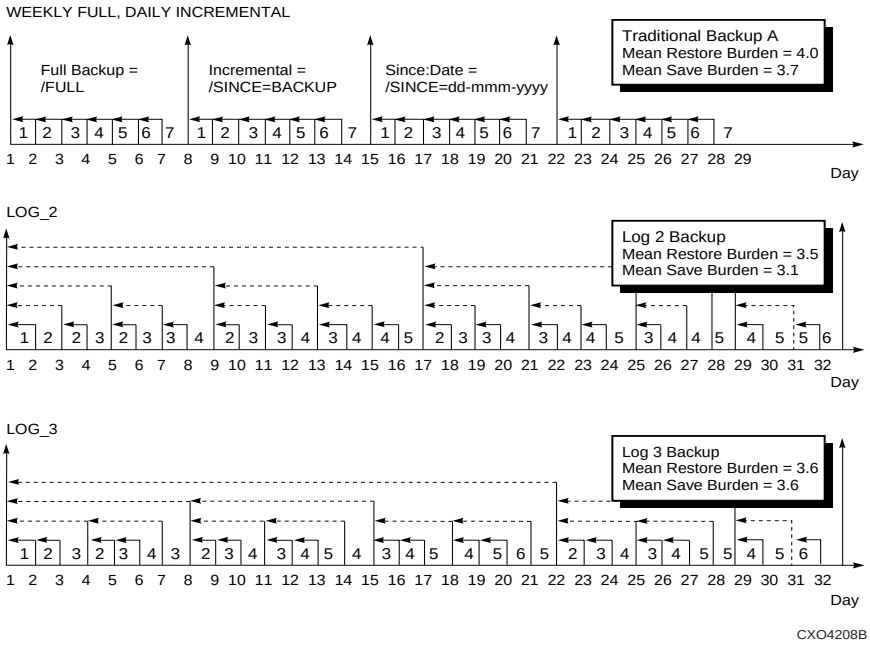
   To make sure that the cleanup operations resume, be sure to redefine the logical.

D

Log–*n* Backup Schedules

Archive Backup System for OpenVMS (ABS) provides a set of backup schedules that ease the burden of restore operations. These schedules are illustrated in Figure D–1.

Figure D–1 Log–*n* Backup Schedules



E

ABS Worksheets

The worksheets in this appendix are provided to assist you in planning and creating ABS policies. Use the worksheets as scratch areas and keep them on file for future reference to maintain ABS software.

The following worksheets are designed to be used along with ABS graphical user interface (GUI).

E.1 Storage Policy Worksheet

The worksheet provided in Table E–1 is designed to help you configure your ABS storage policies. To reuse this worksheet, make a copy of the worksheet and record your entries on the copy.

Reference:
For detailed information about storage policies, refer to Chapter 7, *Creating Storage Policies*.

Table E–1 Storage Policy Worksheet

GUI Options	Entry
Storage Policy Name	_____
Save Data To	
Disk Options	_____
Tape Options	_____
MDMS Options	
Media Type	_____
Pool	_____
Drives	_____
Location	_____
Criteria Under Which ABS Creates New Volume Sets	
Days before Creating a New Volume Set	_____
Save Sets Per Volume Set	_____
Volumes Per Volume Set	_____
Retain Data For	
Days	_____
Expire On	_____

Table E–1 Storage Policy Worksheet

GUI Options	Entry
Catalog and Execution Node	
Write History Information To (Catalog Name)	_____
Execute Save Operation on (Node Name)	_____
Number of Streams	
Maximum Number of Simultaneous Save Operations	_____
Access Control	
Owner	_____
Node	_____
User Name	_____
Rights	_____

E.2 Environment Policy Worksheet

The worksheet provided in Table E–2 is designed to help you configure your environment policies. To reuse this worksheet, make a copy of the worksheet and record your configuration on the copy.

Reference:
For detailed information about environment policies, refer to Chapter 8, *Creating Environment Policies*.

Table E–2 Environment Policy Worksheet

GUI Options	Entry
Environment Policy Name	_____
Save and Restore Environment Options	
How and Who To Notify	_____
When To Notify	_____
Type of Notification	_____
Data Verification	_____
Listing Options	_____
Preprocessing Command	_____
Postprocessing Command	_____
Original File Option	_____
Retry Options	_____
User Profile (ABS, Requester, Specific User)	_____
Open Files	_____

Table E–2 Environment Policy Worksheet

GUI Options	Entry
Number of Tape Drives	_____
Compression (UNIX only)	_____
Links Option (UNIX only)	_____
Span File Systems (UNIX only)	_____
Access Control	
Owner	_____
Node	_____
User Name	_____
Rights	_____

E.3 Save Request Worksheet

The worksheet provided in Table E–3 is designed to help you configure your save requests. To reuse the worksheet, make a copies of the worksheets and record your configuration on the copies.

Reference:
For detailed information about save requests, refer to Chapter 9, *Creating Save Requests*.

Table E–3 Save Request Worksheet

GUI Options	Entry
Save Request Name	_____
What Data To Save	
Type of Data	_____
Node Where Data Resides	_____
Disk or File name (or names)	_____
File name to Exclude	_____
Preprocessing Command	_____
Postprocessing Command	_____
Agent Qualifiers	_____
When to Save Data	
Start Time	_____
Scheduling Option	_____
Where and How	
Storage Policy to Use	_____
Environment Policy to Use	_____

Table E–3 Save Request Worksheet

GUI Options	Entry
Access Control	
Owner	_____
Node	_____
User Name	_____
Rights	_____

F

Troubleshooting

F.1 Logical Names Provide Additional Tracing

Two logical names have been defined to provide additional module tracing information. Typically, these logical names will not be useful to the customer, but they may assist ABS engineering team in tracking problems.

The following logical names are provided in ABS:

Logical Name	Description
POLICY_ABS_DEBUG_FLAG	Enables tracing for the server policy engine only.
ABS_DEBUG_FLAG	Enables tracing for all ABS components. When this logical name is set to "TRUE", it will provide information about the modules executed by the policy engine (client and server).

Note

Do not set these logical names unless specifically told to do so by ABS engineering. These logicals will produce a potentially large log file named ABS_DEBUG_TRACE.LOG.

F.2 Troubleshooting Assistance for NT Clients

Should you encounter problems when saving or restoring data using ABS for an NT client system, ABS provides way to help you troubleshoot the problem. Assign a system variable on the NT client system that, in turn, creates log files about the NT client system during ABS backup operations. These log files will assist you during the troubleshooting process.

Note

Assign this system variable only when you need troubleshooting assistance. Deassign the system variable when it is no longer needed. Do not leave the system variable assigned during normal, day-to-day operations. Because the log files can become extremely large, leaving the system variable assigned could cause performance problems.

To assign the system variable, use the procedure in Table F–1.

Table F–1 Assigning a System Variable for NT Troubleshooting

Step	Action
1.	Log into the administrator account on the NT client node.
2.	Bring up the registry editor (for example, regedt32 from command line)
3.	Go to the window for the following location: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\ - ABSClient\Parameters`
4.	From the EDIT menu select “Add Value...”, enter ABSGtarLog as the value name.
5.	Select the data type as REG_DWORD.
6.	Enter a one (1) as the data in the DWORD window; select decimal.
7.	Click OK, exit from the registry.
8.	Run your save and restore requests as usual. **Result**: The log files generated during the save or restore operation will be located in the system directory. For example, on a NT Version 4.0 server system, the directory system name would be: `c:\Winnt40S\system32` The log files are named as follows: • abs_log_file.txt - This log file contains information about the execution of the file absgtar.exe. • absclient_log_file.txt - This log file contains information about the execution of the file absclient.exe.
9.	When the log files are no longer needed, go to the same registry window and delete the entry. Do this by highlighting the entry; select Delete from the EDIT menu.

F.3 Verifying NT and UNIX Client Quotas

If you are supporting NT or UNIX clients, to ensure successful save and restore operations, set the quotas to the following values on ABS OpenVMS server node:

```
UCX> SET PROTOCOL TCP /QUOTA=(SEND:50000,RECEIVE:50000)
```

Note

If you have to reboot the machine, make sure that you reset these values after rebooting.

F.4 Considerations for Saving Large Disks on UNIX and NT Clients

ABS stores data on tape based on ANSI Standard X3.27-1987, File Structure and Labeling of Magnetic Tapes for Information Exchange. This standard requires that the block length (number of bytes per block for a file) be stored in the header section and the block count (number of blocks in a file) be stored in the end of file section. Together these fields determine the maximum number of bytes that the file contains on tape. So, in theory the following formula is implemented:

```
block length * block count = number of bytes
```

These fields on tape are stored in an ASCII format with the block length being five digits, and the block count being six digits. This allows for a maximum save request disk size of 99999 * 999999 = 99,998,900,001bytes (approximately 99 gigabytes (GB)).

ABS uses a default block length of 10240 bytes/block when it stores data to tape. As a result, the maximum disk size by default is 10240* 999999 = 10,239,989,760 (approximately 10 GB). If the actual number of bytes exceeds this amount, then ABS$UBS will raise the following assertion and the save request will fail:

```
assert error: expression = section_block_count <= 999999
```

The value of the block length is specified to the underlying gtar backup engine as a blocking factor. The blocking factor is defined as a multiple of 512 bytes. The default block length passed to gtar is “-b20”. To determine an appropriate blocking factor or block length for a specific situation, follow these steps:

Step 1. Divide the size of the disk (in bytes) by 99999

Step 2. Divide the resulting number by 512

Step 3. Round up to the next whole number

For example, if the disk size is approximately 30,000,000,000 bytes (30 GB), use the following formula:

```
30,000,000,000 / 999999 / 512 =  58.59 or 59
```

This results in a blocking factor of “-b59”.

You can modify the default block length from the GUI for an NT or UNIX save or restore request on the Agent Qualifiers window (see Section or Section). Specify this value in the Agent Qualifiers window.

Restriction:
ABS will not produce the correct results if the value exceeds “-b127”. If the disk is large enough to exceed this amount, create more than one save request for that particular disk.

To modify the blocking factor, use the procedure described in Table F–2.

Table F–2 Modifying the Blocking Factor

Step	Action
1.	Invoke ABS GUI as described in Chapter *6*
2.	Click Modify or Delete Requests & Policies
3.	Click Save Request
4.	Double-Click the save request name
5.	Double-Click on the file name in the Data To Save area
6.	Click Agent Qualifiers
7.	Click Backup Agent-Specific Qualifiers and enter the value of -b*nn* Where *nn* is a numeric value which when multiplied by 512 bytes gives the block length

Restore requirement:
When restoring data from a save request where the blocking factor has been modified, you must

specify the same blocking factor that was specified on the save request. Otherwise, the restore request will fail due to an invalid block size on the tape. As a default, ABS uses 10240.

F.5 Using The Same Volume Set For Multiple Types of ABS Clients

If you want to back up multiple types of ABS clients to the same volume set, use the same storage policy for those save requests. A single storage policy always uses the same volume set, whether the data comes from an OpenVMS, UNIX, or NT client.

To save data from multiple types of ABS clients to the same volume set, create save requests similar to the following examples:

```
$ ABS SAVE/NAME=VMS_SYSTEM/STORAGE=SYSTEM_BACKUPS -
_$/START=00:00 DISK$USER1: /OBJECT_TYPE=VMS_FILES

$ ABS SAVE/NAME-UNIX_SYSTEM/STORAGE=SYSTEM_BACKUPS -
_$/START=01:00 “/abs” /OBJECT_TYPE=UNIX_FILES_GTAR/SOURCE=unix1

$ ABS SAVE/NAME=NT_SYSTEM/STORAGE=SYSTEM_BACKUPS -
_$/START=02:00 “c:\” /OBJECT_TYPE=WINDOWS_NT_FILES_GTAR/SOURCE=nt1
```

By using the same storage policy, each save request uses the same volume set.

F.6 ABS Log Files

ABS generates a set of log files that contains information about ABS operations. Two of the log files shown in are policy engine log files. These log files contain information about ABS transactions and remain open to record ABS transactions. The third log file is a save or restore request log file that is generated when you execute a save or restore request using either the GUI or DCL.

Table F–3 describes the log file location, log file name, and contents of the specific log file.

Table F–3 ABS Log Files

Location	Log File Name	Contents
ABS$LOG:	ABS$POLICY_<node_name>[a].LOG	Audit information about ABS operations that includes a sequence number and an ABS command.
ABS$LOG:	<request_name>[b].LOG	Log information about the execution of a save or restore request. A new log file is created each time a save or restore request is executed.
ABS$LOG:	ABS$POLICY_ENGINE_<node_name>.LOG	Error information from ABS policy engine.

a. The node name where ABS policy server is executing.
b. The name of the save or restore request

Recommendation:
Do not delete these log files. They contain important information that will assist ABS Engineering with any troubleshooting process. However, if the log files are consuming too much disk space, you can delete previous versions of the log files and keep only the most recent log files:

```
$ PURGE ABS$LOG:filename.log/KEEP=5 ! keeps the latest five versions
$ DELETE ABS$LOG:filename.log/BEFORE=date ! deletes any version before the
specified date
```

F.7 New Logical Name Added To Increase Stack Size On Alpha Systems

A new logical name, ABS$COORD_ALPHA_STACKSIZE, has been added to ABS that can be used to increase the stack size on Alpha systems. ABS now sets the default stack size to 65536, (8 * 8192 = 65536). This corrected several ACCVIO and CMA-F-EXCCOP errors, especially at the end of a tape.

F.8 Additional Error Messages

Additional error messages have been added and may show up in ABS$LOG:ABS$POLICY_ENGINE_<node_name>.LOG and ABS$LOG:ABS$POLICY_<node_name>.LOG files. These messages have been added primarily to aid engineering and the support organizations with diagnosing problems.

F.9 Upgrading ABS

After upgrading ABS, it may appear that ABS has incorrectly recreated ABS database. This could be caused if your OpenVMS system was rebooted and ABS was not restarted before starting the upgrade installation procedure. This may cause ABS logicals not to be defined. The upgrade procedure requires certain logicals to be present on the OpenVMS system.

If you are upgrading ABS, make sure ABS has been running before you start the upgrade installation procedure. Follow these steps:

Step 1. Start up ABS using `@SYS$STARTUP:ABS$STARTUP.COM`

Step 2. Shut down ABS using `@SYS$MANAGER:ABS$SHUTDOWN.COM`

Step 3. Upgrade ABS using `@SYS$UPDATE:VMSINSTAL`

F.10 Logical To Assist with Server Connection Problems

If you receive the following error during a save or restore request ABS_NET_CONN_ACCEPT_FAILED, network accept connection request failed you may set a logical name to help eliminate the problem.

```
$ DEFINE/SYSTEM ABS$MAX_IO_ACCESS_WAIT n
```

Where n is the number of 5 second increments for the server to wait for connections to be established. The default value is 4.

F.11 AUDIT Flags in ABS$POLICY_CONFIG.DAT

There are AUDIT flags in the ABS$SYSTEM:ABS$POLICY_ENGINE_CONFIG.DAT file which enable additional tracing for ABS (for example, ABS$AUDIT_SHOW_EXEC_ENV).

Do NOT change the values of these flags unless specifically told to by ABS support or engineering. These symbols will produce a potentially large amount of data to the ABS policy engine log files, or create additional log files.

F.12 Troubleshooting MDMS Related Problems

If you see delays in allocating, loading, mounting or dismounting volumes for use by ABS, it is helpful to enable an OPCOM TAPE operator on your terminal window. MDMS sends helpful OPCOM message to the TAPE operator when it is having difficulty executing the request.

To enable OPCOM, type:

```
$ REPLY/ENABLE = TAPES
```

To disable OPCOM, type:

```
$ REPLY/DISABLE
```

F.13 Information Required When Reporting Problems

If you report a problem to your COMPAQ support organization, the following information should be included.

- If the problem is related to a save request:
  - ABS SHOW SAVE/FULL save_policy
  - ABS SHOW STORAGE/FULL storage_policy
  - ABS SHOW ENVIRONMENT/FULL environment_policy
  - The log file of the save request
- If the problem is related to a restore request:
  - ABS SHOW RESTORE/FULL restore_policy
  - ABS SHOW STORAGE/FULL storage_policy
  - ABS SHOW ENVIRONMENT/FULL environment_policy
  - The log file of the restore request
  - The log file for a corresponding save request which saved the data
  - ABS LOOKUP of the data being requested in the restore request
- If the problem is related to the policy engine process, or other ABS process (cleanup processes, etc):
  - The log for the process, for example the ABS$LOG:ABS$POLICY_ENGINE_<nodename>.LOG file and the ABS$LOG:ABS$POLICY_<nodename>.LOG file would be required for a policy engine problem.
- If the problem is related to MDMS:
  - MDMS SHOW output of the related volumes, drives, etc
  - Output from OPCOM messages issued by MDMS
  - Pertinent information from the MDMS server log
- Other information may be required, but will be addressed as needed.

G

ABS Error Messages

This appendix presents Archive Backup System for OpenVMS (ABS) error messages and provides descriptions and User Actions for each.

ABS_ACCESS_DENIED,

Explanation: The requested access is denied. The user does not have the proper access controls set to be granted access to the object.

User Action: Ask the owner of the object or the storage administrator to grant access to the object.

ABS_ACTIONNOSENDER,

Explanation: The coordinator received a service request with no data mover designated as the sender.

User Action: ABS internal error. Submit an SPR.

ABS_AFS_DISMOUNT_UNSUPPORTED,

Explanation: An attempt to dismount a volume failed because the archive File system associated with the storage policy does not support dismount operations. The backup agent information erroneously interpreted an output message as entering the DISMOUNT state.

User Action: ABS internal error. Submit an SPR.

ABS_AFS_EXTEND_UNSUPPORTED,

Explanation: An attempt to extend a save operation failed because the archive file system associated with the storage policy does not support extend operations. The backup agent information erroneously interpreted an output message as entering the NEW_VOLUME state.

User Action: ABS internal error. Submit an SPR.

ABS_AGENT_ABORTED,

Explanation: The backup agent moving data was aborted.

User Action: Check the log file for more specific information associated with the error, correct the indicated error, and retry the save or restore request.

ABS_AGENT_CONTEXT_EXITED,

Explanation: The backup subprocess was stopped externally to ABS and the scheduler in use.

User Action: Retry the data movement operation.

ABS_AGENT_LOOKUP_FAILED,

Explanation: An explicitly named backup agent does not exist, or the backup agent information used during a save operation was deleted.

User Action: Specify an existing backup agent.

ABS_AOE_INSNC_NOT_FOUND,

Explanation: An archive object entry instance was not found.

User Action: ABS internal error. Submit an SPR.

ABS_AOE_LIST_NULL,

Explanation: An ABS internal archive object entry list is corrupt.

User Action: ABS internal error. Submit an SPR.

ABS_AOE_NOT_FOUND,

Explanation: An archive object entry was not found in the catalog.

User Action: Use a wildcard specification in the catalog reporting facility to verify the archive object name and specify the correct data object name. If using the wildcard specification does not find the correct data object, no valid save operations of the object have been performed.

ABS_AOE_ON_LIST,

Explanation: The archive object entry is already on ABS internal AOE return list.

User Action: ABS internal error. Submit an SPR.

ABS_AOE_SHOW_CONTEXT,

Explanation: AOE Show Context is not NULL

User Action: ABS internal error. Submit an SPR.

ABS_AOE_VALIDATE_FAIL,

Explanation: Common AOE information specified does not match the catalog.

User Action: ABS internal error. Submit an SPR.

ABS_API_EXCEPTION,

Explanation: An unexpected exception occurred in the API.

User Action: If you are using the GUI, submit an SPR. If you are using the API, validate the parameters you specified to the API. If error still occurs, submit an SPR.

ABS_ARCH_ACCESS_FAILED,

Explanation: An attempt to access archive resources such as tape drives, tape pools, or disk directories failed.

User Action: Check the transaction log file, repair the error, and retry the operation. Note that you may need to check the MDMS log files for access failures to MDMS archive resources.

ABS_ARCH_CLASS_NOT_FOUND,

Explanation: The specified storage policy was not found in ABS policy database.

User Action: Use a wildcard show operation to determine a valid storage policy name and specify the correct storage policy name.

ABS_ARCH_CLOSE_FAILED,

Explanation: An attempt to release archive resources such as tape drives, media sets or disk directories failed.

User Action: Check the transaction log file, repair the error, and retry the operation. Note that you may need to check the MDMS log files for deaccess failures to MDMS archive resources.

ABS_ARCH_EXTEND_FAILED,

Explanation: An attempt to extend a media set failed.

User Action: Check the transaction log file, repair the error, and retry the operation. Note that you may need to check the MDMS log files for deaccess failures to MDMS archive resources.

ABS_ARCH_REQ_NOT_FOUND,

Explanation: A save request matching the specified name, name and version, or UID was not found. Specify a valid restore request name, name and version, or UID.

User Action: Use a wildcard show operation to determine a valid save request name; specify the correct save request name.

ABS_ARCH_TRANS_NOT_FOUND,

Explanation: The specified archive transaction was not found in ABS policy database.

User Action: ABS internal error. Submit an SPR.

ABS_BADSTATE,

Explanation: The backup agent state machine entered a bad state.

User Action: ABS internal error. Submit an SPR.

ABS_BAD_PLATFORM_ERROR,

Explanation: An attempt to translate a platform-specific error failed.

User Action: ABS internal error. Submit an SPR.

ABS_BAD_TYPE_FOR_AGENT,

Explanation: The specified backup agent does not support the specified movement type.

User Action: The backup agent information is incorrect. Submit an SPR.

ABS_BAD_XN_ARCH_CLASS,

Explanation: The transaction added to a session specified a storage policy other than the session's storage policy.

User Action: ABS internal error. Submit an SPR.

ABS_CLOSE_CATALOG_FAILURE,

Explanation: An attempt to close the catalog failed.

User Action: ABS internal error. Submit an SPR.

ABS_CMDERR,

Explanation: There is a syntax error on ABS command line

User Action: Correct the syntax and retry the command.

ABS_CONTINUING,

Explanation: The transaction coordinator encountered an error, but is retrying the operation.

User Action: None.

ABS_COORD_EXCEPTION,

Explanation: An unexpected exception occurred in the coordinator.

User Action: ABS internal error. Submit an SPR.

ABS_COORD_NOUID,

Explanation: No transaction UID was found on the coordinator command line.

User Action: Specify a valid transaction UID on the coordinator command line.

ABS_CREATEEERR,

Explanation: An attempt to create an ABS environment policy failed.

User Action: Additional information should follow this error. Evaluate the additional information to determine the problem. Correct the problem and retry the command.

ABS_CREATESERR,

Explanation: An attempt to create an ABS storage policy has failed.

User Action: Additional information should follow this error. Evaluate the additional information to determine the problem. Correct the problem and retry the command.

ABS_CREPRCERR,

Explanation: An attempt to create a subprocess to house the backup agent failed.

User Action: Check the transaction log file, correct the indicated error, and retry the operation.

ABS_DATA_MOVER_EXCEPTION,

Explanation: An unexpected exception occurred in the data mover.

User Action: ABS internal error. Submit an SPR.

ABS_DB_ATTACH_ERROR,

Explanation: ABS policy engine's attempt to access ABS policy database failed.

User Action: Check to see if ABS account can access to ABS policy database that is pointed to by the logical ABSDATABASE. If this is true, ABS policy database is corrupt. Restore ABS policy database.

ABS_DELERR,

Explanation: An attempt to delete a record from ABS database failed.

User Action: Additional information should follow the error message. Evaluate the additional information to determine the specific problem with ABS policy database. Correct the problem and retry the command.

ABS_DB_DELETE_FAILURE,

Explanation: An attempt to delete a record from persistent store failed.

User Action: Additional information should follow this error message. Evaluate the additional information to determine the specific problem with ABS policy database. Correct the problem and retry the operation.

ABS_DB_INSERT_FAILURE,

Explanation: An attempt to insert a record from persistent store failed.

User Action: Additional information should follow this error message. Evaluate the additional information to determine the specific problem with ABS policy database. Correct the problem and retry the operation.

ABS_DB_SELECT_FAILURE,

Explanation: An attempt to select a record from persistent store failed.

User Action: Additional information should follow this error message. Evaluate the additional information to determine the specific problem with ABS policy database. Correct the problem and retry the operation.

ABS_DB_TRANS_FAILURE,

Explanation: A transaction could not be started on ABS policy database.

User Action: Additional information should follow this error message. Evaluate the additional information to determine the specific problem with ABS policy database. Correct the problem and retry the operation.

ABS_DB_UPDATE_FAILURE,

Explanation: An attempt to update a record from persistent store failed.

User Action: Additional information should follow this error message. Evaluate the additional information to determine the specific problem with ABS policy database. Correct the problem and retry the operation.

ABS_DELPRCERR,

Explanation: An attempt to delete a backup agent subprocess failed.

User Action: Check the transaction log file for more information and correct any errors. The transaction should have completed.

ABS_DIRECTORY_CREATE_ERROR,

Explanation: An attempt to create an on-disk directory for a Files-11 storage policy failed.

User Action: Make sure the primary archive Location specified has a valid Files-11 format. If it does, check the transaction log file for more specific information, correct the indicated error, and retry the operation.

ABS_DRIVE_RVN_MISMATCH,

Explanation: ABS found an unexpected relative volume number (RVN) mounted on a drive.

User Action: ABS internal error. Submit an SPR.

ABS_DRIVE_RVN_NOT_FOUND,

Explanation: ABS could not find a drive with the specified relative volume number (RVN) mounted on it.

User Action: ABS internal error. Submit an SPR.

ABS_DUPLICATE_ARCH_CLASS,

Explanation: An attempt to create an storage policy failed because the storage policy already exists.

User Action: Specify a different storage policy name, or modify the existing storage policy to meet your needs.

ABS_DUPLICATE_ARCH_REQ,

Explanation: An attempt to create a save request failed because the save request already exists.

User Action: Specify a different save request name, or modify the existing save request to meet your needs.

ABS_DUPLICATE_EXECUTION_ENV,

Explanation: An attempt to create a request environment policy failed because the request environment policy already exists.

User Action: Specify a different request environment policy name, or modify the existing request environment policy to meet your needs.

ABS_DUPLICATE_REST_REQ,

Explanation: An attempt to create a restore request failed because the restore request already exists.

User Action: Specify a different restore request name, or modify the existing restore request to meet your needs.

ABS_EE_NODE_MISMATCH

Explanation: The USER request can not be executed on the current node. The Cluster name/Node name specified in the Execution Environment User Profile does not match the current Cluster name/Node name. If the Execution Environment specifies "*" for the User Profile Node, the node where the save request was created is the permitted node.

User Action: Modify the execution environment to allow the execution of the request on the desired node.

ABS_ELEMENT_NOT_FOUND,

Explanation: The specified element was not found in the list.

User Action: ABS internal error. Submit an SPR.

ABS_EXEC_ENVIR_NOT_FOUND,

Explanation: The specified environment policy was not found in ABS policy database.

User Action: Use a wildcard show operation to determine a valid request environment policy name. Specify the valid request environment policy name.

ABS_EXPIRED,

Explanation: The check for media set consolidation interval failed.

User Action: ABS internal error. Submit an SPR.

ABS_EXTEND_SUBSTATE_FINISHED,

Explanation: One step in the archive extend operation has completed.

User Action: ABS internal error. Submit an SPR.

ABS_FAILURE,

Explanation: A failure occurred.

User Action: ABS internal error. Submit an SPR.

ABS_FATAL_ERROR_DETECTED,

Explanation: A fatal error was detected in the data mover.

User Action: Additional information should follow this error message. Evaluate the additional information to determine the specific problem with ABS policy database. Correct the problem and retry the operation.

ABS_FIND_HELD_FAILED,

Explanation: SYS$FIND_HELD service failed.

User Action: ABS internal error. Submit an SPR.

ABS_GET_JPI_FAILED,

Explanation: SYS$GETJPI(W) call failed.

User Action: ABS internal error. Submit an SPR.

ABS_GET_NODENAME_FAIL,

Explanation: An attempt to get cluster node names failed.

User Action: ABS internal error. Submit an SPR.

ABS_GET_PID_FAILED,

Explanation: An attempt to get the current process ID failed.

User Action: ABS internal error. Submit an SPR.

ABS_GET_UAI_FAILED,

Explanation: SYS$GETUAI service failed.

User Action: ABS internal error. Submit an SPR.

ABS_GET_UIC_FAILED, Failed to get current process UIC

Explanation: An attempt to get the current UIC failed.

User Action: ABS internal error. Submit an SPR.

ABS_IDTOASC_FAILED,

Explanation: SYS$IDTOASC service failed.

User Action: ABS internal error. Submit an SPR.

ABS_IMAGEACT_FAILED,

Explanation: An attempt to activate ABS$USSSHR failed.

User Action: Additional information should follow this error. Evaluate the additional information, correct any indicated errors, and retry the operation.

ABS_INSUFF_ARCH_RESOURCES,

Explanation: Some archive resources such as tape drives, media sets, or disk directories were allocated, but not all requested resources could be allocated.

User Action: None. The operation continues with the limited resources.

ABS_INTERNAL_ERROR,

Explanation: An internal error occurred.

User Action: ABS internal error. Submit an SPR.

ABS_INT_TOO_BIG,

Explanation: A value being parsed from the backup agent output is too large to be contained in a 32-bit integer.

User Action: ABS internal error. Submit an SPR. The backup agent information is probably trying to parse an incorrect item.

ABS_INVLD_ACCESS,

Explanation: Invalid catalog access was specified.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_ACCESS_IDENTIFIER,

Explanation: Invalid access identifier parameter. The access identifier parameter exceeds the maximum length.

User Action: Specify a valid length parameter.

ABS_INVLD_ACCESS_LIST,

Explanation: Invalid access rights list parameter was specified.

User Action: Specify a valid access rights list parameter.

ABS_INVLD_ACCESS_RIGHT,

Explanation: Invalid access rights string was specified.

User Action: Specify a valid access rights string.

ABS_INVLD_AGENT_DATA,

Explanation: Invalid backup agent ID data was specified.

User Action: Agent_ID_Data parameter exceeds the maximum valid length. Specify a valid length Agent_ID_Data parameter.

ABS_INVLD_AGENT_FS_ROOT,

Explanation: Invalid agent file system root parameter was specified.

User Action: Correct the agent file system root parameter.

ABS_INVLD_AGENT_IND,

Explanation: Invalid backup agent indicator was specified.

User Action: Specify a valid backup agent indicator.

ABS_INVLD_AGENT_INFO,

Explanation: The specified backup agent information is invalid.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_AGENT_NAME,

Explanation: Specified backup agent name is too long.

User Action: Either the backup agent name is too long or contains invalid characters. Specify a valid backup agent name.

ABS_INVLD_AGENT_ROOT,

Explanation: The backup agent file system root is invalid.

User Action: The agent_filesystem_root is too long. Specify a shorter agent_filesystem_root.

ABS_INVLD_AGENT_SELECTION,

Explanation: Invalid backup agent selection criteria.

User Action: Specify a valid backup agent selection criteria.

ABS_INVLD_AGENT_UID_STRING,

Explanation: The UID string in the backup agent information is invalid.

User Action: If you have made changes to the backup agent information, restore the original backup agent information from ABS distribution kit. If you have not changed the backup agent information, submit an SPR.

ABS_INVLD_ARCH_ATTR,

Explanation: An invalid archive attributes structure or value was specified.

User Action: Specify a valid archive attributes structure or value.

ABS_INVLD_ARCH_CLASS,

Explanation: An invalid storage policy name was specified.

User Action: The storage policy name is either too long or contains invalid characters. Specify a valid storage policy name.

ABS_INVLD_ARCH_DATE,

Explanation: An invalid archive date format was specified.

User Action: Specify a valid archive date format.

ABS_INVLD_ARCH_INTERVAL,

Explanation: An invalid archive interval value was specified.

User Action: Specify a valid archive interval value.

ABS_INVLD_ARCH_OBJ_LOC,

Explanation: Invalid archive object location structure.

User Action: Use ABS_Set_archv_object_location routine to set up a valid archive object location.

ABS_INVLD_ARCH_REQMNTS,

Explanation: An invalid archive requirements structure was specified.

User Action: Use ABS add_archive_reqmnts utility routine to set up a correct archive requirements structure.

ABS_INVLD_ARCH_STATUS,

Explanation: Invalid object archived status.

User Action: Archive object status is too long. Specify a shorter string.

ABS_INVLD_ARCH_TIME,

Explanation: Catalog date archived is invalid.

User Action: Specify a valid date archived parameter.

ABS_INVLD_ARCH_TRANS_UID,

Explanation: The transaction UID is invalid and is specified only in internal routines.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_BACKUP_DATE,

Explanation: Invalid backup date format.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_CATALOG,

Explanation: An invalid catalog was specified.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_CATALOG_TYPE,

Explanation: Invalid catalog type.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_COLLECTION,

Explanation: A nonexistent scratch collection was specified.

User Action: Specify a valid scratch collection when you create or modify the storage policy.

ABS_INVLD_COMPOUND_OBJECT,

Explanation: An invalid compound object set structure or value was specified.

User Action: Specify a valid compound object set structure or value. Use theABS_Set_compound_object_set utility routine to set up a valid structure.

ABS_INVLD_CONNECTIONID,

Explanation: Catalog ConnectionId argument is invalid.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_CONSOLIDATION,

Explanation: An invalid consolidation interval criteria structure or value was specified.

User Action: Specify a valid consolidation interval criteria structure or value.

ABS_INVLD_CREATE_DATE,

Explanation: Invalid creation date format.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_DATA_MOVEMENT,

Explanation: An invalid movement type structure or value was specified.

User Action: Specify a valid movement type structure or value.

ABS_INVLD_DATE_FORMAT,

Explanation: Unable to format a binary date to an ASCII date format.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_DATE_MATCH,

Explanation: An invalid date match criteria was specified.

User Action: A valid date match criteria was specified.

ABS_INVLD_DEFAULT_FLAG,

Explanation: Specified an invalid default flag.

User Action: Specify a valid default flag on ABS_SET_COMPOUND_OBJECT_SET utility routine.

ABS_INVLD_DESTINATION,

Explanation: An invalid archive destination indicator structure or value was specified.

User Action: Specify a valid archive destination indicator structure or value.

ABS_INVLD_DEVICE_NAME,

Explanation: Invalid device name.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_DIAG_BLOCK,

Explanation: Missing the diagnostic block parameter.

User Action: Specify a valid diagnostic block address.

ABS_INVLD_DISK_NAME,

Explanation: An invalid disk name was specified in the include specification.

User Action: Specify a valid disk name on the include specification.

ABS_INVLD_DRIVE_LIST,

Explanation: An invalid drive name list structure or value was specified.

User Action: Specify a valid drive name list structure or value.

ABS_INVLD_DRIVE_NAME,

Explanation: An invalid drive name structure or value was specified.

User Action: Specify a valid MDMS drive name.

ABS_INVLD_EE_NAME,

Explanation: An invalid request environment policy name was specified.

User Action: The specified request environment policy name is either too long or contains invalid characters. Specify a valid request environment policy name.

ABS_INVLD_ENTITY_NAME,

Explanation: An invalid object name was specified.

User Action: Specify a valid object name.

ABS_INVLD_EPILOGUE,

Explanation: An invalid epilogue structure or value was specified.

User Action: Specify a valid epilogue structure or value.

ABS_INVLD_EXECUTION_ENV,

Explanation: Specified an invalid request environment policy name.

User Action: The specified request environment policy does not exist. Perform a wildcard show operation to find the valid request environment policy names. Specify a valid request environment policy.

ABS_INVLD_EXPIRE_DATE,

Explanation: Invalid expiration date format.

User Action: ABS internal error. Submit an SPR

ABS_INVLD_EXP_INTERVAL,

Explanation: Invalid explicit interval.

User Action: Specify a valid explicit interval.

ABS_INVLD_FILE_NAME,

Explanation: Invalid archive file system filename.

User Action: Check the archive file system filename. If it contains valid characters, submit an SPR.

ABS_INVLD_FILE_SYSTEM,

Explanation: An invalid archive file system structure or value was specified.

User Action: Specify a valid archive file system structure or value.

ABS_INVLD_FORMAT_PARAMETER,

Explanation: Invalid parameter for tag formatting.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_IDLE_RETAIN_FLAG,

Explanation: An invalid idle retain flag structure or value was specified.

User Action: Specify a idle retain flag valid structure or value.

ABS_INVLD_INST_CHAR,

Explanation: Invalid instance characteristics.

User Action: ABS internal error. Submit an SPR

ABS_INVLD_LISTING_OPTION,

Explanation: An invalid listing option was specified.

User Action: Specify a valid listing option.

ABS_INVLD_LOG_HANDLE,

Explanation: An invalid log file handle in an open log file was specified.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_LOGGING_OPTION,

Explanation: An invalid logging option was specified.

User Action: Specify a valid logging option.

ABS_INVLD_MEDIA_SET_NAME,

Explanation: A nonexistent volume set name was specified.

User Action: Specify a valid volume set name.

ABS_INVLD_METHOD,

Explanation: Invalid method parameter.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_METHOD_STRING,

Explanation: Invalid method string parameter.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_MODIF_DATE,

Explanation: Invalid revision date format.

User Action: ABS internal error. Submit an SPR

ABS_INVLD_NEWVOL_PROTOCOL,

Explanation: New volume protocol for backup agent is invalid.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_NODE_LIST,

Explanation: An invalid node list structure or value was specified.

User Action: Specify a valid node list structure or value.

ABS_INVLD_NODE_NAME,

Explanation: An invalid node name was specified.

User Action: Specify a valid node name.

ABS_INVLD_OBJECT_NAME,

Explanation: Invalid object name.

User Action: ABS internal error. Submit an SPR

ABS_INVLD_OBJECT_UID,

Explanation: An invalid UID for the object was specified.

User Action: Specify a valid object UID.

ABS_INVLD_OBJECT_VERSION,

Explanation: An object version was specified without an object name.

User Action: You must specify object name when you specify an object version.

ABS_INVLD_OPTIONS_LIST,

Explanation: An invalid options list parameter was specified.

User Action: Specify an options list parameter.

ABS_INVLD_ORIG_DISP,

Explanation: An invalid original disposition structure or value was specified.

User Action: Specify a valid original disposition structure or value.

ABS_INVLD_OUTPUT_SPEC,

Explanation: An invalid output specification pointer was specified.

User Action: Specify a valid output specification pointer.

ABS_INVLD_OWNER,

Explanation: An invalid owner name was specified.

User Action: The owner parameter is too long. Specify a valid owner.

ABS_INVLD_PARAM_MASK,

Explanation: An invalid parameter mask was specified.

User Action: Specify a valid parameter mask to set on the set call.

ABS_INVLD_PARSE_PARAMETER,

Explanation: Invalid parameter for tag parsing.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_PRIVILEGES,

Explanation: An invalid privileges parameter was specified, or the parameter exceeded the maximum length.

User Action: Specify a valid length privileges parameter.

ABS_INVLD_PROLOGUE,

Explanation: An invalid prologue structure or value was specified.

User Action: Specify a valid prologue structure or value.

ABS_INVLD_REASON,

Explanation: Invalid reason parameter.

User Action: Specify a valid notification reason.

ABS_INVLD_REPVOL_PROTOCOL,

Explanation: The backup agent information is incorrect.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_REQ_LOG_NAME,

Explanation: Invalid requestor logical name.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_REQMNTS_LIST,

Explanation: An invalid requirements list head pointer structure or value was specified.

User Action: Specify a valid requirements list head pointer structure or value.

ABS_INVLD_REQUEST_NAME,

Explanation: An invalid request name was specified.

User Action: The specified request name is either too long or contains invalid characters. Specify a valid request name.

ABS_INVLD_RESTART_INTERVAL,

Explanation: An invalid restart interval structure or value was specified.

User Action: Specify a valid restart interval structure or value.

ABS_INVLD_RESTORE_INFO,

Explanation: Invalid restore information parameter.

User Action: Specify a valid alternate restore information structure. Normally, the alternate restore information is not specified, but is retrieved from the catalog.

ABS_INVLD_RESTORE_LEVEL,

Explanation: Invalid incremental restore level.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_RETENTION_CRITERIA,

Explanation: An invalid retention criteria structure was specified.

User Action: You must specify a retention criteria parameter.

ABS_INVLD_RETENTION_CRITERIA,

Explanation: An invalid retention criteria was specified.

User Action: Specify a valid retention criteria.

ABS_INVLD_RETENT_IND,

Explanation: An invalid retention indicator was specified.

User Action: Specify a valid retention indicator.

ABS_INVLD_RETURN_BLOCK,

Explanation: Invalid return block parameter.

User Action: Specify a valid return_block address on a show call.

ABS_INVLD_ROOT_LOCATION,

Explanation: An invalid root archive location was specified.

User Action: The specified primary archive location is too long. Specify a valid primary archive location.

ABS_INVLD_ROS_NAME,

Explanation: Invalid ReferenceObjectSetName.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_ROS_UID,

Explanation: Invalid reference object set UID.

User Action: ABS internal error. Submit an SPR

ABS_INVLD_SCHEDULE_INFO,

Explanation: An invalid schedule_info structure value was specified.

User Action: Use ABS_Set_schedule_info utility routine to create a valid schedule_info structure.

ABS_INVLD_SEGMENT,

Explanation: An invalid template segment was specified.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_SELECT_CRITERIA,

Explanation: Invalid selection criteria structure.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_SERVICE_INFO,

Explanation: Invalid agent service information.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_SIMPLE_OBJECT,

Explanation: An invalid simple object set was specified.

User Action: Use ABS add_simple_object_set to add at least one simple object set to the compound object set.

ABS_INVLD_SIZE_ATTR,

Explanation: Invalid size attributes structure.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_SPECIAL_DAY,

Explanation: An invalid special day parameter was specified.

User Action: The special day parameter is too long. Specify a valid special day class name.

ABS_INVLD_SS_NAME,

Explanation: Invalid save set name.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_SS_SIZE,

Explanation: Invalid save set size format.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_SS_UID,

Explanation: Invalid save set UID.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_STAGING_OPTION,

Explanation: An invalid staging option was specified.

User Action: Specify a valid staging option.

ABS_INVLD_STARTUP_PROTOCOL,

Explanation: Backup agent startup protocol is invalid.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_START_TIME,

Explanation: The start time string is too long.

User Action: The start time string is either too long or contains invalid characters. Specify a valid start time string.

ABS_INVLD_STATUS_PROTOCOL,

Explanation: Backup agent status return protocol is invalid.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_STATUS_VALUE,

Explanation:

User Action: Specify a valid ABS status_t value to ABS GetMessageText routine.

ABS_INVLD_SYS_LOG_NAME,

Explanation: Invalid system logical name.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_TAG,

Explanation: Invalid tag.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_TAG_FORMAT,

Explanation: Invalid tag format in tag template.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_TEMPLATE,

Explanation: Invalid tag template.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_TEMPLATE_LIST,

Explanation: An invalid template list was specified.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_TEMPLATE_SEVERITY,

Explanation: Invalid severity in tag template.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_TEMPLATE_STATE,

Explanation: An invalid state was specified in tag template.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_INVLD_TIME,

Explanation: An invalid time format was specified.

User Action: Specify a valid time on the API call or from the graphical user interface (GUI).

ABS_INVLD_TRANSACTION_STATUS,

Explanation: An invalid parameter was passed to an internal ABS routine.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_USER_NAME,

Explanation: An invalid user name parameter was specified, or the user name parameter exceeds the maximum length.

User Action: Specify a valid length user name parameter.

ABS_INVLD_USER_PROFILE,

Explanation: Specified an invalid user_profile structure.

User Action: Specify a valid user profile structure.

ABS_INVLD_WAIT_FLAG,

Explanation: An invalid wait flag value was specified.

User Action: Specify either TRUE or FALSE.

ABS_INVLD_WORK_PROTOCOL,

Explanation: Agent work request protocol is invalid.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit

ABS_INVLD_XN,

Explanation: Invalid transaction UID.

User Action: The save or restore request was tried to execute with a deleted transaction. Delete the job from OpenVMS Queue Manager or scheduler database being used and, if necessary, recreate the operation.

ABS_INVLD_XN_SEVERITY,

Explanation: Transaction severity is invalid.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_XN_STATUS,

Explanation: Invalid archive transaction status.

User Action: ABS internal error. Submit an SPR.

ABS_INVLD_XN_SUMMARY,

Explanation: Invalid transaction summary string parameter.

User Action: Correct the summary string parameter.

ABS_INVLD_XN_TYPE,

Explanation: Invalid transaction type.

User Action: ABS internal error. Submit an SPR.

ABS_LIBADDTIMES_FAILED,

Explanation: LIB$ADD_TIMES service failed.

User Action: ABS internal error. Submit an SPR.

ABS_LIBEDIV_FAILED,

Explanation: LIB$EDIV service failed.

User Action: ABS internal error. Submit an SPR.

ABS_LIBSUBTIMES_FAILED,

Explanation: LIB$SUB_TIMES service failed.

User Action: ABS internal error. Submit an SPR.

ABS_LOOKUPERR,

Explanation: An attempt to execute an ABS lookup command failed.

User Action: Addition information should follow the error message. Evaluate the additional information to determine the problem. Correct the problem and retry the command.

ABS_LOST_THREAD_CONTEXT,

Explanation: Coordinator could not locate a thread.

User Action: ABS internal error. Submit an SPR.

ABS_MBXIOERR,

Explanation: Error while creating or reading mailbox.

User Action: ABS internal error. Submit an SPR.

ABS_NAME_UID_MISMATCH,

Explanation: Object name and UID mismatch.

User Action: An attempt was made to specify both the name and UID of an object. Specify either the name only, the name and version, or the UID only.

ABS_NO_ACCESS,

Explanation: Owner access to catalog denied.

User Action: Access requested to the catalog does not match the authorized access. Contact the Storage Administrator for access to the specified catalog.

ABS_NO_ACCESS_TO_COLLECTION,

Explanation: Not authorized to access scratch collection.

User Action: Contact the Storage Administrator for access to the specified scratch collection.

ABS_NO_AGENT_FOR_TYPE,

Explanation: No backup agent found to handle specified object type.

User Action: No backup agent was found to save or restore the specified object type. Use the pull-down menu on the GUI to determine valid object types. Specify one of the valid object types.

ABS_NO_AOE_LIST,

Explanation: The AOE list is corrupt.

User Action: ABS internal error. Submit an SPR.

ABS_NO_AOE_SC_MATCH,

Explanation: AOE selection criteria argument does not match show context selection criteria.

User Action: ABS internal error. Submit an SPR.

ABS_NO_AOE_SHOW_CONTEXT,

Explanation: AOE show context is NULL.

User Action: ABS internal error. Submit an SPR.

ABS_NO_CATALOG,

Explanation: A catalog name was not specified on ABS_OpenCatalog.

User Action: Specify a valid catalog name.

ABS_NO_COS_SET,

Explanation: A compound object set was not specified.

User Action: A compound object set must be specified on the utility routine ABS add_simple_object_set. Use ABS_set_compound_object_set to create a compound object set.

ABS_NO_DISMOUNT_FILES11,

Explanation: ABS cannot dismount FILES-11 disk. An attempt was made to perform a full restore operation to a mounted disk.

User Action: To be perform the full restore operation, you must dismount the disk.

ABS_NO_MORE_AOE_ENTRIES,

Explanation: No more entries match AOE selection criteria.

User Action: Do not issue another ABS_ShowObjectEntry.

ABS_NO_MORE_AGENTS,

Explanation: No more agent information is available.

User Action: ABS internal error. Submit an SPR.

ABS_NO_MORE_TLE_ENTRIES,

Explanation: No more entries match transaction log entry selection criteria.

User Action: Do not issue another ABS_ShowLogEntry.

ABS_NO_NODE,

Explanation: The specified node name was not found in the node list.

User Action: Attempted to start a catalog server on a node which is not authorized for the catalog. Start the server on an authorized node, or modify the authorized list for the catalog.

ABS_NO_PRIV,

Explanation: The user process does not have the required privileges to perform the requested function.

User Action: Consult your system manager or storage administrator.

ABS_NO_SELECT_CRITERIA,

Explanation: No selection criteria found.

User Action: Specify at least one selection criteria for catalog lookup. See Chapter 13, *Looking Up Saved Data* for the list of valid selection criteria.

ABS_NO_SUCH_TAG,

Explanation: No such tag.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_NO_TLE_LIST,

Explanation: No transaction log entry return list in transaction log entry show context.

User Action: ABS internal error. Submit an SPR.

ABS_NO_TLE_SC_MATCH,

Explanation: The transaction log entry selection criteria argument does not match show context selection criteria.

User Action: ABS internal error. Submit an SPR.

ABS_NO_TLE_SHOW_CONTEXT,

Explanation: The transaction log entry show context is NULL.

User Action: ABS internal error. Submit an SPR.

ABS_NODE_NOT_LOCAL,

Explanation: A node name that is not in the local cluster was specified.

User Action: Specify a valid node name within the local cluster.

ABS_NOLABELSUPPORT,

Explanation: Volume labels are not supported by the archive file system (AFS).

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_NOMATCH,

Explanation: Input line does not match parse data.

User Action: ABS internal error. Submit an SPR.

ABS_NOMEM,

Explanation: Virtual memory could not be allocated.

User Action: Check the page file quota for the account executing the failing job. Increase the page file quota. If the problem persists, submit an SPR.

ABS_NOMORE_PARSE,

Explanation: No more agent parse data.

User Action: ABS internal error. Submit an SPR.

ABS_NOMORE_SOS,

Explanation: No more single object sets in the transaction.

User Action: ABS internal error. Submit an SPR.

ABS_NONEXIST_ARCH_TRANS,

Explanation: Nonexistent archive transaction in the transaction chain.

User Action: ABS internal error. Submit an SPR.

ABS_NONEXISTENT_CATALOG,

Explanation: Specified an nonexistent catalog name.

User Action: A non existent catalog name was specified in a storage policy or on a restore request. Specify a valid catalog name.

ABS_NOT_SPECIAL_TAG,

Explanation: The specified tag was not a special tag.

User Action: ABS internal error. Submit an SPR.

ABS_NOT_YET_IMPLEMENTED,

Explanation: Routine not yet implemented.

User Action: If this message is returned from the GUI, submit an SPR. If this status is returned from the API, check API document for supported functions and parameters.

ABS_OBJECT_NOT_FOUND,

Explanation: The specified object was not found in ABS policy database.

User Action: ABS internal error. Submit an SPR.

ABS_OPEN_CATALOG_FAILURE,

Explanation: Failed to open the catalog.

User Action: Examine the associated error messages for specific reasons why the catalog could not opened.

ABS_OPEN_LOG_FAILURE,

Explanation: Failed to open the log file.

User Action: Check the associated error messages for specific reasons why the log file could not be opened.

ABS_OVERFLOW,

Explanation: Tag formatting overflowed the buffer.

User Action: The command formatting template in the agent information is too long to fit into the 1024 byte command buffer. Reduce the size of the command formatting template so that it fits within a 1024 byte command buffer.

ABS_OVERRUN,

Explanation: Message information overflows field.

User Action: The information parsed from the backup agent exceeds the size of the available field. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_PIPECREATEERR,

Explanation: Error creating pipe.

User Action: Check the associated error messages for specific reasons why the pipe could not be created.

ABS_PIPEDELETEERR,

Explanation: Error deleting pipe.

User Action: Check the associated error messages for specific reasons why the pipe could not be deleted.

ABS_PIPEIOERR,

Explanation: Error while reading or writing pipe.

User Action: Check the associated error messages for specific reasons why the pipe could not be read from or written to.

ABS_PLATFORM_SPECIFIC_ERROR,

Explanation: Platform-specific error in diagnostic block.

User Action: This message is provided as additional information to help diagnose a failure. The subsequent message contains VMS or other operating system-specific information.

ABS_REF_DELETE_FAILURE,

Explanation: Delete failed because the record is in use.

User Action: Wait until all references in the catalog to this object have expired, then delete the object. If desired, you can change the expiration date of the catalog entries to expire before the original expiration date.

ABS_RELEASE_NO_ARCH,

Explanation: Thread released with no archive access.

User Action: ABS internal error. Submit an SPR.

ABS_REST_REQ_NOT_FOUND,

Explanation: The restore request was not found in ABS policy database.

User Action: A restore request matching the specified name, name and version, or UID was not found. Specify a valid restore request name, name and version, or UID.

ABS_RESTOREERR,

Explanation: An attempt to execute a restore request failed.

User Action: Examine the restore log in ABS$LOG directory. Determine the reason for failure and correct the problem. Re-run the restore request.

ABS_RETRY_EXHAUSTED,

Explanation: Agent retry exhausted.

User Action: The retry count specified in the request environment policy has been exceeded. The operation will not be retried.

ABS_SAVEERR,

Explanation: An attempt to execute a save request failed.

User Action: Examine the save log in ABS$LOG directory. Determine the reason for failure and correct the problem. Re-run the save request.

ABS_SCHEDULE_ERROR,

Explanation: Failed to schedule a request job using the current scheduler interface option.

User Action: Check the associated error messages for specific reasons why the job could not be scheduled. For scheduler interface option EXT_QUEUE_MANAGER and EXT_SCHEDULER check logfiles ABS$LOG:ABS$EXT_QUEUE_MANAGER*.LOG or ABS$LOG:ABS$EXT_SCHEDULER*.LOG for more information.

ABS_SCHED_CREATE_ERROR,

Explanation: Failed to schedule a request job using the current scheduler interface option.

User Action: Check the associated error messages for specific reasons why the job could not be created. For scheduler interface option EXT_QUEUE_MANAGER and EXT_SCHEDULER check logfiles ABS$LOG:ABS$EXT_QUEUE_MANAGER*.LOG or ABS$LOG:ABS$EXT_SCHEDULER*.LOG for more information.

ABS_SCHED_DELETE_ERROR,

Explanation: Failed to delete the request job using the current scheduler interface option.

User Action: Check the associated error messages for specific reasons why the job could not be deleted. For scheduler interface option EXT_QUEUE_MANAGER and EXT_SCHEDULER check logfiles ABS$LOG:ABS$EXT_QUEUE_MANAGER*.LOG or ABS$LOG:ABS$EXT_SCHEDULER*.LOG for more information.

ABS_SCHED_ENUM_ERROR,

Explanation: Invalid enumeration in scheduler utility.

User Action: ABS internal error. Submit an SPR.

ABS_SCHED_INVLD_PARAM,

Explanation: Invalid parameter to scheduler utility.

User Action: ABS internal error. Submit an SPR.

ABS_SCHED_INVLD_TIME,

Explanation: An invalid time format was specified for start time or explicit interval.

User Action: Specify a valid OpenVMS date/time.

ABS_SCHED_MODIFY_ERROR,

Explanation: Failed to modify the request job using the current scheduler interface option.

User Action: Check the associated error messages for specific reasons why the job could not be modified. For scheduler interface option EXT_QUEUE_MANAGER and EXT_SCHEDULER check logfiles ABS$LOG:ABS$EXT_QUEUE_MANAGER*.LOG or ABS$LOG:ABS$EXT_SCHEDULER*.LOG for more information.

ABS_SCHED_NONEXIST,

Explanation: A job entry was not found for this request.

User Action: ABS internal error. Submit an SPR.

ABS_SCHED_SHOW_ERROR,

Explanation: Failed to show a request job for the current scheduler interface option.

User Action: Check the associated error messages for specific reasons why the job could not be shown. For scheduler interface option EXT_QUEUE_MANAGER and EXT_SCHEDULER check logfiles ABS$LOG:ABS$EXT_QUEUE_MANAGER*.LOG or ABS$LOG:ABS$EXT_SCHEDULER*.LOG for more information.

ABS_SESSION_IN_PROGRESS,

Explanation: Archive session is in progress.

User Action: The archive session is currently waiting for more work requests to come in. No User Action is required.

ABS_SETDEVCTX_FAILED,

Explanation: Set device context failed.

User Action: Check the associated error messages for specific reasons why the operation failed. Check ABS account to make sure it has SETPRV and CMKRNL privileges set.

ABS_SETEERR,

Explanation: An attempt to set an ABS environment policy failed.

User Action: Addition information should follow the error message. Evaluate the additional information to determine the problem. Correct the problem and retry the command.

ABS_SETRESTOREERR,

Explanation: An attempt to set an ABS restore request failed.

User Action: Addition information should follow the error message. Evaluate the additional information to determine the problem. Correct the problem and retry the command.

ABS_SETSAVEERR,

Explanation: An attempt to set an ABS save request failed.

User Action: Addition information should follow the error message. Evaluate the additional information to determine the problem. Correct the problem and retry the command.

ABS_SETSERR,

Explanation: An attempt to set an ABS storage policy failed.

User Action: Addition information should follow the error message. Evaluate the additional information to determine the problem. Correct the problem and retry the command.

ABS_SHOWERR,

Explanation: An attempt to show a database record (ABS object) failed.

User Action: Additional information should follow the error message. Evaluate the additional information to determine the problem. Correct the problem and retry the command.

ABS_SHUTERR,

Explanation: An attempt to shut down ABS Policy Engine failed.

User Action: Additional information should follow the error message. Evaluate the additional information to determine the problem. Correct the problem and retry the shutdown procedure. Also check ABS$LOG:ABS$POLICY_ENGINE_T.LOG file. Errors may be present in that log file which may help determine the problem. If the shutdown still fails, then delete ABS$POLICY process using STOP/ID=pid.

ABS_SPECIFY_AGENT_ID_DATA,

Explanation: Agent ID data argument was not specified.

User Action: ABS internal error. Submit an SPR.

ABS_SPECIFY_ARCH_OBJ_LOC,

Explanation: Archive object location structure was not specified.

User Action: To create an object entry, you must specify the archive object location.

ABS_SPECIFY_ARCH_XNUID,

Explanation: ArchiveTransactionUID was not specified.

User Action: To create a log entry, you must specify the archive transaction UID.

ABS_SPECIFY_DEVICE,

Explanation: Device name was not specified.

User Action: You must specify the device name on ABS_set_aoe_selection_criteria. Specify a wildcard to look at all devices.

ABS_SPECIFY_DEVICE_NAME,

Explanation: The device name was not specified in the correct format.

User Action: Specify either physical_disk_name, system_ logical_name, or requestor_logical_name.

ABS_SPECIFY_FILENAME,

Explanation: The archive file system file name was not specified.

User Action: The save set name (archive file system file name) must be specified in ABS_Set_saveset_info routine.

ABS_SPECIFY_OBJECT,

Explanation: Object name was not specified.

User Action: You must specify the object name to be found in the catalog in ABS_Set_aoe_selection_criteria call.

ABS_SPECIFY_OBJECT_ID,

Explanation: Object identification was not specified.

User Action: The object identification must be specified in ABS_CreateObjectEntry call.

ABS_SPECIFY_OBJECT_LOC,

Explanation: Archive object location was not specified.

User Action: You must specify the object location on ABS_CreateObjectEntry call.

ABS_SPECIFY_PATHNAME,

Explanation: Pathname was not specified.

User Action: You must specify the location of the saveset (archive file system pathname) on ABS_Set_saveset_info call.

ABS_SPECIFY_RETURN_BLOCK,

Explanation: The return block pointer argument was not specified.

User Action: The return block pointer was NULL on either ABS_ShowLogEntry or ABS_ShowObjectEntry.

ABS_SPECIFY_SELECT_CRITERIA,

Explanation: Selection criteria argument was not specified.

User Action: You must specify the selection criteria parameter on either ABS_ShowLogEntry or ABS ShowObjectEntry call.

ABS_SPECIFY_SSI,

Explanation: The save set information was not specified.

User Action: You must specify the save set information parameter in ABS_CreateLogEntry call. Use ABS_Set_saveset_info utility routine to create this structure.

ABS_SPECIFY_SS_UID,

Explanation: The save set UID was not specified.

User Action: The save set UID must be specified in ABS set_saveset_info call.

ABS_SS_UID_NOMATCH,

Explanation: The save set UID specified in object location does not match log entry.

User Action: The save set UID in the object location must match the save set UID in the current log entry for the active catalog connection. Specify the same save set UID in the object location as specified on ABS_CreateObjectEntry, or issue another ABS_CreateLogEntry.

ABS_STRING_OVERFLOW,

Explanation: String copy overflowed output buffer.

User Action: ABS internal error. Submit an SPR.

ABS_SUBPROCDELETEERR,

Explanation: Error deleting subprocess.

User Action: Check the associated error messages for specific reasons why the subprocess could not be deleted.

ABS_SUCCESS,

Explanation: Normal successful operation was completed.

User Action: None.

ABS_SYSALLOC_FAILED,

Explanation: Failed to allocate device.

User Action: Check the associated error messages for specific reasons why the device could not be allocated.

ABS_SYSASSIGN_FAILED,

Explanation: SYS$ASSIGN failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSBINTIM_FAILED,

Explanation: SYS$BINTIM service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSCLOSE_FAILED,

Explanation: SYS$CLOSE service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSCONNECT_FAILED,

Explanation: SYS$CONNECT service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSDALLOC_FAILED,

Explanation: Failed to deallocate device.

User Action: Check the associated error messages for specific reasons why the device could not be deallocated.

ABS_SYSDELETE_FAILED,

Explanation: SYS$DELETE service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSDISCONNECT_FAILED,

Explanation: SYS$DISCONNECT service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSDISMOU_FAILED,

Explanation: SYS$DISMOU service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSGETDVI_FAILED,

Explanation: SYS$GETDVI service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSGET_FAILED,

Explanation: SYS$GET service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSGET_FAILED_AOE,

Explanation: SYS$GET service failed on an archive object entry.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSGET_FAILED_TLE,

Explanation: SYS$GET service failed on a transaction log entry.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSMOUNT_FAILED,

Explanation: SYS$MOUNT service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSOPEN_FAILED,

Explanation: SYS$OPEN service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSPARSE_FAILED,

Explanation: SYS$PARSE service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSPUT_FAILED,

Explanation: SYS$PUT service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSPUT_FAILED_AOE,

Explanation: SYS$PUT service failed on an archive object entry.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSPUT_FAILED_TLE,

Explanation: SYS$PUT service failed on a transaction log entry.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSREWIND_FAILED,

Explanation: SYS$QIOW with IO$_REWIND failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSSKIPFILE_FAILED,

Explanation: SYS$QIOW with IO$_SKIPFILE failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSSNDOPR_FAILED,

Explanation: Failed to send a message to the operator.

User Action: Check the associated error messages for specific reasons why the operator did not receive a message.

ABS_SYSTEM_ERROR,

Explanation: An unexpected error occurred in the graphical user interface (GUI).

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSUPDATE_FAILED,

Explanation: SYS$UPDATE service failed.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSUPDATE_FAILED_AOE,

Explanation: SYS$UPDATE service failed on an archive object entry.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_SYSUPDATE_FAILED_TLE,

Explanation: SYS$UPDATE service failed on a transaction log entry.

User Action: Check the associated error messages for specific reasons why the operation failed.

ABS_TLE_LIST_NULL,

Explanation: ABS internal transaction log list is corrupt.

User Action: ABS internal error. Submit an SPR.

ABS_TLE_NOT_FOUND,

Explanation: A transaction log entry was not found in the catalog.

User Action: Validate the selection criteria specified on ABS_Set_tle_selection_criteria.

ABS_TLE_ON_LIST,

Explanation: The transaction log entry is already on ABS internal transaction log entry return list.

User Action: ABS internal error. Submit an SPR.

ABS_TLE_SHOW_CONTEXT,

Explanation: The transaction log entry show context is not NULL.

User Action: ABS internal error. Submit an SPR.

ABS_TOOMANYVALUES,

Explanation: Too many values found in tag template.

User Action: The backup agent information is incorrect. If you have modified the template information, restore the original template information from ABS distribution kit. If you have not modified the template information, submit an SPR.

ABS_TRANSNOTFND,

Explanation: An attempt to synchronize on an ABS save or restore request failed.

User Action: Execute an ABS SHOW of the save or restore request. If it does not exist, create it.

ABS_USER_NAME_NOT_FOUND,

Explanation: The specified user name does not exist in the current cluster.

User Action: Specify a valid user name within the current cluster.

ABS_XMTEXT_WRITE_FAIL,

Explanation: XmText write has failed, check widget ID.

User Action: ABS internal error. Submit an SPR.

ABS_XN_FAILED,

Explanation: Transaction completed with failure status.

User Action: Check the transaction log file for specific reasons for the failure.

ABS_XN_QUALIFIED_SUCCESS,

Explanation: Transaction completed with qualified success.

User Action: Check the transaction log file for specific warnings or non fatal errors.

ABS_XOFFD,

Explanation: Pipe is busy.

User Action: ABS internal error. Submit an SPR.

H

MDMS Error Messages

This Appendix presents Media and Device Management Services for OpenVMS Version 3 (MDMS) error messages and provides descriptions and User Actions for each.

ABORT request aborted by operator

Explanation: The request issued an OPCOM message that has been aborted by an operator. This message can also occur if no terminals are enabled for the relevant OPCOM classes on the node.

User Action: Either enable an OPCOM terminal, contact the operator and retry
or
no action.

ACCVIO access violation

Explanation: The MDMS software caused an access violation. This is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

ALLOCDRIVEDEV drive string allocated as device string

Explanation: The named drive was successfully allocated, and the drive may be accessed with DCL commands using the device name shown.

User Action: None.

ALLOCDRIVE drive string allocated

Explanation: The named drive was successfully allocated.

User Action: None.

ALLOCVOLUME volume string allocated

Explanation: The named volume was successfully allocated.

User Action: None.

APIBUGCHECK internal inconsistency in API

Explanation: The MDMS API (MDMS$SHR.EXE) detected an inconsistency. This is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

APIUNEXP unexpected error in API string line number

Explanation: The shareable image MDMS$SHR detected an internal inconsistency.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

BINDVOLUME volume string bound to set string

Explanation: The specified volume (or volume set) was successfully bound to the end of the named volume set.

User Action: None.

BUGCHECK, internal inconsistency

Explanation: The server software detected an inconsistency. This is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis. Restart the server.

CANCELLED, request cancelled by user

Explanation: The request was cancelled by a user issuing a cancel request command.

User Action: None, or retry command.

CONFLITEMS, conflicting item codes specified

Explanation: The command cannot be completed because there are conflicting item codes in the command. This is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis

CREATVOLUME, volume string created

Explanation: The named volume was successfully created.

User Action: None.

DBLOCACC, local access to database

Explanation: This node has the database files open locally.

User Action: None.

DBRECERR, error string record for string:

Explanation: The search for a database server received an error from a remote server.

User Action: Check the logfile on the remote server for more information. Check the logical name MDMS$DATABASE_SERVERS for correct entries of database server node.

DBREMACC, access to remote database server on node string

Explanation: This node has access to a remote database server.

User Action: None.

DBREP, Database server on node string reports:

Explanation: The remote database server has reported an error condition. The next line contains additional information.

User Action: Depends on the additional information.

DCLARGLSOVR DCL extended status format, argument list overflow

Explanation: During formatting of the extended status, the number of arguments exceeded the allowable limit.

User Action: This is an internal error. Contact Compaq.

DCLBUGCHECK internal inconsistency in DCL

Explanation: You should never see this error. There is an internal error in the DCL.

User Action: This is an internal error. Contact Compaq.

DCSCERROR error accessing jukebox with DCSC

Explanation: MDMS encountered an error when performing a jukebox operation. An accompanying message gives more detail.

User Action: Examine the accompanying message and perform corrective actions to the hardware, the volume or the database, and optionally retry the operation.

DCSCMSG string

Explanation: This is a more detailed DCSC error message which accompanies DCSCERROR.

User Action: Check the DCSC error message file.

DECNETLISEXIT, DECnet listener exited

Explanation: The DECnet listener has exited due to an internal error condition or because the user has disabled the DECNET transport for this node. The DECnet listener is the server's routine to receive requests via DECnet (Phase IV) and DECnet-Plus (Phase V).

User Action: The DECnet listener should be automatically restarted unless the DECNET transport has been disabled for this node. Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis if the transport has not been disabled by the user.

DECNETLISRUN, listening on DECnet node string object string

Explanation: The server has successfully started a DECnet listener. Requests can now be sent to the server via DECnet.

User Action: None.

DEVNAMICM device name item code missing

Explanation: During the allocation of a drive, a drive's drive name was not returned by the server. This is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

DRIVEEXISTS specified drive already exists

Explanation: The specified drive already exists and cannot be created.

User Action: Use a set command to modify the drive, or create a new drive with a different name.

DRVACCERR error accessing drive

Explanation: MDMS could not access the drive.

User Action: Verify the VMS device name, node names and/or group names specified in the drive record. Fix if necessary.

Verify MDMS is running on a remote node. Check status of the drive, correct and retry.

DRVALRALLOC drive is already allocated

Explanation: An attempt was made to allocate a drive that was already allocated.

User Action: Wait for the drive to become deallocated, or if the drive is allocated to you, use it.

DRVEMPTY drive is empty

Explanation: The specified drive is empty.

User Action: Check status of drive, correct and retry.

DRVINITERR error initializing drive on platform

Explanation: MDMS could not initialize a volume in a drive.

User Action: There was a system error initializing the volume. Check the log file.

DRVINUSE drive is currently in use

Explanation: The specified drive is already in use.

User Action: Wait for the drive to free up and re-enter command, or try to use another drive.

DRVLOADED drive is already loaded

Explanation: A drive unload appeared to succeed, but the specified volume was still detected in the drive.

User Action: Check the drive and check for duplicate volume labels, or if the volume was reloaded.

DRVLOADING drive is currently being loaded or unloaded

Explanation: The operation cannot be performed because the drive is being loaded or unloaded.

User Action: Wait for the drive to become available, or use another drive. If the drive is stuck in the loading or unloading state, check for an outstanding request on the drive and cancel it. If all else fails, manually adjust the drive state.

DRVNOTALLOC drive is not allocated

Explanation: The specified drive could not be allocated.

User Action: Check again if the drive is allocated. If it is, wait until it is deallocated. Otherwise there was some other reason the drive could not be allocated. Check the log file.

DRVNOTALLUSER drive is not allocated to user

Explanation: You cannot perform the operation on the drive because the drive is not allocated to you.

User Action: In some cases you may be able to perform the operation by specifying a user name. Do that to check if it works or defer the operation.

DRVNOTAVAIL drive is not available on system

Explanation: The specified drive was found on the system, but is not available for use.

User Action: Check the status of the drive and correct.

DRVNOTDEALLOC drive was not deallocated

Explanation: MDMS could not deallocate a drive.

User Action: Either the drive was not allocated or there was a system error deallocating the drive. Check the log file.

DRVNOTFOUND drive not found on system

Explanation: The specified drive cannot be found on the system.

User Action: Check that the OpenVMS device name, node names and/or group names are correct for the drive. Verify MDMS is running on a remote node.

Re-enter command when corrected.

DRVNOTSPEC drive not specified or allocated to volume

Explanation: When loading a volume a drive was not specified, and no drive has been allocated to the volume.

User Action: Retry the operation and specify a drive name.

DRVREMOTE drive is remote

Explanation: The specified drive is remote on a node where it is defined to be local.

User Action: Check that the OpenVMS device name, node names and/or group names are correct for the drive. Verify MDMS is running on a remote node. Re-enter command when corrected.

DRVSINUSE all drives are currently in use

Explanation: All of the drives matching the selection criteria are currently in use.

User Action: Wait for a drive to free up and re-enter command.

ERROR error

Explanation: A general MDMS error occurred.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

EXECOMFAIL execute command failed, see log file for more explanation

Explanation: While trying to execute a command during scheduled activities, a system service called failed.

User Action: Check the log file for the failure code from the system server call.

FAILALLOCDRV failed to allocate drive

Explanation: Failed to allocate drive.

User Action: The previous message is the error that caused the failure.

FAILCONSVRD, failed connection to server via DECnet

Explanation: The DECnet(Phase IV) connection to an MDMS server either failed or could not be established. See additional message lines and/or check the server's logfile.

User Action: Depends on additional information.

FAILCONSVRT, failed connection to server via TCP/IP

Explanation: The TCP/IP connection to an MDMS server either failed or could not be established. See additional message lines and/or check the server's logfile.

User Action: Depends on additional information.

FAILCONSVR, failed connection to server

Explanation: The connection to an MDMS server either failed or could not be established. See additional message lines and/or check the server's logfile.

User Action: Depends on additional information.

FAILDEALLOCDRV failed to deallocate drive

Explanation: Failed to deallocate drive.

User Action: The previous message is the error that caused the failure.

FAILEDMNTVOL failed to mount volume

Explanation: MDMS was unable to mount the volume.

User Action: The error above this contains the error that caused the volume not to be mounted.

FAILICRES failed item code restrictions

Explanation: The command cannot be completed because there are conflicting item codes in the command. This is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

FAILINIEXTSTAT failed to initialize extended status buffer

Explanation: The API could not initialize the extended status buffer. This is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

FAILURE fatal error

Explanation: The MDMS server encountered a fatal error during the processing of a request.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

FILOPNERR, file string could not be opened

Explanation: An MDMS database file could not be opened.

User Action: Check the server's logfile for more information.

FIRSTVOLUME specified volume is first in set

Explanation: The specified volume is the first volume in a volume set.

User Action: You cannot deallocate or unbind the first volume in a volume set. However, you can unbind the second volume and then deallocate the first, or unbind and deallocate the entire volume set.

FUNCFAILED, Function string failed with:

Explanation: An internal call to a system function has failed. The lines that appear after this error message identify the function called and the failure status.

User Action: Depends on information that appears following this message.

ILLEGALOP illegal move operation

Explanation: You attempted to move a volume within a DCSC jukebox, and this is not supported.

User Action: None.

INCOMPATOPT incompatible options specified

Explanation: You entered a command with incompatible options.

User Action: Examine the command documentation and re-enter with allowed combinations of options.

INCOMPATVOL volume is incompatible with volumes in set

Explanation: You cannot bind the volume to the volume set because some of the volume's attributes are incompatible with the volumes in the volume set.

User Action: Check that the new volume's media type, onsite location and offsite location are compatible with those in the volume set. Adjust attributes and retry, or use another volume with compatible attributes.

INSCMDPRIV insufficient privilege to execute request

Explanation: You do not have sufficient privileges to enter the request.

User Action: Contact your system administrator and request additional privileges, or give yourself privileges and retry.

INSOPTPRIV insufficient privilege for request option

Explanation: You do not have sufficient privileges to enter a privileged option of this request.

User Action: Contact your system administrator and request additional privileges, or give yourself privileges and retry. Alternatively, retry without using the privileged option.

INSSHOWPRIV some volumes not shown due to insufficient privilege

Explanation: Not all volumes were shown because of restricted privilege.

User Action: None if you just want to see volumes you own. You need MDMS_SHOW_ALL privilege to see all volumes.

INSSVRPRV insufficient server privileges

Explanation: The MDMS server is running with insufficient privileges to perform system functions.

User Action: Refer to the Installation Guide to determine the required privileges. Contact your system administrator to add these privileges in the MDMS$SERVER account.

INTBUFOVR, internal buffer overflow

Explanation: The MDMS software detected an internal buffer overflow. This an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis. Restart the server.

INTINVMSG, internal invalid message

Explanation: An invalid message was received by a server. This could be due to a network problem or, a remote non-MDMS process sending messages in error or, an internal error.

User Action: If the problem persists and no non-MDMS process can be identified then provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

INVABSTIME invalid absolute time

Explanation: The item list contained an invalid absolute date and time. Time cannot be earlier than 1-Jan-1970 00: 00: 00 and cannot be greater than 7-Feb-2106 06: 28: 15

User Action: Check that the time is between these two times.

INVALIDRANGE invalid volume range specified

Explanation: The volume range specified is invalid.

User Action: A volume range may contain up to 1000 volumes, where the first 3 characters must be alphabetic and the last 3 may be alphanumeric. Only the numeric portions may vary in the range. Examples are ABC000-ABC999, or ABCD01-ABCD99.

INVDBSVRLIS, invalid database server search list

Explanation: The logical name MDMS$DATABASE_SERVERS contains invalid network node names or is not defined.

User Action: Correct the node name(s) in the logical name MDMS$DATABASE_SERVERS in file MDMS$SYSTARTUP.COM. Redefine the logical name in the current system. Then start the server.

INVDELSTATE object is in invalid state for delete

Explanation: The specified object cannot be deleted because its state indicates it is being used.

User Action: Defer deletion until the object is no longer being used, or otherwise change its state and retry.

INVDELTATIME invalid delta time

Explanation: The item list contained an invalid delta time.

User Action: Check that the item list has a correct delta time.

INVDFULLNAM, invalid DECnet full name

Explanation: A node full name for a DECnet-Plus (Phase V) node specification has an invalid syntax.

User Action: Correct the node name and retry.

INVEXTSTS invalid extended status item desc/buffer

Explanation: The error cannot be reported in the extended status item descriptor. This error can be cause by one of the following: Not being able to read any one of the item descriptors in the item list

Not being able to write to the buffer in the extended status item descriptor

Not being able to write to the return length in the extended status item descriptor

Not being able to initialize the extended status buffer

User Action: Check for any of the above errors in your program and fix the error.

INVITCODE invalid item code for this function

Explanation: The item list had an invalid item code. The problem could be one of the following: Item codes do not meet the restrictions for that function.

An item code cannot be used in this function.

User Action: Refer to the API specification to find out which item codes are restricted for each function and which item codes are allowed for each function.

INVITDESC invalid item descriptor, index number

Explanation: The item descriptor is in error. The previous message gives the error. Included is the index of the item descriptor in the item list.

User Action: Refer to the index number and the previous message to indicate the error and which item descriptor is in error.

INVITLILENGTH invalid item list buffer length

Explanation: The item list buffer length is zero. The item list buffer length cannot be zero for any item code.

User Action: Refer to the API specification to find an item code that would be used in place of an item code that has a zero buffer length.

INVMEDIATYPE media type is invalid or not supported by volume

Explanation: The specified volume supports multiple media types where a single media type is required, or the volume does not support the specified media type.

User Action: Re-enter the command specifying a single media type that is already supported by the volume.

INVMSG, invalid message via string

Explanation: An invalid message was received MDMS software. This could be due to a network problem or, a non-MDMS process sending messages in error or, an internal error.

User Action: If the problem persists and no non-MDMS process can be identified then provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

INVNODNAM, invalid node name specification

Explanation: A node name for a DECnet (Phase IV) node specification has an invalid syntax.

User Action: Correct the node name and retry.

INVPORTS, invalid port number specification

Explanation: The MDMS server did not start up because the logical name MDMS$TCPIP_SND_PORTS in file MDMS$SYSTARTUP.COM specifies and illegal port number range. A legal port number range is of the form "low_port_number-high_port_number".

User Action: Correct the port number range for the logical name DMS$TCPIP_SND_PORTS in file MDMS$SYSTARTUP.COM. Then start the server.

INVPOSITION invalid jukebox position

Explanation: The position specified is invalid.

User Action: Position is only valid for jukeboxes with a topology defined. Check that the position is within the topology ranges, correct and retry. Example: /POSITION=(1,2,1)

INVSELECT invalid selection criteria

Explanation: The selection criteria specified on an allocate command are invalid.

User Action: Check the command with the documentation and re-enter with a valid combination of selection criteria.

INVSLOTRANGE invalid slot range

Explanation: The slot range was invalid. It must be of the form: 1-100 1,100-200,300-400. The only characters allowed are:
, (comma), - (dash), and numbers (0-9).

User Action: Check that you are using the correct form.

INVSRCDEST invalid source or destination for move

Explanation: Either the source or destination of a move operation was invalid (does not exist).

User Action: If the destination is invalid, enter a correct destination and retry. If a source is invalid, either create the source or correct the current placement of the affected volumes or magazines.

INVTFULLNAM, invalid TCP/IP full name

Explanation: A node full name for a TCP/IP node specification has an invalid syntax.

User Action: Correct the node name and retry.

INVTOPOLOGY invalid jukebox topology

Explanation: The specified topology for a jukebox is invalid.

User Action: Check topology definition; the towers must be sequentially increasing from 0; there must be a face, level and slot definition for each tower.

Example:

/TOPOLOGY=(TOWER=(0,1,2), FACES=(8,8,8), - LEVELS=(2,3,2), SLOTS=(13,13,13))

INVVOLPLACE invalid volume placement for operation

Explanation: The volume has an invalid placement for a load operation.

User Action: Re-enter the command and use the move option.

INVVOLSTATE volume in invalid state for operation

Explanation: The operation cannot be performed on the volume because the volume state does not allow it.

User Action: Defer the operation until the volume changes state. If the volume is stuck in a transient state (e.g. moving), check for an outstanding request and cancel it. If all else fails, manually change the state.

JUKEBOXEXISTS specified jukebox already exists

Explanation: The specified jukebox already exists and cannot be created.

User Action: Use a set command to modify the jukebox, or create a new jukebox with a different name.

JUKENOTINIT jukebox could not be initialized

Explanation: An operation on a jukebox failed because the jukebox could not be initialized.

User Action: Check the control, robot name, node name and group name of the jukebox, and correct as needed. Check access path to jukebox (HSJ etc.), correct as needed. Verify MDMS is running on a remote node. Then retry operation.

JUKETIMEOUT timeout waiting for jukebox to become available

Explanation: MDMS timed out waiting for a jukebox to become available. The timeout value is 10 minutes.

User Action: If the jukebox is in heavy use, try again later. Otherwise, check requests for a hung request - cancel it. Set the jukebox state to available if all else fails.

JUKEUNAVAIL jukebox is currently unavailable

Explanation: The jukebox is disabled.

User Action: Re-enable the jukebox.

LOCATIONEXISTS specified location already exists

Explanation: The specified location already exists and cannot be created.

User Action: Use a set command to modify the location, or create a new location with a different name.

LOGRESET, Log file string by string on node string

Explanation: The server logfile has been closed and a new version has been created by a user.

User Action: None.

MAGAZINEEXISTS specified magazine already exists

Explanation: The specified magazine already exists and cannot be created.

User Action: Use a set command to modify the magazine, or create a new magazine with a different name.

MBLISEXIT, mailbox listener exited

Explanation: The mailbox listener has exited due to an internal error condition. The mailbox listener is the server's routine to receive local user requests through mailbox MDMS$MAILBOX.

User Action: The mailbox listener should be automatically restarted. Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

MBLISRUN, listening on mailbox string logical string

Explanation: The server has successfully started the mailbox listener. MDMS commands can now be entered on this node.

User Action: None.

MEDIATYPEEXISTS specified media type already exists

Explanation: The specified media type already exists and cannot be created.

User Action: Use a set command to modify the media type, or create a new media type with a different name.

MOVEINCOMPL move is incomplete

Explanation: When moving volumes into and out of a jukebox, some of the volumes were not moved.

User Action: Check that there are enough empty slots in the jukebox when moving in and retry. On a move out, examine the cause of the failure and retry.

MRDERROR error accessing jukebox with MRD

Explanation: MDMS encountered an error when performing a jukebox operation. An accompanying message gives more detail.

User Action: Examine the accompanying message and perform corrective actions to the hardware, the volume or the database, and optionally retry the operation.

MRDMSG String

Explanation: This is a more detailed MRD error message which accompanies MRDERROR.

User Action: Check the MRU error message file.

NOBINDSELF cannot bind a volume to itself

Explanation: A volume cannot be bound to itself.

User Action: Use another volume.

NOCHANGES no attributes were changed in the database

Explanation: Your set command did not change any attributes in the database because the attributes you entered were already set to those values.

User Action: Double-check your command, and re-enter if necessary. Otherwise the database is already set to what you entered.

NOCHECK drive not accessible, check not performed

Explanation: The specified drive could not be physically accessed and the label check was not performed. The displayed attributes are taken from the database.

User Action: Verify the VMS device name, node name or group name in the drive object. Check availability on system.

Verify MDMS is running on a remote node. Determine the reason the drive was not accessible, fix it and retry.

NODEEXISTS specified node already exists

Explanation: The specified node already exists and cannot be created.

User Action: Use a set command to modify the node, or create a new node with a different name.

NODENOPRIV, node is not privileged to access database server

Explanation: A remote server access failed because the user making the DECnet(Phase IV) connection is not MDMS$SERVER or the remote port number is not less than 1024.

User Action: Verify with DCL command SHOW PROCESS that the remote MDMS server is running under a username of MDMS$SERVER and/or, verify that logical name MDMS$TCPIP_SND_PORTS on the remote server node specifies a port number range between 0-1023.

NODENOTENA, node not in database or not fully enabled

Explanation: The server was not allowed to start up because there is no such node object in the database or its node object in the database does not specify all network full names correctly.

User Action: For a node running DECnet (Phase IV) the node name has to match logical name SYS$NODE on that node.

For a node running DECnet-Plus (Phase V) the node's DECNET_PLUS_FULLNAME has to match the logical name SYS$NODE_FULLNAME on that node. For a node running TCP/IP the node's TCPIP_FULLNAME has to match the full name combined from logical names *INET_HOST and *INET_DOMAIN.

NODENOTINDB, no node object with string name string in database

Explanation: The current server could not find a node object in the database with a matching DECnet (Phase IV) or DECnet-Plus (Phase V) or TCP/IP node full name.

User Action: Use SHOW SERVER/NODES=(...) to see the exact naming of the server's network names. Correct the entry in the database and restart the server.

NODRIVES no drives match selection criteria

Explanation: When allocating a drive, none of the drives match the specified selection criteria.

User Action: Check spelling and re-enter command with valid selection criteria.

NODRVACC, access to drive disallowed

Explanation: You attempted to allocate, load or unload a drive from a node that is not allowed to access it.

User Action: The access field in the drive object allows local, remote or all access, and your attempted access did not conform to the attribute. Use another drive.

NODRVSAVAIL no drives are currently available

Explanation: All of the drives matching the selection criteria are currently in use or otherwise unavailable.

User Action: Check to see if any of the drives are disabled or inaccessible. Re-enter command when corrected.

NOJUKEACC, access to jukebox disallowed

Explanation: You attempted to use a jukebox from a node that is not allowed to access it.

User Action: The access field in the jukebox object allows local, remote or all access, and your attempted access did not conform to the attribute. Use another jukebox.

NOJUKESPEC jukebox required on vision option

Explanation: The jukebox option is missing on a create volume request with the vision option.

User Action: Re-enter the request and specify a jukebox name and slot range.

NOMAGAZINES no magazines match selection criteria

Explanation: On a move magazine request using the schedule option, no magazines were scheduled to be moved.

User Action: None.

NOMAGSMOVED no magazines were moved

Explanation: No magazines were moved for a move magazine operation. An accompanying message gives a reason.

User Action: Check the accompanying message, correct and retry.

NOMEDIATYPE no media type specified when required

Explanation: An allocation for a volume based on node, group or location also requires the media type to be specified.

User Action: Re-enter the command with a media type specification.

NOMEMORY not enough memory

Explanation: The MDMS server failed to allocate enough memory for an operation.

User Action: Shut down the MDMS server and restart. Contact Compaq.

NOOBJECTS no such objects currently exist

Explanation: On a show command, there are no such objects currently defined.

User Action: None.

NOPARAM required parameter missing

Explanation: A required input parameter to a request or an API function was missing.

User Action: Re-enter the command with the missing parameter, or refer to the API specification for required parameters for each function.

NORANGESUPP, slot or space ranges not supported with volset option

Explanation: On a set volume, you entered the volset option and specified either a slot range or space range.

User Action: If you want to assign slots or spaces to volumes directly, do not use the volset option.

NORECVPORTS, no available receive port numbers for incoming connections

Explanation: The MDMS could not start the TCP/IP listener because none of the receive ports specified with this node's TCPIP_FULLNAME are currently available.

User Action: Use a suitable network utility to find a free range of TCP/IP ports which can be used by the MDMS server.

Use the MDMS SET NODE command to specify the new range with the /TCPIP_FULLNAME then restart the server.

NOREMCONNECT, unable to connect to remote node

Explanation: The server could not establish a connection to a remote node. See the server's logfile for more information.

User Action: Depends on information in the logfile.

NOREQUESTS no such requests currently exist

Explanation: No requests exist on the system.

User Action: None.

NORESEFN, not enough event flags

Explanation: The server ran out of event flags. This is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis. Restart the server.

NOSCRATCH scratch loads not supported for jukebox drives

Explanation: You attempted a load drive command for a jukebox drive.

User Action: Scratch loads are not supported for jukebox drives. You must use the load volume command to load volumes in jukebox drives.

NOSENDPORTS, no available send port numbers for outgoing connection

Explanation: The server could not make an outgoing TCP/IP connection because none of the send ports specified for the range in logical name MDMS$TCPIP_SND_PORTS are currently available.

User Action: Use a suitable network utility to find a free range of TCP/IP ports which can be used by the MDMS server.

Change the logical name MDMS$TCPIP_SND_PORTS in file MDMS$SYSTARTUP.COM. Then restart the server.

NOSLOT not enough slots defined for operation

Explanation: The command cannot be completed because there are not enough slots specified in the command, or because there are not enough empty slots in the jukebox.

User Action: If the jukebox is full, move some other volumes out of the jukebox and retry. If there are not enough slots specified in the command, re-enter with a larger slot range.

NOSTATUS, no status defined

Explanation: An uninitialized status has been reported. This an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

NOSUCHDEST specified destination does not exist

Explanation: In a move command, the specified destination does not exist.

User Action: Check spelling or create the destination as needed.

NOSUCHDRIVE specified drive does not exist

Explanation: The specified drive does not exist.

User Action: Check spelling or create drive as needed.

NOSUCHGROUP specified group does not exist

Explanation: The specified group does not exist.

User Action: Check spelling or create group as needed.

NOSUCHINHERIT specified inherited object does not exist

Explanation: On a create of an object, the object specified for inherit does not exist.

User Action: Check spelling or create the inherited object as needed.

NOSUCHJUKEBOX specified jukebox does not exist

Explanation: The specified jukebox does not exist.

User Action: Check spelling or create jukebox as needed.

NOSUCHLOCATION specified location does not exist

Explanation: The specified location does not exist.

User Action: Check spelling or create location as needed.

NOSUCHMAGAZINE specified magazine does not exist

Explanation: The specified magazine does not exist.

User Action: Check spelling or create magazine as needed.

NOSUCHMEDIATYPE specified media type does not exist

Explanation: The specified media type does not exist.

User Action: Check spelling or create media type as needed.

NOSUCHNODE specified node does not exist

Explanation: The specified node does not exist.

User Action: Check spelling or create node as needed.

NOSUCHOBJECT specified object does not exist

Explanation: The specified object does not exist.

User Action: Check spelling or create the object as needed.

NOSUCHPOOL specified pool does not exist

Explanation: The specified pool does not exist.

User Action: Check spelling or create pool as needed.

NOSUCHREQUESTID specified request does not exist

Explanation: The specified request does not exist on the system.

User Action: Check the request id again, and re-enter if incorrect.

NOSUCHUSER no such user on system

Explanation: The username specified in the command does not exist.

User Action: Check spelling of the username and re-enter.

NOSUCHVOLUME specified volume(s) do not exist

Explanation: The specified volume or volumes do not exist.

User Action: Check spelling or create volume(s) as needed.

NOSVRACCOUNT, username string does not exist

Explanation: The server cannot startup because the username MDMS$SERVER is not defined in file SYSUAF.DAT.

User Action: Enter the username of MDMS$SERVER (see Installation manual for account details) and then start the server.

NOSVRMB, no server mailbox or server not running

Explanation: The MDMS server is not running on this node or the server is not servicing the mailbox via logical name MDMS$MAILBOX.

User Action: Use the MDMS$STARTUP procedure with parameter RESTART to restart the server. If the problem persists, check the server's logfile and file SYS$MANAGER:MDMS$SERVER.LOG for more information.

NOTALLOCUSER volume is not allocated to user

Explanation: You cannot perform the operation on the volume because the volume is not allocated to you.

User Action: Either use another volume, or (in some cases) you may be able to perform the operation specifying a user name.

NOUNALLOCDRV no unallocated drives found for operation

Explanation: On an initialize volume request, MDMS could not locate an unallocated drive for the operation.

User Action: If you had allocated a drive for the operation, deallocate it and retry. If all drives are currently in use, retry the operation later.

NOVOLSMOVED no volumes were moved

Explanation: No volumes were moved for a move volume operation. An accompanying message gives a reason.

User Action: Check the accompanying message, correct and retry.

NOVOLSPROC no volumes were processed

Explanation: In a create, set or delete volume command, no volumes were processed.

User Action: Check the volume identifiers and re-enter command.

NOVOLUMES no volumes match selection criteria

Explanation: When allocating a volume, no volumes match the specified selection criteria.

User Action: Check the selection criteria. Specifically check the relevant volume pool. If free volumes are in a volume pool, the pool name must be specified in the allocation request, or you must be a default user defined in the pool. You can re-enter the command specifying the volume pool as long as you are an authorized user. Also check that newly-created volumes are in the FREE state rather than the UNITIALIZED state.

OBJECTEXISTS specified object already exists

Explanation: The specified object already exists and cannot be created.

User Action: Use a set command to modify the object, or create a new object with a different name.

OBJNOTEXIST referenced object !AZ does not exist

Explanation: When attempting to allocate a drive or volume, you specified a selection object that does not exist.

User Action: Check spelling of selection criteria objects and retry, or create the object in the database.

PARTIALSUCCESS some volumes in range were not processed

Explanation: On a command using a volume range, some of the volumes in the range were not processed.

User Action: Verify the state of all objects in the range, and issue corrective commands if necessary.

POOLEXISTS specified pool already exists

Explanation: The specified pool already exists and cannot be created.

User Action: Use a set command to modify the pool, or create a new pool with a different name.

QUEUED operation is queued for processing

Explanation: The asynchronous request you entered has been queued for processing.

User Action: You can check on the state of the request by issuing a show requests command.

RDFERROR error allocating or deallocating RDF device

Explanation: During an allocation or deallocation of a drive using RDF, the RDF software returned an error.

User Action: The error following this error is the RDF error return.

SCHEDULECONFL schedule qualifier and novolume qualifier are incompatible

Explanation: The /SCHEDULE and /NOVOLUME qualifiers are incompatible for this command.

User Action: Use the /SCHEDULE and /VOLSET qualifiers for this command.

SCHEDVOLCONFL schedule qualifier and volume parameter are incompatible

Explanation: The /SCHEDULE and the volume parameter are incompatible for this command.

User Action: Use the /SCHEDULE qualifier and leave the volume parameter blank for this command.

SETLOCALEFAIL an error occurred when accessing locale information

Explanation: When executing the SETLOCALE function an error occurred.

User Action: A user should not see this error.

SNDMAILFAIL send mail failed, see log file for more explanation

Explanation: While sending mail during the scheduled activities, a call to the mail utility failed.

User Action: Check the log file for the failure code from the mail utility.

SPAWNCMDBUFOVR spawn command buffer overflow

Explanation: During the mount of a volume, the spawned mount command was too long for the buffer. This is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

SVRBUGCHECK internal inconsistency in SERVER

Explanation: You should never see this error. There is an internal error.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis. Restart the server.

SVRDISCON, server disconnected

Explanation: The server disconnected from the request because of a server problem or a network problem.

User Action: Check the server's logfile and file SYS$MANAGER:MDMS$SERVER.LOG for more information. Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

SVREXIT, server exited

Explanation: Server exited. Check the server logfile for more information.

User Action: Depends on information in the logfile.

SVRLOGERR, server logged error

Explanation: The server failed to execute the request. Additional information is in the server's logfile.

User Action: Depends on information in the logfile.

SVRRUN, server already running

Explanation: The MDMS server is already running.

User Action: Use the MDMS$SHUTDOWN procedure with parameter RESTART to restart the server.

SVRSTART, Server stringnumber.number-number started

Explanation: The server has started up identifying its version and build number.

User Action: None.

SVRTERM, Server terminated abnormally

Explanation: The MDMS server was shut down. This could be caused by a normal user shutdown or it could be caused by an internal error.

User Action: Check the server's logfile for more information. If the logfile indicates an error has caused the server to shut down then provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

SVRUNEXP, unexpected error in SERVER string line number

Explanation: The server software detected an internal inconsistency.

User Action: Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis.

TCPIPLISEXIT, TCP/IP listener exited

Explanation: The TCP/IP listener has exited due to an internal error condition or because the user has disabled the TCPIP transport for this node. The TCP/IP listener is the server's routine to receive requests via TCP/IP.

User Action: The TCP/IP listener should be automatically restarted unless the TCPIP transport has been disabled for this node. Provide copies of the MDMS command issued, the database files and the server's logfile for further analysis if the transport has not been disabled by the user.

TCPIPLISRUN, listening on TCP/IP node string port string

Explanation: The server has successfully started a TCP/IP listener. Requests can now be sent to the server via TCP/IP.

User Action: None.

TOOLARGE, entry is too large

Explanation: Either entries cannot be added to a list of an MDMS object or existing entries cannot be renamed because the maximum list size would be exceeded.

User Action: Remove other elements from list and try again.

TOOMANYSORTS too many sort qualifiers, use only one

Explanation: When you specify more than one field to sort on.

User Action: Specify only one field to sort on.

TOOMANY too many objects generated

Explanation: You attempted to perform an operation that generated too many objects.

User Action: There is a limit of 1000 objects that may be specified in any volume range, slot range or space range.

Re-enter command with a valid range.

UNDEFINEDREFS object contains undefined referenced objects

Explanation: The object being created or modified has references to undefined objects.

User Action: This allows objects to be created in any order, but some operations may not succeed until the objects are defined. Show the object and verify the spelling of all referenced objects or create them if not defined.

UNSUPPORTED1, unsupported function string

Explanation: You attempted to perform an unsupported function.

User Action: None.

UNSUPPORTED unsupported function

Explanation: You attempted to perform an unsupported function.

User Action: None.

UNSUPRECVER, unsupported version for record string in database string

Explanation: The server has detected unsupported records in a database file. These records will be ignored.

User Action: Consult the documentation about possible conversion procedures provided for this version of MDMS.

USERNOTAUTH user is not authorized for volume pool

Explanation: When allocating a volume, you specified a pool for which you are not authorized.

User Action: Specify a pool for which you are authorized, or add your name to the list of authorized users for the pool.

Make sure the authorized user includes the node name or group name in the pool object.

VISIONCONFL vision option and volume parameter are incompatible

Explanation: You attempted to create volumes with the vision option and the volume parameter. This is not supported.

User Action: The vision option is used to create volumes with the volume identifiers read by the vision system on a jukebox.

Re-enter the command with either the vision option (specifying jukebox and slot range), or with volume identifier(s), but not both.

VOLALRALLOC specified volume is already allocated

Explanation: You attempted to allocate a volume that is already allocated.

User Action: Use another volume.

VOLALRINIT volume is already initialized and contains data

Explanation: When initializing a volume, MDMS detected that the volume is already initialized and contains data.

User Action: If you are sure you still want to initialize the volume, re-enter the command with the overwrite option.

VOLIDICM, volume ID code missing

Explanation: The volume ID is missing in a request.

User Action: Provide volume ID and retry request

VOLINDRV volume is currently in a drive

Explanation: When allocating a volume, the volume is either moving or in a drive, and nopreferred was specified.

User Action: Wait for the volume to be moved or unloaded, or use the preferred option.

VOLINSET volume is already bound to a volume set

Explanation: You cannot bind this volume because it is already in a volume set and is not the first volume in the set.

User Action: Use another volume, or specify the first volume in the volume set.

VOLLOST volume location is unknown

Explanation: The volume's location is unknown.

User Action: Check if the volume's placement is in a magazine, and if so if the magazine is defined. If not, create the magazine. Also check the magazine's placement.

VOLMOVING volume is currently being moved

Explanation: In a move, load or unload command, the specified volume is already being moved.

User Action: Wait for volume to come to a stable placement and retry. If the volume is stuck in the moving placement, check for an outstanding request and cancel it. If all else fails, manually change volume state.

VOLNOTALLOC specified volume is not allocated

Explanation: You attempted to bind or deallocate a volume that is not allocated.

User Action: None for deallocate. For bind, allocate the volume and then bind it to the set, or use another volume.

VOLNOTBOUND volume is not bound to a volume set

Explanation: You attempted to unbind a volume that is not in a volume set.

User Action: None.

VOLNOTINJUKE volume is not in a jukebox

Explanation: When loading a volume into a drive, the volume is not in a jukebox.

User Action: Use the move option and retry the load. This will issue OPCOM messages to move the volume into the jukebox.

VOLNOTLOADED the volume is not loaded in a drive

Explanation: On an unload request, the volume is not recorded as loaded in a drive.

User Action: If the volume is not in a drive, none. If it is, issue an unload drive command to unload it.

VOLONOTHDRV volume is currently in another drive

Explanation: When loading a volume, the volume was found in another drive.

User Action: Wait for the volume to be unloaded, or unload the volume and retry.

VOLSALLOC String volumes were successfully allocated

Explanation: When attempting to allocate multiple volumes using the quantity option, some but not all of the requested quantity of volumes were allocated.

User Action: See accompanying message as to why not all volumes were allocated.

VOLUMEEXISTS specified volume(s) already exist

Explanation: The specified volume or volumes already exist and cannot be created.

User Action: Use a set command to modify the volume(s), or create new volume(s) with different names.

VOLWRTLCK volume loaded with hardware write-lock

Explanation: The requested volume was loaded in a drive, but is hardware write-locked when write access was requested.

User Action: If you need to write to the volume, unload it, physically enable it for write, and re-load it.

WRONGVOLUME wrong volume is loaded in drive

Explanation: On a load volume command, MDMS loaded the wrong volume into the drive.

User Action: Check placement (jukebox, slot etc.) of both the volume in the drive and the requested volume. Modify records if necessary. Unload volume and retry.

SLS and ABS Comparisons

If you are migrating from SLS to ABS as your backup product, the information presented here may help you equate SLS to ABS backup functions.

Table I–1 lists the attributes in an SLS SBK file and gives the equivalent ABS attribute.

Table I–1 Comparing SLS and ABS Backup Attributes

SBK Attribute	Equivalent ABS DCL Attribute
$ DAYS_1 :==	INTERVAL on the save request. If INTERVAL=EXPLICIT is used, you must set the EXPLICIT qualifier.
$ TIME_1 :==	START_TIME on the save request.
$ NODE_1 :==	Defaulted to the node where the save request is created. For UNIX and NT save requests, EXECUTION_NODE means the node specified for the storage policy used for the UNIX or NT save request.
$ PREALLOC ==	VOLUME_SET on the storage policy. You must manually allocate the volume and set VOLUME_SET to the first volume in the volume set.
$ BACKUP_TYPE :==	OBJECT_TYPE on the save request.
$ FILES_1 :==	The include specification on a save request. You can create a comma separated list of disk or file names. To add or remove disk or file names on an existing save request (or restore request), use the /ADD or /REMOVE qualifiers.
$ PROGRESS :==	None
$ QUALIFIERS :== RECORD CRC INTERLOCK IMAGE INCREMENTAL BEFORE SINCE EXCLUDE PRIVS :==	See the following equivalences: /ACTION=option on the environment policy. DATA_SAFETY=option on the environment policy. LOCKING_OPTION=option on the environment policy. /FULL on the save request. /INCREMENTAL on the save request. /BEFORE on the save request. /SINCE on the save request. /EXCLUDE on the save request. /PRIVS on the environment policy. All other qualifiers can be specified using /QUALIFIERS on the save request.
$ SAVESET_GEN :==	/NAME=Save_request_name.
$ HISTORY_SET :==	/CATALOG on the storage policy.

Table I–1 Comparing SLS and ABS Backup Attributes

SBK Attribute	Equivalent ABS DCL Attribute
$ SBUPDT_Q :==	None
$ LISTING_GEN :==	None
$ FULL :==	/LISTING_OPTION on the environment policy.
$ PRINT_Q :==	None
$ AUTOSEL :==	None
$ TAPE_POOL :==	/TAPE_POOL on the storage policy.
$ REEL_SIZE :==	None
$ NOTES :==	None
$ SCRATCH_DAYS :=	/EXPIRATION or /RETAIN on the storage policy or the save request. /EXPIRATION and /RETAIN are mutually exclusive. Use one or the other depending on whether you wish to specify a date (EXPIRATION) or number of days (RETAIN).
$ CONTINUE :==	/VOLUME_SET on the storage policy. This is set automatically after first save operation using that storage POLICY. Use the CONSOLIDATION options on the storage policy to control the volume set by either INTERVAL, COUNT or SIZE.
$ MEDIA_TYPE :==	/TYPE_OF_MEDIA on the storage policy.
$ DENSITY :==	None
$ N_DRIVES :==	/DRIVE_COUNT on the environment policy.
$ DRIVE_TYPE :==	DRIVE_LIST on the storage policy.
$ STATUS_MAIL :==	/NOTIFICATION=options on the environment policy.
$ LOG_FILE :==	ABS always creates a log file. Use /LOG_OPTION on the environment policy to determine when to keep the log file.
$ OFFSITE_DATE :=	None
$ ONSITE_DATE :==	None
$ CONTLOADOPT ==	None. ABS defaults to CONTLOADOPT=0.
$ TAPE_LABELS ==	None
$ MNTFLAGS ==	None
$ NEXT_JOB :==	None. However, you could set up dependencies in the scheduler interface being used.
$ QUICKLOAD :==	ABS$storage_class_name LOAD_TIMEOUT (user-defined logical in the file SYS$MANAGER:SYLOGICALS.COM)
$ POST_PROCESSING_FIRST_:==	PROLOGUE on the environment policy.
$ PRE_PROCESSING_EACH :==	PROLOGUE on the save request.
$ POST_PROCESSING_EACH :==	EPILOGUE on the save request.

Table I–1 Comparing SLS and ABS Backup Attributes

SBK Attribute	Equivalent ABS DCL Attribute
$ POST_PROCESSING_LAST_:==	EPILOGUE on the environment policy.

J

SLS To ABS Conversion

J.1 SLS To ABS Conversion

This section is intended for SLS users who are considering a conversion from SLS to ABS.

The following sections:

- Give a comparison of SLS and ABS System Backup Policy Management
- Provide an overview of SLS and ABS System Backup Operations
- Give a general overview of the Media and Device Management Services (MDMS)
- Identify the process for converting from SLS to ABS but does not
- Give a general overview of SLS usage and capabilities
- Include information about converting SLS Standby Archiving to ABS

J.2 Why Convert from SLS to ABS?

SLS is a legacy product of Compaq Computer Corporation. Although it is fairly reliable once it is configured, learning to configure SLS is quite a challenge. In addition, when problems do occur, diagnosing the problem and making a fix to the source code is very difficult, and sometimes impossible.

ABS was released in 1995. ABS has the following advantages over SLS:

- NT and UNIX Clients
- Consolidated Policy Management
- More Intuitive Policy Organization, with Shared Policies
- Better Logging and Diagnostic Capabilities
- Automatic Full and Incremental Operations
- More Versatile User-requested Operations
- On Disk Backups
- A Sophisticated and Reliable Media Management Subsystem

ABS uses a new version of the Media and Device Management Services (MDMS). A brief overview of MDMS is given in this Appendix with more information elsewhere in this guide. MDMS provides a utility that automatically converts the SLS volume, slot and magazine databases, and the TAPESTART.COM command definitions, to MDMS databases. The MDMS conversion is discussed in the Appendix "Converting SLS/MDMS V2 to V3".

ABS has the ability to read old SLS history sets and restore data from old SLS backups, so conversion to ABS can be performed in stages on different nodes over time - this is known as a rolling upgrade.

J.2.1 Consolidated Policy Management

ABS policy, what gets backed up and how it is stored in a single policy database on an OpenVMS cluster in your network is called Central Security Domain, or CSD. Through this centralized policy database, ABS supports the ability to control your entire network's backup policy from a single location, or allows you to distribute the responsibility for backups and restores to other systems on other OpenVMS nodes. This is contrary to SLS method of storing SBK files on each node to be backed up, where a minor change in tape drive configuration or policy can take hours or days to propagate through all SBK files on a large network.

J.2.2 More Intuitive Policy Organization

ABS Policy is organized into simple policy objects:

- Storage Class objects, contain information about where data is stored after it is backed up. This includes what type of media is used, how long the data is stored, and what access is allowed to the data.
- Execution Environment objects, contain information about how data is moved into and out of Storage Classes. This includes data safety options, the user profile under which the backup is performed, logging and listing options, and client file interlocking options.
- Save Request objects, contain information about what data is to be backed up and on what schedule. Multiple sets of disks, files or databases can be combined in a single Save Request. The scheduling options of ABS include "complex" schedules, such as a Weekly Full backup with a Daily Incremental, or Log based schedules.
- Restore Request objects, contain information about restoring data from a Storage Class. This includes what data to restore, and what selection criteria (such as before or after dates) to be used to select the data. ABS supports a "Full Restore" operation, through which both Full and Incremental restore of a disk or set of files are performed automatically.
- Catalog objects, contain information about what data is stored in the ABS Storage Classes. The Catalogs are used for generating reports on the content of Storage Classes, as well as to provide information for restoring files, disks or other data objects.

J.2.3 Better Logging and Diagnostic Capabilities

ABS provides improved logging and diagnostic capabilities. Notification of job completion and status can be sent via MAIL or OPCOM. ABS log files are easier to read and interpret than SLS log files.

J.2.4 UNIX and NT Clients

ABS provides backup and restore capabilities for NT and UNIX clients. This allows the workstation disks to be backed up and cataloged with the reliability and availability of OpenVMS.

J.2.5 Automatic Full and Incremental Operations

ABS makes backup scheduling easy by providing "complex" backup schedules, such as Weekly Full with Daily Incremental, and log based scheduling. These backup schedule minimize tape usage and backup time by only doing occasional Full backups, with Incremental backups making up the bulk of the data movement operations. ABS also provides the Full Restore capability for a disk or other data object, automatically restoring the necessary Full and Incremental backups to retrieve the data.

J.2.6 More versatile User requested Operations

ABS provides users the ability to backup and archive their own files, if allowed by the site management. By setting Access Control Lists (ACL's) on Storage Classes and Execution Environments, you can allow users to save data and restore data without intervention of the system manager.

J.2.7 Disk Storage Classes

ABS provides the ability to backup disk savesets. This ability is especially useful for optical media, since many optical devices appear as disk devices to the operating system. In addition, disk storage classes can be used for backup operations in which the savesets need to be online for quick restores.

J.3 SLS and ABS System Backup Policy Overview

This section gives you an overview of backup policy as implemented in SLS and ABS, comparing the two products representation and organization of policy.

Backup policy can be viewed as:

- What gets backed up
- When it gets backed up
- Where it gets backed up to
- Who backs it up, and who can access the backed up data
- How it gets backed up

J.3.1 SLS Policy with ABS Equivalents

J.3.1.1 System Backup Policy Configuration

SLS Backup policy is stored in SBK files, which are DCL command procedures. These SBK files are located on the system where the backup is to be run. The SBK files define a variety of DCL symbols, which identify the What, When, Where, Who and How of each backup to be performed.

Table J–1 gives the primary DCL symbols of the SBK file which identify each component of the backup policy. There are numerous other parameters in SBK and ABS policies, but this table gives an overview.

See Section J.7 for a complete description of each SBK Symbol and its ABS equivalent.

Table J–1 DCL Symbols and ABS Equivalent

Policy Component	SBK Symbol(s)	ABS Equivalent
What gets backed up	BACKUP_TYPE FILES_n NODE_n	Save Request Include Specification(s) and Object Type
When it gets backed up	DAYS_n TIME_n	Save Request Scheduling Option and Start Time

Where it gets backed up to	MEDIA_TYPE TAPE_POOL SCRATCH_DAYS DRIVE_TYPE	Storage Class
Who backs it up, and who can access the backed up data	SLS PROTECTION	Execution Environment User Profile, Storage Class ACL
How it gets backed up	QUALIFIERS QUALIFIERS_n PRIVS N_DRIVES	Execution Environment

In ABS, Backup Policy is consolidated in a network wide Policy Database. The policy is created and modified using ABS DCL Interface, or ABS Graphical User Interface (GUI).

J.3.1.2 Defining Your System Backup Policy

To define a backup policy in SLS, you log onto the system where the backup is to be performed, copy SYSBAK.TEMPLATE to a new SBK file, and then edit the SBK files using a regular text editor. You define each SBK symbol according to what you want the backup policy to do.

This manual editing of SBK files tends to be error prone. Each SBK symbol must be defined in a particular syntax, and it is easy to make typing errors. When syntactic errors are made in the SBK symbols, it is often unclear in the SLS log files what the actual problem is, and how it can be fixed.

In addition, in a fairly large network, the management of the SBK files can become quite cumbersome, requiring substantial time and organization on your part.

In ABS, Backup Policy is consolidated in a “Policy Engine”, which contains all the backup policy for your network. ABS Backup Policy is defined by creating one or more Storage Classes, one or more Execution Environments, and one or more Save Requests. This is done using either the DCL command interface, or using the Graphical User Interface (GUI). ABS has an easy-to-use GUI for convenience of defining your backup policy.

Storage Classes and Execution Environments can be shared by multiple Save Requests. From the Central Security Domain (where your ABS Policy Engine is installed), you can create Save Requests for disks, files or other data objects on remote nodes, which share the common Storage Classes and Execution Environments.

J.3.1.3 Restoring Data

Selective File Restores

In SLS, to restore a particular set of files for a user, the Operator or System Administrator must regularly get involved. This is because the SLS History Sets, which stores information needs to perform restore, are not accessible to regular users.

In ABS, Access Control List (ACL) on the Storage Class can be set up to allow access to all users, or only selected users. This allows individual users to restore their own files from these Storage Classes.

When using the SLS Storage Restore command, the user (or Operator) must either specify the specific volume, saveset and version of the file to restore. When using the Storage Restore screen, no choice is given about the version of the file to be restored.

In ABS, a date oriented selection for a file to restore is provided. Normally, users will request the most recent version of a file to be restored, since user restores are regularly due to accidental

deletion. However, in ABS, the user can specify that the most recent copy before a particular date should be restored. This allows the user to fine tune the restore operation.

Full Restore Operations

In SLS, an operation called a Full Disk Restore is provided. This is a fairly manual method by which a Full backup and associated Incremental backups can be used to restore a full disk.

To perform a Full Disk Restore in SLS, you must manually select the Full and Incremental backups to be applied, and in what order. Although this provides a great deal of versatility, it can also be error prone. Once an automated backup policy is set up, many customers are not familiar with the specific backups which are done, and determining the correct order of a restore can be difficult.

In ABS, when a Restore Request is created for a Disk, the type of restore can be specified as “Full Restore”. This causes ABS to automatically find the most recent Full Backup, and all subsequent incremental backups (of appropriate level in the case of log-based backup schedules), and commence the restore in the correct order.

J.3.1.4 Media Management

ABS uses a new Media And Device Management Services (MDMS) component that supports the concept of a domain. An MDMS domain has scope across multiple geographical locations each with their own nodes, jukeboxes, drives and volumes. Communication within the domain utilizes TCP/IP, DECnet Phase IV and/or DECnet-Plus at the user's choice. When upgrading from SLS to ABS, it is first necessary to convert the SLS volume, magazine and slot databases, and the TAPESTART.COM definitions, to MDMS databases. A utility is provided for this purpose and this is described in the Appendix "Converting SLS/MDMS V2 to MDMS V3". MDMS has been designed so that this conversion can be performed as a rolling upgrade, starting with the set of nodes designated as database servers. These MDMS V3 database servers can support V2.x clients running ABS or SLS.

J.3.2 ABS Overview with SLS Equivalents

J.3.2.1 Policy Configuration

In ABS, the Backup Policy is stored in five different types of policy objects:

- Storage Classes
- Execution Environments
- Save Requests
- Restore Requests
- Catalogs

These policy objects each have a Name, and Access Control List (although a catalog’s access is controlled through Storage Class). They are created using the ABS DCL Command, or using the Graphical User Interface (GUI).

These policy objects (except the catalogs) are stored in a central location, called the Policy Database. The Policy Database resides on a single OpenVMS cluster in your network, called the Central Security Domain (CSD). The CSD should also support the MDMS database servers. The CSD controls all of the Storage Classes and Execution Environments used throughout the network, but Save and Restore Requests can be created from any OpenVMS node in the network.

The ABS policy can be completely controlled from the CSD, or responsibility for creating backups and restores can be distributed to the system managers on other OpenVMS nodes. The

Access Control on Storage Classes and Execution Environments determine which OpenVMS nodes in the network are allowed to create Save and Restore requests referencing these objects.

Catalogs are stored on each node where backups are performed. This substantially reduces the network bandwidth required for doing backups across the network to a centrally located robot or storage facility.

J.3.2.2 Storage Class

ABS Storage Class contains information about where backed up files and other data objects (such as databases or UNIX and NT file systems) are to be stored. As many Storage Classes as necessary can be created in ABS Policy Database. Multiple Save Requests can share a single Storage Class.

The information in a Storage class, with each parameter's SBK file equivalent are given in Table J–2.

Table J–2 Storage Class Parameter and SBK File Equivalent

Storage Class Parameter	SBK Equivalent	Meaning
Name	CONTINUE	A common name which can be refer enced by multiple Save Requests
Primary Archive Location	<None>	For on disk backups, this gives the disk and top level directory where the savesets will be stored.
Primary Archive Type	<None>	This determines whether the Storage Class is tape based (type is MDMS) or disk based (type is FILES-11)
Owner	<None>	Determines the NODE::USER of the owner of the Storage Class. The owner always has CONTROL access.
ACL	PROTECTION	Determines the access to the backed up data. ABS provides full ACL based access, while SLS provides only Open VMS style system, owner, group and world access.
Tape Pool	TAPE_POOL	The MDMS pool from which vol umes will be allocated for backups.
Type of Media	MEDIA_TYPE	The MDMS media type to be allocated for backups.
Retain	SCRATCH_DAYS	The number of days the backed up data will be saved before the tapes are recy cled. Note that a Save Request can specify a retention *shorter or equal to* the value in the Storage Class.
Consolidation Criteria	CONTINUE	This set of parameters determines how backup savesets will be consolidated onto tapes. For example, if the Consol idation Interval is set to 7 days, savesets will be appended onto a vol ume set for 7 days before a new vol ume set is created.

Catalog Name	HISTORY_SET	The name of the catalog which stores data about what files have been backed up, and where they are located.
Number of Streams	<None>	The number of simultaneous Save requests which can be writing into the Storage Class. Determines the number of MDMS volume sets simultaneously active in the Storage Class.
Media Location	<None>	The MDMS onsite location field to match for allocating volumes for back-ups.
Drive List	DRIVE_TYPE	The list of specific drives to be used for backup operations in this Storage Class. Normally, this should be man aged through MDMS drive objects.

J.3.2.3 Execution Environment

An ABS Execution Environment (or simply Environment) object stores information about *how* backups are to be performed. This includes parameters regarding data safety, file interlocking, notification, and so forth. As many Environment objects as necessary can be created in the ABS Policy Database. Multiple Save Requests can share a single Execution Environment.

The information in an ABS Environment, along with each parameter's SBK equivalent, is given in Table J–3.

Table J–3 ABS and SBK Equivalent

Equivalent Parameter	SBK Equivalent	Meaning
Name	<None>	Identifies the Environment for ref erence by Save Requests
Owner and ACL	<None>	Identifies the owner and access to the Environment.
Data Safety Options	QUALIFIERS	A bitmask containing the data safety options to be applied during the backup. Data safety options include CRC checking, and full data verification.
Listing Options	LISTING_GEN FULL	Determines whether a listing file is produced, and whether the listing is a "full" listing or a "brief" list-ing.
Span Filesystem Options	<None>	For UNIX file systems, determines whether the entire filesystem is backed up, even if it crosses multi-ple physical devices.

Links Only	<None>	For UNIX file systems, determines whether ABS backs up only the logical links, or backs up the data as well.
Compression Options	<None>	For UNIX file systems, determines the type of compression to be applied to the savesets.
Original Object Action	QUALIFIERS	Determines the action to be taken on the original data objects (e.g. the files backed up). Options include None, Record Backup Date, or Delete
User Profile	PRIVS	Determines the username, privi-leges and access rights used during the backup operation. The special keyword "<REQUESTER>" indi cates the backup operations should be performed with the username, privileges and access rights of the person issuing the ABS Save Com mand.
Notification Criteria	REPLY_MSG STATUS_MAIL	Determines when notification occurs, what method is used, and who is notified.
Locking Options	QUALIFIERS	Determines how much interlocking is done between the backup in progress and an active file system. Options include Ignore File Writ ers, and Hot Backup.
Number of Drives	N_DRIVES	Determines the number of tape drives to be used during the backup operations.
Staging Option	<None>	Determines whether catalog entries are staged to a sequential file and then inserted into the actual catalog at a later time. This improves per formance of the backup.
Retry Interval and Count	<None>	Determines how often and how many times a failed backup should be retried.
Prolog Command	PRE_PROCESS_FIRST	A command to be executed when the backup starts. Contrast to Save Request Prolog.
Epilog Command	POST_PROCESS_LAST	A command to be executed when the backup completes. Contrast to Save Request Epilog.

J.3.2.4 Save Request

An ABS Save Request identifies the data to be backed up, the Storage Class and Environment to be used for the backup(s), and the schedule on which the request is to be executed. As many Save Requests as necessary can be created in the ABS Policy Database, and multiple Save Requests can share any Storage Class or Execution Environment.

The information stored in a Save Request, with each parameter's SBK equivalent, is given in the Table J–4.

Table J–4 Save Request and SBK Equivalent

Save Request Parameter	SBK Equivalent	Meaning
Name	SBK File name	Identifies the group of backup operations to be performed.
Movement Type	QUALIFIERS	Determines whether the opera tions are Full, Incremental or Selective (i.e. individual file) oper ations.
Source Node	NODE_n	Node on which the data resides.
Include Specification	FILES_n	Identifies the data to be backed up. Multiple include specifications can be given on a single Save Request, and each can have a different Source Node and Object Type.
Object Type	BACKUP_TYPE	Gives the type of the data. ABS Supports many different types of data, including OpenVMS Files, UNIX Files, Oracle RDB Data bases, and so forth.
Agent Qualifiers	QUALIFIERS	Allows backup agent specific qual ifiers to be added to the command used to backup the data.
Since and Before Date	QUALIFIERS	Determine whether data objects to be backed up should be selected based upon creation/modification date.
Exclude Specification	QUALIFIERS	Determines selected data objects to be excluded from the backup.
Storage Class Name	CONTINUE	Gives the name of the Storage Class into which the data is backed up.
Environment Name	<None>	Gives the name of the Execution Environment to be used for the backup operations.

Start Time	TIME_n	Indicates the time at which the Save Request should start each time it is scheduled. Note that although an SBK can provide mul tiple DAYS_n and TIME_n parameters, an ABS Save Request is restricted to a single Start Time and Interval.
Schedule Options and Explicit	DAYS_n	Identifies the repeat interval for the Save Request. ABS provides a variety of pre defined simple inter vals, such as Daily, Weekly, Monthly, as well as several “com plex” intervals, such as Weekly Full with Daily Incremental, and log based schedules. See the Appendix "Log-n Backup Schedules" for a full description of log based schedules.
Prolog Command	PRE_PROCESS_EACH	A command to be executed before each backup operation within the Save Request starts. Contrast to Environment Prolog.
Epilog Command	POST_PROCESS_EACH	A command to be executed after each backup operation within the Save Request completes. Contrast to Environment Epilog.

J.3.2.5 Restore Request

An ABS Restore Request stores information about files, disks, or other data objects to be restored from a Storage Class. As many Restore Requests as necessary can be created in the ABS Policy Database.

The information stored in a Restore Request are given in Table J–5. Because SLS does not provide a formal Restore Request mechanism, no SBK or other SLS equivalents are given in the Table J–5.

Table J–5 Restore Request Parameter Information

Restore Request Parameter	Meaning
Name	Identifies the set of restore operations to be performed.
Movement Type	Identifies whether a Full, Incremental or Selective restore is performed. Note that when a Full restore is requested, ABS automatically applies the necessary Incremental restores to bring the restored data up-to-date.
Source Node	Gives the node where the data to be restored resided when it was backed up.
Include Specification	Identifies the disk, filesystem, set of files or other data objects to be restored. Multiple include specifications can be given in a single Restore Request, each with its own Object Type.
Object Type	Identifies the type of data to be restored.

Agent Qualifiers	Allows backup agent specific qualifiers to be added to the command used to restore the data.
Since and Before Date	Determine that data objects *as of a particular date* should be restored. For example, specifying a Before Date of last Thursday would indicate that the most recent copy of the data *before this date* should be restored.
Storage Class Name	Gives the name of the Storage Class from which the data should be restored. The Storage Class identifies the catalog from which restore information is retrieved.
Environment Name	Gives the name of the Execution Environment to be used for the restore operations.
Output Location	Indicates the data should be restored to an alternate loca tion. By default, ABS restores the data to its original loca tion.

J.3.2.6 Catalog

An ABS Catalog object stores information about what backup operations have been performed, and what files or other data objects have been backed up. ABS Catalog object combines the SLS concepts of a Summary File, a System History Set, and a User History Set.

Catalog objects are accessed through one or more Storage Classes. More than one Storage Class can share a Catalog, or each Storage Class can have a separate catalog. The ABS Catalogs can also be queried using the ABS LOOKUP command (or associated GUI function) and the ABS REPORT SAVE command.

ABS Catalogs are created using ABS$SYSTEM:ABS_CATALOG_OBJECT.EXE utility. An ABS Catalog has the following parameters, as specified in Table J–6 when it is created:

Table J–6 ABS Parameter and SLS Equivalent

Catalog Parameter	SLS Equivalent	Meaning
Name	SBK HISTORY_SET TAPESTART HSTNAM_n	Gives the name of the catalog to be refer enced by a Storage Class.
Type	System History or User History	Provides the type of information stored in the catalog. The type can be Brief, SLS or Full Restore catalog. The SLS Catalog type is provided to allow a Storage Class to retrieve information from SLS History sets.
Owner	<None>	Determines the Owner of the catalog. The Owner has full access to the catalog.
Use Staging	Default behavior for SLS history files	Determines whether catalog entries are staged to a sequential file during the backup, and inserted in the actual catalog at a later time.

J.4 SLS and ABS Operation Overview

This section gives you a comparison of SLS and ABS operations. This gives information about how Save Requests are executed in each product, with similarities and differences between the two pointed out.

J.4.1 Scheduling

J.4.1.1 SBK Symbols for Scheduling

In SLS, the SBK symbols DAYS_n and TIME_n identify the schedule and start time for the save request to execute. Each DAYS_n symbol can have one of the following forms:

- A list of day names
  - Example: MONDAY, WEDNESDAY, FRIDAY
- A day offset within the month
  - Example: MONTH + 2 (second day of the month)
  - Example: MONTH - 10 (10 days before the end of the month)
- A day within a week offset within a month
  - Example: MONTH + 2 * THURSDAY (second Thursday of the month)
  - Example: MONTH - 2 * Sunday (two Sunday before the end of the month)

Every night at midnight, a special utility process in SLS is executed, which scans all the SBK files on the system. It determines which SBK files are to be executed that day and at what time. It then submits a Batch job for each SBK, specifying the start time of the batch job to match the TIME_n parameter.

J.4.1.2 ABS Scheduler Interface Options

ABS allows the use of different scheduler interfaces to schedule its Save Requests. A Save Request is scheduled using a Start Time and a Scheduling Option. ABS uses the programming interface the OpenVMS Queue Manager as the default scheduler interface option.

A variety of pre-defined scheduling options are provided:

- One time only (the Save Request is removed after 72 hours)
- On demand (run with an explicit SCHEDULE RUN command)
- Daily
- Weekly
- Bi-Weekly
- Monthly
- Quarterly
- Semi-Annually
- Daily with Weekly Full
- Log-based schedules

In addition, ABS provides access to explicit interval schedules. This requires a 3rd party scheduler product which supports complex interval setting.

An important difference between SLS and ABS is that an ABS Save Request can only have a single schedule and start time, while an SLS SBK file can have multiple DAYS_n and TIME_n parameters. However, because the ABS Save Request can be run using a 3rd party scheduler, any number of scheduler jobs can be created to run the Save Request as needed.

Another important difference between ABS and SLS is SLS's ability to specify a list of day names. To provide the same functionality in ABS, a 3rd party scheduler product is required which allows setting specific days for scheduled jobs.

J.4.2 Types of Operations

This section discusses the various types of operations performed by SLS and/or ABS, and identifies similarities and differences between the two products.

J.4.2.1 System Backups

System Backups are the type of backup which is performed by the system on behalf of the users. Normally, this type of backup backs up entire disks or file systems, which can then be used to restore any particular file or set of files for any particular user.

The System Backup is what implemented via the SLS SBK file. It has these characteristics:

- It is always executed using the SLS account, with the privileges identified by the PRIVS symbol.
- The type of the backup is identified in the QUALIFIERS symbol (or QUALIFIERS_n). For example, if /IMAGE is included in QUALIFIERS, this indicates a full, image backup of the disk should be performed. In addition, the QUALIFIERS symbol provides many of the operational characteristics of the backup, such as whether to do a record pass, whether to ignore file writers, and so forth.
- SLS Executes the SBK file by submitting a Batch job, which then runs a command procedure called SYSBAK.COM. This command procedure is responsible for executing the SBK file, allocating tape drives, mounting volumes, and so forth.
- The actual backup operations are performed using OpenVMS Backup or Oracle RMU Backup, based upon the backup type specified by the BACKUP_TYPE symbol. These utilities are executed in a subprocess created by SYSBAK.COM, which then communicate to SYSBAK via mailboxes, or pseudo terminals in the case of Oracle RMU backup.
- The actual backup operations are run using a utility called SLS$SYSBAK.EXE. This utility creates the subprocesses to house OpenVMS or Oracle RMU backup, and communicates via mailboxes.
- The log file created in SLS$SYSBAK_LOGS gives the whole output of the execution of SYSBAK.COM.
- Many functions in SLS are provided by "DCL Utilities". These are images which are executed from SYSBAK.COM to perform functions such as sending mail, opening and reading mailboxes, mounting tapes, and so forth.
- Some of the tape, robot and drive operations performed during an SLS system backup are performed by SYSBAK.COM via "DCL Utilities", while others are performed in the SLS$SYSBAK image itself. This can cause inconsistencies in the way that load requests are handled depending on where in the backup operation the request is issued.
- The history set entries for each saveset created and each file backed up during the System Backup operation are written to a temporary file. After the System Backup completes, a batch job is created, which runs SYS$SBUPDT.EXE (System Backup Update). This utility moves the history set entries from the temporary file into the actual SLS History Files.

In ABS, a System Backup is performed by setting up an Execution Environment whose User Profile indicates the ABS account (with particular privileges and access rights). Then, any Save Request which uses that Environment would be considered a "System Backup".

ABS installation kit provides out-of-the box policy objects for performing system backups. These are the SYSTEM_BACKUPS Storage Class, and the SYSTEM_BACKUPS_ENV execution environment. If the parameters on these policy objects are not suitable to your environment, they can be modified using the ABS SET command (or equivalent GUI functions).

An ABS System Backup has the following characteristics:

- It is run using the User Profile of the ABS account.
- The type of the backup is given on the Save Request, in combination with the scheduling option if a “complex” scheduling option was chosen. For example, if the “Daily with Weekly Full” scheduling option was chosen, ABS will automatically decide whether to do a FULL or INCREMENTAL backup on any particular day.
- The Save Request is run using the selected scheduling mechanism - by default OpenVMS batch queues are used. The request is executed on the interval specified by the Save Request, and always runs under the ABS account.
- The Save Request runs a program called the ABS Coordinator. It is called this because it coordinates the execution of a Save Request. As its command line parameter, the ABS Coordinator is given the UID (Universal Identifier) of the Save Request to be executed.
- The ABS Coordinator retrieves the Save Request and its associated Storage Class and Execution Environment from the Policy Database. This may require a network connection if the Policy Database is on another node.
- The ABS Coordinator then allocates volumes (if necessary) and loads them using the Media and Device Management Services. The parameters for these volume operations come from the Storage Class.
- Once volumes are accessed, the Coordinator then creates multiple “Data Mover” threads to execute each backup operation specified by the Save Request. Each Data Mover then creates a subprocess to hold the backup agent for the given backup operation.
- The actual backup operations are performed by ABS using “Backup Agents”. ABS Supports many different Backup Agents, including OpenVMS Backup, Oracle RMU Backup, and gtar in the case of UNIX and Microsoft Windows/NT clients.
- During each backup operation, the Catalog entries for each operation performed and each data object backed up are either written directly to the Catalog, or are staged into a temporary staging file. After the Save Request completes, the staged entries are moved into the actual ABS Catalog using a detached process.

As shown above SLS and ABS System Backup operations are very similar in their overall operation. Both use subprocesses to perform the actual backup operations using other utilities, or Backup Agents. Both produce a catalog of the operations performed and the files (or data objects) backed up.

However, SLS and ABS System Backup operations have these important differences:

- ABS Save Requests are run using the current scheduler interface option, while SLS SBK files are executed using an SLS specific Batch job.
- ABS System Backups execute under the ABS account, while SLS SBK files execute under the SLS account.
- ABS Save Requests are coordinated using a program called the Coordinator, which provides good error reporting and recovery, as well as consistent tape management and logging. SLS uses SYSBAK.COM, which is a convoluted, hard-to-understand command procedure.

- The policy to be executed by ABS is retrieved from a central Policy Database, while the SLS SBK file is executed locally, and the SBK Symbols used accordingly. ABS Save Requests can share Storage Classes or Environments.
- ABS supports many different Backup Agents, while SLS supports just OpenVMS Backup and Oracle RMU Backup.
- ABS provides optional catalog staging, while SLS system backups always require staging.
- The ABS Log Files are clearer and easier to understand than the SLS log files.

J.4.2.2 Full and Incremental Operations

A “Full” backup operation is one which saves all the information on a disk, including any file system specific information. OpenVMS Backup calls these “Image” backups.

An “Incremental” backup only saves data and directory structure that has changed since either a particular date, or since each file was backed up.

Usually, a backup policy combines these two types of operations. Although a Full backup is desirable because it contains all of the data on a disk or filesystem at the time of the backup, it also uses more tape, is more time consuming, and occupies more catalog space. An Incremental backup uses less tape, less time, and less catalog space, but requires more time during the restore of a full disk.

SLS provides indirect access to Full and Incremental operations. In the SBK file, the QUALIFIERS symbol can be defined to contain the string “/IM” (Image) or “/SINCE=BACKUP” (Incremental) to manually determine whether a particular save operation is a Full or Incremental backup. You must explicitly set up the Full and Incremental schedule using the QUALIFIERS symbol.

In ABS, Full and Incremental operations can be automated using the “complex” schedules:

- Daily with Weekly Full
- Log-2
- Log-3

See the Appendix "Log-n Backup Schedules" for full information on these schedules.

The “complex” scheduling options on a Save Request cause the ABS Coordinator to automatically decide the correct operation to perform each time it is executed. For example, a fully functional, efficient backup schedule for a set of disk can be set up to run each night at 6:00PM with the single ABS Command:

```
$ ABS Save DISK$USER1:,DISK$USER2:,DISK$USER3:,DISK$USER4: -
_$ /Name=NIGHTLY_BACKUPS/Start=”18:00”/Schedule=LOG-2 -
_$ /Storage=SYSTEM_BACKUPS
```

Then, if one of these disks goes bad, for example DISK$USER3:, you can restore the whole disk to the previous night’s backup by issuing the commands:

```
$ DISMOUNT/NOUNLOAD/CLUSTER DISK$USER3! prepare for restore
$ ABS Restore/Full DISK$USER3:/Storage=SYSTEM_BACKUPS
```

ABS will automatically find the most recent Full save, apply that to the disk, and then apply each subsequent Incremental backup in the correct order to the disk.

J.4.2.3 Selective Operations

Selective Backup operations are portions of an entire disk or filesystem. For example, you might want to only backup a particular user’s directory, or a particular file or set of files.

A Selective Operation in SLS is identified when neither "/IM" (Image) nor "/SINCE=BACKUP" (Incremental) is found in the QUALIFIERS symbol. This indicates that the files given in FILES_n should be backed up "as is", and not as part of a whole disk or file system.

In ABS, the type of operation is specified on the Save Request. If the operation is specified as "Selective", then sets of files or other data objects can be backed up.

J.4.2.4 User Requested Operations

SLS provides the Storage Save command, which allows individual users or the system administrator to backup files "on demand". This type of operation is called a "User Backup" under SLS, because it has the following characteristics:

- It is normally requested by a user
- The tapes used for the backup are owned by the user

In ABS, any user with appropriate access levels can issue an ABS Save command, and back up their own files. It is access to the Storage Classes and Execution Environments which constrain which tapes and tape drives can be used by individual users.

For example, if you set the Access Control List (ACL) on the SYSTEM_BACKUPS Storage Class to be:

```
/ACCESS=(USER=*::*,ACCESS="READ+WRITE")
```

then any user can write into the Storage Class (do backups to it) or Read from the Storage Class (restore files from it).

The ABS installation kit provides an out-of-the-box set of policy objects for user backup operations. These are the Storage Class USER_BACKUP and the Environment USER_BACKUP_ENV.

The User Profile in the Execution Environment determines the context in which the backup operations are performed. Except in special cases, Environments which are accessible to the average users will have a user profile specifying the keyword "<REQUESTER>" as the user under which the backups are to be performed. This causes ABS to capture the user's username, privileges and access rights when they create a Save Request using this Environment, and use these parameters during the backup operations.

In ABS, all volumes are owned and managed by the ABS account. The primary goal of ABS is data safety. This primary goal precludes allowing individual users to manage their own tapes, since the user may destroy data accidentally, or misuse the tapes. Access to the tapes owned by ABS is allowed by setting the Access Control on Storage Classes.

Using the USER_BACKUP Storage Class and USER_BACUPS_ENV execution environment, users can issue their own Save and Restore requests. These backup operations share a common pool of tapes and a common catalog with other users, but each user can only access their own backed up data on these tapes.

If you want a user to be limited to a particular set of tapes, or record their backups in a separate catalog, ABS also allows this to be configured by issuing the steps below:

1. Create an MDMS Pool for the user and specify that the user is authorized to access volumes in the pool. This limits the volumes the user is able to use for their personal backups. This step is not required if you wish to allow users to share a pool of tapes.
2. Create a personal catalog for the user using the ABS_CATALOG_OBJECT utility. If desired, you can move the catalog files to the user's directory, then redefine the ABS$CAT-

ALOG logical name to include that user's directory. Note that access to the catalog is through the Storage Class, created below.

3. Create an ABS Storage Class for the user, identifying the Owner as the user, and granting the user full access via the ACL. Specify the correct pool and catalog, as created above. Deny access to other users.
4. Create an ABS Execution Environment for the user, or let it default to the DEFAULT_ENV object provided by the ABS Installation Kit. The User Profile in the Environment must indicate the keyword "<REQUESTER>" in the user profile as the user under which the backup operations are performed.
5. Notify the user of their Storage Class Name. This Storage Class should be included on all of the user's ABS Save commands using the /STORAGE qualifier.

J.4.3 Media and Device Management

This section describes differences in how SLS and ABS handle media and device management.

J.4.3.1 New Media Manager

ABS uses a new Media And Device Management Services (MDMS) component that supports the concept of a domain. An MDMS domain has scope across multiple geographical locations each with their own nodes, jukeboxes, drives and volumes. MDMS utilizes a network-wide database that supports the following types of objects:

Domain - the entire scope of operations covered by a single MDMS database and managed as a single environment, networked together by a choice of protocols

Drive - a device that can read or write data to/from tape volumes, and that may reside in a jukebox

Group - a group of nodes that have some kind of relationship; the group name can be used as a convenience for multiple node names in a variety of contexts (e.g. OpenVMS clusters)

Jukebox - a device that performs random-access loading and unloading of volumes into drives

Location - a physical location that contains nodes, jukeboxes and volumes, which can be configured in a hierarchy and may contain spaces for volume and magazine storage

Magazine - a collection of volumes in a physical magazine that are moved as a group

Media Type - a logical description of the type of media of a volume, including attributes such as density, compaction and length

Node - an OpenVMS system running MDMS and ABS

Pool - a collection of volumes that may be allocated by authorized users

Volume - a piece of tape media that ABS uses to backup and restore customer data

Communication within the domain utilizes TCP/IP, DECnet Phase IV and/or DECnet-Plus at the user's choice. A Java-based GUI is also provided for MDMS operations, and this runs on Alpha VMS and Windows platforms. A comprehensive, consistent DCL syntax and the GUI replace the STORAGE commands and the forms interface provided with SLS; all useful functions can be performed from either interface. All database changes can be applied dynamically without the need to restart MDMS, and are applicable in all parts of the domain.

This is substantially different from the media management supplied by SLS. In that product, many definitions are stored in a configuration file called TAPESTART.COM. Not only was there a tendancy for this file to vary across nodes, but any change to the configuration required SLS to be restarted. Other definitions, such as volumes, magazines and slots were in bona-fide databases, but access to the database was inconsistent and incomplete. For example, many of the volume's attributes could not be modified using standard commands. In addition, the DCL and forms interfaces, while overlapping, were not complete in their own right; certain operations

could only be performed using one of the interfaces, meaning that the user/operator probably had to learn and use both of them.

One other substantial difference with MDMS is that it utilizes no device-specific code; MDMS attempts to perform operations on devices and if there are errors takes corrective action as needed, transparent to the user. One great advantage of this approach is that new devices are automatically supported, rather than having to wait for a software upgrade.

When upgrading from SLS to ABS, it is first necessary to convert the SLS volume, magazine and slot databases, together with the TAPESTART.COM definitions, to MDMS databases. A utility is provided for this purpose. This utility, together with a full description of how the media manager in SLS and MDMS are different, are described in the Appendix "Converting SLS/MDMS V2 to MDMS V3". MDMS has been designed so that the conversion can be performed as a rolling upgrade, starting with the set of nodes designated as database servers. These MDMS V3 database servers can support V2.x clients running ABS or SLS.

There are also a couple of significant differences between the way SLS and ABS handle volume management:

- Management of Volume Sets
- Consistency of Volume and Drive Management

J.4.3.2 Volume Set Management

Volume Sets are a collection of tape volumes that are treated as a single entity. The volumes are logically appended to one another, allowing more data to be stored and wasting less tape.

SLS only provides very rudimentary volume set management in the form of the CONTINUE symbol. Any SBK file which uses a consistent value of the CONTINUE symbol will have its savesets appended to a volume set managed by SLS. However, the user must explicitly name the volume sets (i.e. the value of the CONTINUE symbol), and creating new volume sets is problematic.

In ABS, data is written to volume sets automatically. Each Storage Class has one or more volume sets which it manages. The number of volume sets managed by a Storage Class is called the Number of Streams, or Number of Simultaneous Read and Write Operations.

ABS automatically creates new volume sets based upon the Storage Class's Consolidation Criteria. The Consolidation Criteria determines how data is consolidated onto volume sets. There are three criteria which can be used to limit the amount of data written by ABS to a volume set:

- Consolidation Interval - the number of days data will be appended to the volume set
- Consolidation Size - the maximum number of volumes in the volume set
- Consolidation Count - the maximum number of savesets to append to the volume set

Once ABS determines the consolidation criteria has been exceeded on a volume set, it automatically "retires" the volume set. Retiring a volume set allows data to be restored from it, but no more data is written to the volume set. ABS then automatically creates a new volume set for backing up data in the Storage Class.

J.4.3.3 Consistency of Volume and Drive Management

SLS is very inconsistent in its management of volumes and drives. For example, SLS System Backups do a different style of tape load than SLS User Backups, and the source code is completely different. In addition, the way that tapes are appended to volume sets, the messages indicating problems, and the methods for debugging problems in these areas are completely inconsistent.

ABS does all volume and drive management via MDMS through the ABS Coordinator. All types of data movements, from Selective to Full, Saves and Restores, and all types of Backup Agents are managed by the ABS Coordinator. This means that volume, robot and drive management are completely consistent across all operations.

J.4.4 Cataloging

This section identifies similarities and differences between how SLS and ABS handle cataloging operations. Catalogs (called History Sets in SLS) record what backup operations have taken place, and what files or other data objects have been backed up.

J.4.4.1 SLS History Sets

SLS History Sets come in two varieties: System History Sets and User History Sets. System History Sets are updated for system backups (i.e. SBK files), while User History Sets are updated for user backups.

Creating and configuring history sets is troublesome on SLS. The TAPESTART.COM command procedure is used to determine what System History Sets are created, and where they are located. The User History Sets are created on a per user basis, using the ASNUSRBAK.COM command procedure. Configuring User Histories depends upon the user executing the SLS$TAPSYMBOLS.COM procedure in their LOGIN.COM.

Although the information stored in both System and User History Sets are similar, the source code to write to them, look up files and restore files is totally different. This is a problem for maintenance, and prevents consistency in the data available for data backed up by SLS.

System History Sets are “staged”, which means that during the backup operation, the history records are written to a sequential temporary file. This improves the performance of the backups. Then, at a later time, the temporary files are loaded into the actual System History Sets, which can then be used to restore files. User History Sets are never staged.

SLS History Sets contain no protection information. The entire history set must be protected against individual users reading and restoring data. This means that at most sites, restoring data from system backups must be done by the Operator or System Administrator to prevent users from restoring data they would not normally have access to.

J.4.4.2 ABS Catalogs

ABS has only one catalog format, called a Brief catalog. This catalog provides basic information about what backup operations have been done, and what files or other data objects have been backed up. All ABS Catalogs have the same format, and are written by the ABS Coordinator, so information is consistent across all backups.

Creating catalogs in ABS is done by running the ABS$SYSTEM:ABS_CATALOG_OBJECT utility. Using this utility, you can create new catalogs with any name and owner. The utility also allows you to specify whether the catalog supports staging or not. Staging is where the catalog records are written to a temporary sequential file during the backup to improve performance, and then loaded into the actual catalog at a later time.

By default, ABS stores all catalogs in a single location on each system where backups are performed. This location is referenced by the ABS$CATALOG logical name. If you want to place the catalogs in other locations, you can move the catalog files to another directory, and redefine the ABS$CATALOG logical name.

J.4.4.3 Restoring data with ABS from SLS History Sets

ABS provides the ability to perform lookups and selective restores using SLS System History Sets. SLS System History Sets are written by SLS system backup operations to locate the tapes

and savesets containing backed up data. This ability is an important function for any conversion plan, since the data backed up previously by SLS may be needed even after ABS has been fully deployed.

ABS cannot perform full disk restores or Oracle RDB Database restores using SLS history sets. It can restore VMS files selected using a wildcard file specification.

The steps for restoring data using ABS with SLS History Sets are:

1. Create an ABS Catalog with the type of SLS. This is done using the ABS_CATALOG_OBJECT utility, as follows:

   ```
   $ Catalog_obj := $ABS$SYSTEM:ABS_CATALOG_OBJECT.EXE
   $ Catalog_obj  Create <hisnam> SLS ABS NO
   ```

   where <hisnam> is the name of the SLS history set from TAPESTART.COM. This will also be the name of the ABS catalog.

2. Create a read only Storage Class in ABS which references the SLS type catalog. The Storage Class should be read only, since SLS History Sets can only be used for restores.

3. Issue ABS Lookup or ABS Restore commands using the Storage Class created above. The SLS History Set will be used for the lookup or restore.

Note

You must have ABS_LOOKUP_ALL access right granted to your account to display entries in the SLS History Files from ABS.

J.5 Conversion Process

This chapter identifies the Conversion Process from SLS to ABS. First, the steps involved in converting from SLS to ABS are presented and explained. A Conversion Utility to help with the conversion is then presented.

J.5.1 Steps for Conversion

This section identifies the steps involved in converting a site's backup management from SLS to ABS. These steps are intended as guidelines, since each site has different requirements and need for their backup management.

J.5.1.1 Convert the MDMS Database

The first step in converting from SLS to ABS is to convert the volume, slot and magazine databases, and the media and device portions of TAPESTART.COM to MDMS databases. A command procedure is provided for this purpose and this procedure is documented in the Appendix "Converting SLS/MDMS V2 to MDMS V3". Please note that this version of ABS and all future versions require the accompanying version of MDMS included in the installation kit.

J.5.1.2 Determine your use of SLS

The next step in converting from SLS to ABS is to identify how you use SLS. There are three major uses of SLS:

- System Backups
- Standby Archiving
- User Backups

ABS provides the same functionality as SLS System Backups and SLS User Backups. However, ABS cannot perform the same function as SLS Standby Archiving. SLS Standby Archiving has the following characteristics:

- "Archive Classes" are created by the System Administrator
- When users request a Save, they indicate the Archive Class on their Save request, but the request is only queued, it is not executed immediately or scheduled.
- At some point in time, the system Operator starts a "standby archive session" for each Archive Class
- SLS then mounts the tape associated with the Archive Class, and appends any requests queued since the last session to the tape.
- Until the Operator stops the session, the tape sits on the drive waiting for more requests to come in.
- When the Operator stops the session, the tape is dismounted and requests go back to being queued.

If you use SLS System Backups (as many sites do), then converting to ABS is fairly simple. If you use SLS User Backups, converting to ABS is slightly more involved, but is still fairly straightforward. If you use SLS Standby Archiving, ABS will not provide equivalent functionality.

J.5.1.3 Converting SLS System Backups to ABS

This section describes how to convert SLS System Backups (SBK files) into ABS Policy.

Determine valid SBK files

At many sites, only a few of the SBK files which reside in SLS$SYSBAK are actually used. The other SBK files are a result of experimentation, false starts at configuring SLS, or are obsolete.

In order to simplify the conversion of SLS System Backups, you should first identify the SBK files which you actually use. Usually, SBK Files are used at your site if:

- They are scheduled regularly by SLS
- They are manually executed by you or the Operator

A simple way of finding the SBK files which are scheduled by SLS is to search the SBK files for the DAYS_1 symbol. Any SBK file which does not define DAYS_1, or defines it as blank, is not scheduled by SLS for automatic execution. These SBK files are prime candidates for obsolete or unused files.

After identifying the SBK files which are not automatically scheduled, carefully determine which of the files may be invoked manually by you or the Operator.

Once you have identified the obsolete or unused SBK files, you can remove them from SLS$SYSBAK (after backing them up in case of mistakes, of course).

Convert the Valid SBK Files to ABS Policy

Once you have cleaned up your SLS$SYSBAK directory to only contain those SBK files you actually use, it is time to convert these SBK files to ABS Storage Classes, Execution Environments and Save Requests.

Compaq provides a utility to help in this conversion process. The conversion utility is called SLS_CONVERT.COM, and is included as an installation kit on the ABS Kit. To install the SLS to ABS Conversion utility, issue the command:

```
$ @SYS$UPDATE:VMSINSTAL   SLSTOVABS031    ABS$SYSTEM:! VAX System OR
$ @SYS$UPDATE:VMSINSTAL   SLSTOAABS031    ABS$SYSTEM:! Alpha System
```

This command will install the conversion utility into ABS$SYSTEM:SLS_CONVERT.COM, and will create a subdirectory under the ABS$ROOT called SLS_CONVERSION. In addition, a logical name, ABS$SLS_CONVERSION will be defined to point to the work directory for the conversion effort.

The conversion utility creates DCL Command Procedures which issues appropriate ABS DCL commands equivalent to each SBK file. No changes are made to your ABS Policy Configuration directly. This allows you to experiment with the conversion utility safely, without affecting either the execution of your SLS SBK files, or starting ABS Save Requests inadvertently.

Once you have installed the conversion utility, you can create ABS command procedures for all of your SBK files by issuing the command:

```
$ @ABS$SYSTEM:SLS_CONVERT   *
```

The asterisk indicates you want to convert all SBK files to ABS DCL Commands. If you only want to convert one SBK file, you can specify the name of the SBK without the _SBK.COM or SLS$SYSBAK on the command line. For example, to convert NIGHTLY_SBK.COM, you would issue the command:

```
$ @ABS$SYSTEM:SLS_CONVERT NIGHTLY
```

Evaluate the ABS Conversion Command Files

After running the conversion utility, the ABS$SLS_CONVERSION directory will contain one ABS DCL Command Procedure for each SBK file converted. These output command files will contain:

- Comments explaining the conversion process
- ABS DCL Commands to create ABS Policy Objects
- A Prolog and Epilog command to complete the functions performed by the SBK file

You should not execute these command procedures blindly. The conversion utility attempts to duplicate the backup policy reflected in each SBK file, but you should carefully examine each command file produced to ensure that errors were not made, and that the ABS Policy to be created correctly reflects the backup policy you expect.

The things you should check for in the produced command procedures are:

- Naming conventions used in the conversion may not be what you want
- Errors in converting the SBK policy
- Possible ABS Policy Consolidation

The command procedure for each SBK file processed will contain the ABS DCL Commands to create one Storage Class, one Execution Environment and one or more Save Requests.

Naming Conventions Used

The name of the Storage Class created for an SBK file will be the value of the CONTINUE symbol (if defined) followed by the suffix “_SC”. If the CONTINUE symbol is not defined, the Storage Class will be named the same as the SBK file with the “_SC” suffix.

The Environment created for an SBK will be named the same as the SBK file, but with an “_ENV” suffix. When a Save Request specifies a Storage Class, the default Environment used will be the same name as the Storage Class, but with the “_ENV” suffix. Thus, the Environment created should be used by default.

Each SBK File will produce one or more ABS Save Requests. More than one ABS Save Request will be produced from an SBK file if all the following conditions are met:

- There are more than one FILES_n is specified
- The QUALIFIERS_n differ in the type of operation
- More than eight include specifications are found in a single SBK file

For example, if an SBK file has QUALIFIER_1 defined as "/IM" indicating an Image (or Full) backup operation, but QUALIFIERS_2 is defined as "/SINCE=BACKUP" indicating an Incremental operation, then ABS will need two separate Save Requests to implement this policy. This is because an ABS Save Request will only do Full, Incremental or Selective operations, not a mix of them.

If all QUALIFIERS_n specify the same movement type and there are fewer than eight FILES_n, then ABS can combine all the operations into a single Save Request.

The Save Requests created will be named the same as the SBK file, but with "_FULL", "_INC" or "_SEL" to indicate the data movement type included in the Save Request. For example, if the SBK file NIGHTLY_SBK.COM defines FILES_1 through FILES_20, and all qualifiers include the "/IM" Image qualifier, then the conversion tool will create three Save Requests, called NIGHTLY_FULL_1 through NIGHTLY_FULL_3. Because ABS has a limit of 8 operations per save request, NIGHTLY_FULL_1 and NIGHTLY_FULL_2 would perform 8 Full backup operations, and NIGHTLY_FULL_3 would perform the last four.

Consolidate ABS Policy Objects

The Conversion Utility shipped with ABS is a very simple utility. It converts each SBK file into the appropriate ABS DCL Commands to create the Storage Classes, Execution Environments and Save Requests necessary to reflect the backup policy in the SBK file.

No attempt is made to consolidate the Storage Classes and Execution Environments, or to overlay the Save Requests for more optimum performance.

Before executing the command procedures to create the ABS Policy objects, you should try to consolidate Storage Classes and Execution Environments. Save Requests may be combined if warranted by the intended policy, but in some cases, breaking a Save Request into several is better for reducing nightly backup time, simplifying an overall backup policy, or backing up different objects at different intervals.

Consolidating Storage Classes

Consolidating the Storage Classes is done by comparing their parameters. For each pair of Storage Classes, you can determine whether they can be combined by using the Table J–7. Note that in all cases, you can decide that one or the other parameter is correct for *both*, and consolidate based upon that decision.

Table J–7 Storage Class Parameter

Storage Class Parameter	Matching Criteria
Name	Doesn't matter choose meaningful name
Media Type	Should match
Tape Pool	Should match
Media Location	Should match

Access Control	OK if different, choose best for intended Storage Class use
Owner	Should both be ABS
Retention	OK if different, choose best for intended Storage Class use
Volume Set Name	Not set, doesn't matter
Consolidation Criteria	OK if different, choose best for intended Storage Class use
Catalog name	OK if different
Number of Streams	Always set to 1 from Conversion utility
Execution Node	Should match
Drive name	OK if different, choose best for intended Storage Class use

Only the Administrator at a site can truly determine if two separate Storage Classes can be consolidated based upon the intended use of the Storage Class.

Consolidating Execution Environments

Consolidating Execution Environments is again done by comparing the parameters of pairs of Environments, and then combining those Environments if your decisions indicate they can serve the same purpose. Use Table J–8 as a guide:

Table J–8 Execution Environment Parameter

Environment Parameter	Matching Criteria
Name	Doesn't Matter choose meaningful name
Data Safety options	OK if different choose best for intended Environment's use
Listing options	OK if different Digital recommends *not* producing listings
Span FileSystems	Should Match
Links Only	Should Match
Original object action	Should Match
User Profile	Will always be ABS from conversion utility, choose PRIVILEGES best for intended Environment's use
Notification	OK if different choose best for intended Environment's use
Locking options	Should match
Number of drives	OK if different choose best for intended Environment's use
Retry options	OK if different choose best for intended Environment's use
Prolog and Epilog	Should match, or be combined

As with Storage Classes, only the Administrator at a site can truly determine if two separate Environments can be consolidated based upon the intended use of the Environment.

Implement ABS Policy

This section discusses the steps involved in implementing the ABS Policy as produced by the conversion utility and evaluated by the site Administrator.

Executing the Command Procedures –After you have examined the raw output command files from the conversion utility and done what consolidation or modifications seem appropriate, the command files can simply be executed using the at sign (@) operator at DCL. When each command procedure is invoked, it will:

- Create a Storage Class
- Create an Execution Environment
- Create one or more Save Requests and associated Scheduler jobs
- Possibly create a Prolog command procedure
- Possibly create an Epilog command procedure

Integrating the Prolog and Epilog Commands –There are several features of an SBK file which are not directly supported by ABS. The conversion utility creates a Prolog command file and an Epilog command file which implement some of these other features.

For example, ABS does not support the Offsite Date or Onsite Date in the SBK file directly. However, by issuing the appropriate MDMS SET VOLUME command, this can be implemented. The conversion utility writes these commands into the Prolog or Epilog command files.

When the conversion utility produces a Prolog and Epilog command procedure, they will be created in the ABS$SLS_CONVERSION directory, and will be called, the same name as the Save Request, but will have "_PROLOG" or "_EPILOG" appended. For example, if you convert the NIGHTLY_SBK.COM, you will end up with the Prolog and Epilog command files:

```
ABS$SLS_CONVERSION:NIGHTLY_ABS_PROLOG.COM
ABS$SLS_CONVERSION:NIGHTLY_ABS_EPILOG.COM
```

If you need the features implemented in the Prolog or Epilog command procedures, you should integrate these into your own Prolog and Epilog command procedures (if any). Both the Execution Environment and the Save Request may have Prolog and Epilog commands associated with them, which are usually the execution of a site specific command procedure. If you want the features implemented in the Prolog or Epilog command procedures produced by the conversion utility, you should invoke them from your site specific command procedure.

SBK Symbols and ABS Logicals

When executing an SBK file, SLS makes various DCL symbols accessible to the prolog and epilog command files you invoke. For example, SLS will define the DCL symbol DO_DISK as the name of the disk being backed up during an SBK execution. The objective is to allow the prolog or epilog commands to produce log messages, or perform other operations based upon the SBK file execution.

ABS provides logical names which provide similar functionality. For example, ABS defines the logical name ABS_OS_OBJECT_SET_1 as the set of files being backed up in the first data movement operation. Thus, it can be used in the place of the FILES_n symbol in an SBK file.

The conversion utility kit provides a command procedure, SLS_SYMBOLS.COM, which attempts to define many of the same DCL symbols as an SBK file does based upon the ABS logical names. For example, it defines the DO_DISK symbol based upon the ABS logical name ABS_OS_OBJECT_SET_1.

See ABS$SLS_CONVERSION:SBK_SYMBOLS.COM command procedure for details on the definition of each SBK symbol. Not all SLS DCL symbols defined are supported by the command procedure.

Disable the SLS SBK Files –It is very important to note that once you have executed the DCL Command procedures produced by the conversion utility, the ABS Save Requests will be executing according to their schedules. This means that you will be doing *both* SLS and ABS backups if you do not disable the SLS SBK files.

The SLS SBK files can be disabled by changing their DAYS_n and TIME_n qualifiers to empty. This causes SLS to no longer schedule the SBK files for execution.

Since SLS and ABS use different media management subsystems, it is highly recommended that you do not use both products on the same node. If you do, you will find that the SLS and MDMS volume databases may get out of synchronization, and there may be contention and other unexpected troubles with drives and jukeboxes. If you wish to stage your SLS to ABS conversion across your network, the following approach is recommended:

Define your Central Security Domain as your first set of nodes to convert; these nodes will run the ABS policy engine and the MDMS database server

Perform the MDMS conversion on these nodes - see Appendix "Converting SLS/MDMS V2 to V3".

Perform the ABS conversion on these nodes

On other client nodes still running SLS, define the appropriate TAPESTART.COM to point to nodes in the ABS/MSMS Central Security Domain in the symbol DB_NODES

At this point, your volume, magazine and slot databases are being managed by MDMS, but your client systems are still able to use SLS as the backup paradigm. It is recommended that you convert the remainder of your systems to ABS/MDMS as soon as practical, because some of the more unusual features of SLS/MDMS are not supported by the new MDMS database server.

J.5.1.4 Converting User Backup policy

The conversion utility does not convert User Backup policy automatically. It is only intended to make converting SBK files easier or automatic.

To allow a particular user to do their own backups, follow the steps as outlined in Section . Note that there is no automatic way to set up Storage Classes for the entire user population, or a large set of users except by creating a DCL command procedure issuing the correct ABS DCL commands.

J.5.1.5 Monitor ABS Activity

After implementing your backup policy in ABS, you should carefully monitor the activities of ABS until you are confident that your policy is being executed as intended.

There are three ways to monitor ABS activity:

- Use the appropriate show command for the current scheduler interface option to list scheduled requests.
- Set up Notification criteria on the Execution Environments to send you mail when ABS operations complete. The mail will contain the name of the job and the final status.
- Examine the ABS Log files. All ABS Log files are created in the ABS$LOG: directory, and are called the same name as the Save Request.
- For catalog operations:
- Monitor the Staging Log files

  - These are named ABS$LOG:<catalog_name>_<stream>.LOG
- Monitor the Catalog cleanup log files
  - These are named ABS$LOG:ABS_CLEAN_CATLG_<node>.LOG
- Monitor the Policy Engine Log files on your Central Security Domain (CSD)
  - ABS$ROOT:[000000]ABS$START_POLICY_ENGINE.LOG
  - ABS$LOG:ABS_CLEAN_DB_UTIL.LOG

J.5.1.6 Restoring from SLS History Sets

ABS has the ability to restore data backed up by SLS. After you have implemented your backup policy using ABS, it may be necessary to restore data which was backed up using SLS prior to the conversion. Please see Section for more information on this capability.

J.6 Conversion Utility Reference

J.6.1 Command Syntax

```
$ @ABS$SYSTEM:SLS_CONVERT  <wildcard_SBK_spec>  [<match1>] [<match2>…]
```

<wildcard_SBK_spec>

This parameter identifies the set of SBK files to be converted by this command. The string given should *not* include SLS$SYSBAK: or the _SBK.COM suffix. For example, if you want to convert the SBK file SLS$SYSBAK:NIGHTLY*_SBK.COM, you should issue the command:

```
$ @ABS$SYSTEM:SLS_CONVERT NIGHTLY*
```

<match1> … <match7>

These optional parameters allow you to search the SBK files defined by the <wildcarded_SBK_spec> and only process those files which contain ALL of the given strings. The strings must all appear on the same line in the SBK file, since the SLS_CONVERT command procedure uses a /MATCH=AND on the Search command.

J.6.2 Output Command File naming and contents

The output of the SLS_CONVERT conversion utility is one DCL command procedure for each SBK file processed. The command procedures will be created in the ABS$SLS_CONVERSION directory.

Each command procedure will be named the same as the SBK file, but substituting “ABS” for “SBK”. For example, if the SBK file SYSTEM_DISK_SBK.COM is converted, the output command procedure will be ABS$SLS_CONVERSION:SYSTEM_DISK_ABS.COM.

Although the command procedures can be executed immediately, it is highly recommended that you review their contents before executing them to ensure that the ABS Policy objects which will be created accurately reflect your intended backup policy.

Each output command file will contain:

- A block of comments indicating that the file was produced by the conversion utility, and the date and time of the conversion.
- The name of the SBK file represented in the command file
- The list of SBK parameters which are not handled by the conversion utility, and the reason they are not.

- An ABS Create Storage command to create a Storage Class.
- An ABS Create Environment command to create an Execution Environment.
- One or more ABS Save commands and ABS Set Save commands to create Save Requests. See Section J.5.1.2 for information on why a single SBK file might produce multiple Save Requests.
- The creation of a Prolog command file. The Prolog command file should be integrated with any site specific prolog command files to complete the functions defined by the SBK.
- The creation of an Epilog command file. The Epilog command file should be integrated with any site specific epilog command files to complete the functions defined by the SBK.

J.7 SBK Symbols in ABS Terminology

Table J–9 SBK Symbols in ABS Terminology

Table J–9, lists SBK Symbols in ABS Terminology.

SBK Symbol	ABS Equivalent	Meaning
DAYS_n	Save Request /SCHEDULE and /EXPLICIT	Defines how often the backup oper ations are performed
TIME_n	Save Request /START_TIME	Defines when the backup operation starts
NODE_n	Save Request /SOURCE_NODE	Defines the node in your network where the data resides
BACKUP_TYPE	Save Request /OBJECT_TYPE	Defines the type of data to be backed up or restored.
PRE_PROCESS_FIRST	Environment /PROLOG	Defines a command to be executed when the backup job starts
PRE_PROCESS_EACH	Save Request /PROLOG	Defines a command to be executed prior to each backup operation within a job
POST_PROCESS_EACH	Save Request /EPILOG	Defines a command to be executed when each operation within a job finishes
POST_PROCESS_LAST	Environment /EPILOG	Defines a command to be executed when the backup job completes.
NEXT_JOB	Use dependencies in current scheduler interface option if available	Defines what job to run next after the current job completes
SUMMARY_FILE	ABS REPORT SAVE/FULL and ABS Catalogs	Gives overview information about a save operation in a job.
PRIVS	Environment /PRO FILE=(PRIVS)	Defines the set of privileges to be used when executing the operation
FILES_n	Save Request Include Specifi cation	Defines the set of files or other data objects to be backed up or restored.

QUALIFIERS and QUALIFIERS_n	Environment /ACTION, /LOCK and /DATA_SAFETY Save Request /FULL/SINCE=BACKUP /SELECT	Defines characteristics of the save operation, such as the type of data movement, and options for execu tion of the backup.
MNTFLAGS	Not supported. ABS controls mounting of tapes.	Defines how tapes are mounted.
SAVESET_GEN	Not supported. ABS gener-ates the saveset names.	Defines the name of the saveset stored on tape.
PROTECTION	Storage Class /ACCESS	Defines what access is available to the backed up data.
MEDIA_TYPE	Storage Class /TYPE_OF_MEDIA	Defines which MDMS media type is to be used for backup operations.
DENSITY	Density is an attribute of the MDMS media type object.	Defines the tape density to be used for backup operations.
REEL_SIZE	This maps to the length attribute of the MDMS media type object.	For 9 track tapes, defines the length of the tape (e.g. 2400 feet)
TAPE_POOL	Storage Class /TAPE_POOL	Defines the MDMS pool from which tapes are drawn for backup operations.
QUICKLOAD	The MDMS drive attribute "AUTO_REPLY" can be specified on a per-drive basis to determine whether a drive comes on line.	Determines whether MDMS will automatically recognize when a tape drive comes online without opera tor intervention.
QUICKLOAD_RETRIES	Not supported	Defines how long a LOAD request should remain outstanding before being canceled.
PREALLOC	ABS allocates and manages volume sets automatically.	Determines the number of volumes to pre allocate before a backup begins, forming them into a volume set.
AUTOSEL	ABS always automatically selects new volumes to append to volume sets, if needed.	Determines whether SLS is allowed to automatically select new volumes from the volume database if needed.
CONTLOADOPT	Logical Name: ABS$DISABLE_SCRATCH_ LOADS set to one. By default, ABS will request and accept scratch tapes. The logical can be defined to force specific tapes to be mounted.	Determines whether the operator can substitute a valid tape for the requested tape.
UNATTENDED_BACKUP S	ABS always attempts to per form the backup without oper ator intervention.	Determines whether SYSBAK should default responses to ques tions rather than require operator intervention.

CONTINUE	ABS Storage Class Name. Each Storage Class manages one or more volume sets, and appends data to these volume sets until the Consolidation Criteria is exceeded.	Determines how data is consoli dated onto volume sets.
HISTORY_SET	Catalog Name Storage Class /CATALOG	Determines the catalog into which a record of the operations performed and the files backed up is written.
SBUPDT_Q	Not supported. If a catalog supports staging, ABS always performs the catalog update in a detached process.	Determines the Batch Queue in which the System history set update is performed.
SCRATCH_DAYS	Storage Class /RETAIN	Determines how long data is saved before the tapes are recycled and catalog entries removed.
OFFSITE_DATE ONSITE_DATE	MDMS volumes support OFFSITE_DATE and ONSITE_DATE attributes, and will automatically gener-ate MOVE VOLUME com-mands when these dates are reached.	Determines when the volume sets are moved offsite or onsite (i.e. vaulting).
TAPE_LABELS	Not supported.	Determines if paper labels are printed for the volumes used in the backup
NOTES	The MDMS description field in the volume object is the equivalent attribute.	Store a free form text note in the volume record for the volumes used in the backup.
DRIVE_TYPE	Storage Class /DRIVE_LIST	Determines the list of tape drives to be used. It is recommended that MDMS media types be set up cor rectly rather than using this field.
N_DRIVES	Environment /DRIVE_COUNT	Determines the number of tape drives to use during a backup opera tion.
PROGRESS	Not supported	Notifies the operator after a certain number of files have been backed up.
REPLY_MSG	Not supported. MDMS issues all OPCOM messages in a standard format	Determines the notification to be performed when each backup opera tion starts and ends.
STATUS_MAIL	Environment /NOTIFY	Determines who should be sent MAIL when the job completes.
LOG_FILE	Not supported. ABS gener-ates a log file in ABS$LOG with the same name as the Save Request.	Determines the name of the log file for the operation.

LISTING_GEN	Environment /LISTING. ABS will generate listing files, but they are always located in ABS$LISTINGS, and are named the same as the Save Request and the operation number.	Determines the name of a backup listing file to be produced from each operation.
FULL	Environment /LIST ING=FULL	Determines if the listing file pro vides all information about backed up files, or only brief information.
PRINT_Q	Not supported.	Determines the Print queue on which the listing file is printed.

J.8 ABS Policy Attributes in SBK Terminology

Table J–10, lists ABS Storage Class object parameters and their SLS SBK Equivalents.

Table J–10 ABS Storage Classes and SLS SBK Equivalent

Storage Class Parameter	SBK Equivalent	Meaning
Name	CONTINUE	A common name which can be referenced by multiple Save Requests
Primary Archive Loca tion	<None>	For on disk backups, this gives the disk and top level directory where the savesets will be stored.
Primary Archive Type	<None>	This determines whether the Storage Class is tape based (type is MDMS) or disk based (type is FILES-11)
Owner	<None>	Determines the NODE::USER of the owner of the Storage Class. The owner always has CON TROL access.
ACL	PROTECTION	Determines the access to the backed up data. ABS provides full ACL based access, while SLS provides only OpenVMS-style system, owner, group and world access.
Tape Pool	TAPE_POOL	The MDMS pool from which volumes will be allocated for backups.
Type of Media	MEDIA_TYPE	The MDMS media type to be allocated for back ups.
Retain	SCRATCH_DAYS	The number of days the backed up data will be saved before the tapes are recycled. Note that a Save Request can specify a retention *shorter or equal to* the value in the Storage Class.
Consolidation Criteria	CONTINUE	This set of parameters determines how backup savesets will be consolidated onto tapes. For example, if the Consolidation Interval is set to 7 days, savesets will be appended onto a volume set for 7 days before a new volume set is created.
Catalog Name	HISTORY_SET	The name of the catalog which stores data about what files have been backed up, and where they are located.
Number of Streams	<None>	The number of simultaneous Save requests which can be writing into the Storage Class. Determines the number of MDMS volume sets simultaneously active in the Storage Class.
Media Location	<None>	The MDMS onsite location field to match for allocat ing volumes for backups.
Drive List	DRIVE_TYPE	The list of specific drives to be used for backup operations in this Storage Class. Normally, this should be managed through MDMS drive objects.

Table J–11, lists ABS Execution Environment parameters and their SLS SBK Equivalents

Table J–11 ABS Execution Environment Parameter and SLS SBK Equivalent

Environment Parameter	SBK Equivalent	Meaning
Name	<None>	Identifies the Environment for reference by Save Requests
Owner and ACL	<None>	Identifies the owner and access to the Environment.
Data Safety Options	QUALIFIERS	A bitmask containing the data safety options to be applied during the backup. Data safety options include CRC checking, and full data verification.
Listing Options	LISTING_GEN FULL	Determines whether a listing file is produced, and whether the listing is a “full” listing or a “brief” list ing.
Span Filesystem Options	<None>	For UNIX file systems, determines whether the entire filesystem is backed up, even if it crosses multiple physical devices.
Links Only	<None>	For UNIX file systems, determines whether ABS backs up only the logical links, or backs up the data as well.
Compression Options	<None>	For UNIX file systems, determines the type of com pression to be applied to the savesets.
Original Object Action	QUALIFIERS	Determines the action to be taken on the original data objects (e.g. the files backed up). Options include None, Record Backup Date, or Delete
User Profile	PRIVS	Determines the username, privileges and access rights used during the backup operation. The special keyword “<REQUESTER>” indicates the backup operations should be performed with the username, privileges and access rights of the person issuing the ABS Save Command.
Notification Criteria	REPLY_MSG STATUS_MAIL	Determines when notification occurs, what method is used, and who is notified.
Locking Options	QUALIFIERS	Determines how much interlocking is done between the backup in progress and an active file system. Options include Ignore File Writers, and Hot Backup.
Number of Drives	N_DRIVES	Determines the number of tape drives to be used during the backup operations.
Retry Interval and Count	<None>	Determines how often and how many times a failed backup should be retried.
Prolog Command	PRE_PROCESS_FI RST	A command to be executed when the backup starts. Contrast to Save Request Prolog.
Epilog Command	POST_PROCESS_ LAST	A command to be executed when the backup com pletes. Contrast to Save Request Epilog.

Table J–12, lists ABS Save Request parameters and their SLS SBK equivalents.

Table J–12 ABS Save Request Parameter and SLS SBK Equivalent

Save Request Parameter	SBK Equivalent	Meaning
Name	SBK File name	Identifies the group of backup operations to be per formed.
Movement Type	QUALIFIERS	Determines whether the operations are Full, Incre mental or Selective (i.e. individual file) operations.
Source Node	NODE_n	Node on which the data resides.
Include Specification	FILES_n	Identifies the data to be backed up. Multiple include specifications can be given on a single Save Request, and each can have a different Object Type.
Object Type	BACKUP_TYPE	Gives the type of the data. ABS Supports many dif ferent types of data, including OpenVMS Files, UNIX Files, Oracle RDB Databases, and so forth.
Agent Qualifiers	QUALIFIERS	Allows backup agent specific qualifiers to be added to the command used to backup the data.
Since and Before Date	QUALIFIERS	Determine whether data objects to be backed up should be selected based upon creation/modifica tion date.
Exclude Specification	QUALIFIERS	Determines selected data objects to be excluded from the backup.
Storage Class Name	<None>	Gives the name of the Storage Class into which the data is backed up.
Environment Name	<None>	Gives the name of the Execution Environment to be used for the backup operations.
Start Time	TIME_n	Indicates the time at which the Save Request should start each time it is scheduled. Note that an SBK can provide multiple DAYS_n and TIME_n parameters, an ABS Save Request is restricted to a single Start Time and Interval.
Scheduling Option and Explicit	DAYS_n	Identifies the repeat interval for the Save Request. ABS provides a variety of predefined simple inter vals, such as Daily, Weekly, Monthly, as well as several “complex” intervals, such as Weekly Full with Daily Incremental, and log based schedules. See the Appendix "Log-n Backup Schedules"ABS for a full description of log based schedules.
Prolog Command	PRE_PROCESS_E ACH	A command to be executed before each backup operation within the Save Request starts. Contrast to Environment Prolog.
Epilog Command	POST_PROCESS_ EACH	A command to be executed after each backup oper ation within the Save Request completes. Contrast to Environment Epilog.

K

Differences Between MDMS Version 2 and MDMS Version 3

This Appendix addresses differences between MDMS Version 2 and MDMS Version 3 (V3.0 and later). It describes differences in command syntax, software features replacing the MDMS User, Operator, and Administrator interfaces, and features replacing the TAPESTART.COM command procedure.

K.1 Comparing STORAGE and MDMS Commands

For MDMS version 3.0 and later, the MDMS command set replaces the STORAGE command set. Table K–1compares the STORAGE command set with MDMS commands.

Table K–1 Comparing MDMS Version 2 and Version 3 Commands

MDMS Version 2 Commands...	MDMS Version 3 Commands...
STORAGE ADD DRIVE	MDMS SET DRIVE/ENABLED
STORAGE ADD MAGAZINE	MDMS CREATE MAGAZINE
STORAGE ADD VOLUME	MDMS CREATE VOLUME
STORAGE APPEND	MDMS BIND VOLUME
STORAGE BIND	MDMS MOVE VOLUME
STORAGE CREATE LABEL	No equivalent feature
STORAGE EXPORT ACS	MDMS MOVE VOLUME
STORAGE EXPORT CARTRIDGE	MDMS MOVE VOLUME
STORAGE EXPORT MAGAZINE	MDMS MOVE MAGAZINE
STORAGE IMPORT ACS	MDMS MOVE VOLUME
STORAGE IMPORT CARTRIDGE	MDMS MOVE VOLUME
STORAGE IMPORT MAGAZINE	MDMS MOVE MAGAZINE
STORAGE INVENTORY ACS	MDMS INVENTORY JUKEBOX
STORAGE INVENTORY JUKEBOX	MDMS INVENTORY JUKEBOX
STORAGE LABEL	No equivalent feature
STORAGE LOAD	MDMS LOAD DRIVE MDMS LOAD VOLUME
STORAGE RELEASE	MDMS SET VOLUME /RELEASE

Table K–1 Comparing MDMS Version 2 and Version 3 Commands

STORAGE REMOVE DRIVE	MDMS SET DRIVE/DISABLED
STORAGE REMOVE MAGAZINE	MDMS DELETE MAGAZINE
STORAGE REMOVE VOLUME	MDMS DELETE VOLUME
STORAGE REPORT SLOT	MDMS REPORT VOLUME SPACES/SORT
STORAGE REPORT VOLUME	MDMS REPORT VOLUME
STORAGE SELECT	MDMS ALLOCATE DRIVE
STORAGE SET VOLUME	MDMS SET VOLUME
STORAGE SHOW JUKEBOX	MDMS SHOW JUKEBOX
STORAGE SHOW LAST_ALLOCATED	No equivalent feature
STORAGE SHOW MAGAZINE	MDMS SHOW MAGAZINE
STORAGE SHOW VOLUME	MDMS SHOW VOLUME
STORAGE SPLIT	MDMS UNBIND VOLUME
STORAGE UNBIND	MDMS MOVE VOLUME
STORAGE UNLOAD DRIVE	MDMS UNLOAD DRIVE
STORAGE UNLOAD VOLUME	MDMS UNLOAD VOLUME

K.2 MDMS V2 Forms Interface Options

The MDMS Version 2 forms interface provides features that are not found in the command set. This section compares the features of the three forms interfaces with MDMS Version 3 commands.

Table K–2 Comparing MDMS V2 Forms and MDMS V3 Features

MDMS Version 2 Forms Features...	MDMS Version 3 Features...
SLSUSER Menu COMMANDS Section Show Volume Deallocate Volume Modify Scratch Date Modify Volume Note DCL Storage Command	 MDMS SHOW VOLUME MDMS DEALLOCATE VOLUME MDMS SET VOLUME/SCRATCH_DATE MDMS SET VOLUME/DESCRIPTION MDMS commands
SLSUSER Menu REPORTS Section All Owned Volumes Volumes by Scratch Date	 MDMS REPORT VOLUME/USER MDMS REPORT VOLUME/FORECAST
SLSOPER Menu COMMANDS Section Release Volumes	 MDMS SET VOLUME/RELEASE
Update Clean Data Initialize Volumes DCL Storage Command Delete User Histories Tapejuke Initialize Volume	None MDMS INITIALIZE VOLUME All MDMS Commands None MDMS INITIALIZE VOLUME

Table K–2 Comparing MDMS V2 Forms and MDMS V3 Features

SLSOPER Menu MENUS Section Maintenance Option	
Add Volume	MDMS CREATE VOLUME
Add Volume Series	MDMS CREATE VOLUME
Remove Volume	MDMS DELETE VOLUME
Show Volume	MDMS SHOW VOLUME
Modify Volume	MDMS SET VOLUME
Modify Volume Series	MDMS SET VOLUME
Add Slot Definitions	MDMS SET LOCATION /SPACES
Remove Slot Definitions	MDMS SET LOCATION /SPACES
Generate Volume Report	MDMS REPORT VOLUME
SLSOPER Menu MENUS Section Vault Management Option	
Change to Onsite	MDMS MOVE VOLUME
Change to Offsite	MDMS MOVE VOLUME
Mass Movement	MDMS MOVE VOLUME
Change Onsite Date	MDMS SET VOLUME
Change Offsite Date	MDMS SET VOLUME
Volumes Offsite	MDMS REPORT VOLUME
Volumes to go Offsite	MDMS REPORT VOLUME
Volumes to come Onsite	MDMS REPORT VOLUME
Vault Profile Report	MDMS REPORT VOLUME
Change Name for Current Process (Vault)	MDMS SET VOLUME
SLSOPER Menu MENUS Section Standby Archive SYSCLN	None
SLSOPER Menu MENUS Section ACS Management Option	
Inventory Volume Series	MDMS INVENTORY JUKEBOX
Import Volume(s)	MDMS MOVE VOLUME
Initialize Volume Series	MDMS INITIALIZE VOLUME
Load Volume Onto Drive	MDMS LOAD VOLUME or DRIVE
Unload Drive	MDMS UNLOAD DRIVE
Unload Volume	MDMS UNLOAD VOLUME
Export Volume(s)	MDMS MOVE VOLUME
SLSOPER Menu REPORTS Section Free Volumes Allocated Volumes Down Volumes Volumes in Transition Volumes Due for Allocation Volumes Due for Cleaning Quantity Control	MDMS REPORT VOLUME
SLSOPER Menu MISC Section Repair Tape Jukebox Volume State	MDMS SET VOLUME
SLSMGR Menu Exit	None
Volume Pool Authorization Database Access Authorization HELP Screen for Keypad Definitions	MDMS CREATE or SET POOL MDMS Rights None

K.3 TAPESTART.COM Command Procedure

The command procedure TAPESTART.COM is no longer used. shows TAPESTART.COM symbols and the comparable features of the MDMS Version 3.

Table K–3 Comparison of TAPESTART.COM to MDMS Version 3 Features.

TAPESTART.COM Feature...	MDMS Version 3 Feature...
PRIMAST symbol	MDMS$SYSTARTUP.COM symbol MDMS$DATABASE_LOCATION
NET_REQUEST_TIMEOUT symbol	Domain object record network timeout attribute
NODE symbol	Node object record for each node
Media Triplet MTYPE_n symbol DENS_n symbol DRIVES_n symbol	Media type object record, name attribute, density attribute, Drive object record media types attribute
TAPE_JUKEBOXES symbol USER_DEFINED_NAME_n symbol (including the jukebox and drive device names)	All jukebox object records Jukebox object record name attribute, robot attribute, Drive object record jukebox attribute
MGRPRI symbol	Domain object record priority attribute
VERBOSE symbol	There is no equivalent feature
Software Privileges PRIV_SEEANY symbol PRIV_MODANY symbol PRIV_MAXSCR symbol PRIV_LABEL symbol PRIV_CLEAN symbol PRIV_MODOWN symbol	MDMS rights do not map directly. See Command Reference Guide for descriptions for setting MDMS rights.
Operator Terminal Controls	There is no equivalent feature
LOC symbol	Domain object record onsite location attribute
PROTECTION symbol	Domain object record protection attribute
ALLOCSIZE symbol	There is no equivalent feature
LBL symbol	There is no equivalent feature
FRESTA symbol	Domain object record transition time attribute. When it has no value, volumes transition to the free state when the scratch date arrives.
TRANS_AGE symbol	Domain object record transition time attribute. When it has a value, volumes transition to the transition state when the scratch date arrives.
ALLOCSCRATCH symbol	Domain object record scratch time attribute
MAXSCRATCH symbol	Domain object record maximum scratch time attribute
TAPEPURGE_WORK symbol	Domain object record mail users attribute
TAPEPURGE_MAIL symbol	Domain object record mail users attribute

Table K–3 Comparison of TAPESTART.COM to MDMS Version 3 Features.

VLT symbol	Domain object record offsite location attribute
Drive Controls ALLDEV symbol SELDEV symbol	No equivalent features
ALLTIM symbol	No equivalent feature
TOPERS symbol	Domain object record OPCOM classes attribute
LOAD symbols QUICKLOAD symbol QUICKLOAD_RETRIES symbol	No equivalent features
UNATTENDED_BACKUPS symbol	No equivalent features

L

Sample Configuration of MDMS

This appendix shows a sample configuration of Media and Device Management System (MDMS) including examples for the steps involved.

L.1 Configuration Order

Configuration - which involves the creation or definition of MDMS objects, should take place in the following order:

1. Location
2. Media type
3. Node
4. Jukebox
5. Drives
6. Pools
7. Volumes

Creating these objects in the above order ensures that the following informational message, does not appear:

```
%MDMS-I-UNDEFINEDREFS, object contains undefined referenced objects
```

This message appears if an attribute of the object is not defined in the database. The object is created even though the attribute is not defined. The sample configuration consists of the following:

- Four nodes

```
SMITH1 - ACCOUN cluster node
SMITH2 - ACCOUN cluster node
SMITH3 - ACCOUN cluster node
JONES - a client node
```

- TL826 Jukebox with robot 1DUA560 and the following six drives:

```
$1$MUA560
$1$MUA561
$1$MUA562
$1$MUA563
$1$MUA564
$1$MUA565
```

The following examples illustrate each step in the order of configuration.

L.1.1 Configuration Step 1 Example - Defining Locations

This example lists the MDMS commands to define an offsite and onsite location for this domain.

```
$ !
$ !   create onsite location
$ !
$ MDMS CREATE LOCATION BLD1_COMPUTER_ROOM -
  /DESCRIPTION="Building 1 Computer Room"
$ MDMS SHOW LOCATION BLD1_COMPUTER_ROOM
    Location: BLD1_COMPUTER_ROOM
 Description: Building 1 Computer Room
      Spaces:
 In Location:
$ !
$ !  create offsite location
$ !
$ MDMS CREATE LOCATION ANDYS_STORAGE -
  /DESCRIPTION="Andy's Offsite Storage, corner of 5th and Main"
$ MDMS SHOW LOCATION ANDYS_STORAGE
    Location: ANDYS_STORAGE
 Description: Andy's Offsite Storage, corner of 5th and Main
      Spaces:
 In Location:
```

L.1.2 Configuration Step 2 Example - Defining Media Type

This example shows the MDMS command to define the media type used in the TL826.

```
 !
$ !  create the media type
$ !
$ MDMS CREATE MEDIA_TYPE TK88K -
  /DESCRIPTION="Media type for volumes in TL826 with TK88 drives" -
  /COMPACTION      ! volumes are written in compaction mode
$ MDMS SHOW MEDIA_TYPE TK88K
  Media type: TK88K
 Description: Media type for volumes in TL826 with TK88 drives
     Density:
  Compaction: YES
    Capacity: 0
      Length: 0
```

L.1.3 Configuration Step 3 Example - Defining Domain Attributes

This example shows the MDMS command to set the domain attributes. The reason this command is not run until after the locations and media type are defined, is because they are default attributes for the domain object. Note that the deallocation state (transition) is taken as the default. All of the rights are taken as default also.

```
$ !
$ !  set up defaults in the domain record
$ !
$ MDMS SET DOMAIN -
  /DESCRIPTION="Smiths Accounting Domain" - ! domain name
  /MEDIA_TYPE=TK88K -                    ! default media type
  /OFFSITE_LOCATION=ANDYS_STORAGE -      ! default offsite location
  /ONSITE_LOCATION=BLD1_COMPUTER_ROOM -  ! default onsite location
  /PROTECTION=(S:RW,O:RW,G:RW,W)         ! default protection for volumes
$ MDMS SHOW DOMAIN/FULL
      Description: Smiths Accounting Domain
```

```
            Mail: SYSTEM
 Offsite Location: ANDYS_STORAGE
  Onsite Location: BLD1_COMPUTER_ROOM
  Def. Media Type: TK88K
 Deallocate State: TRANSITION
      Opcom Class: TAPES
         Priority: 1536
       Request ID: 2576
       Protection: S:RW,O:RW,G:RW,W
   DB Server Node: SPIELN
   DB Server Date: 1-FEB-1999 08:18:20
 Max Scratch Time: NONE
     Scratch Time:  365 00:00:00
  Transition Time:   14 00:00:00
  Network Timeout:    0 00:02:00
        ABS Rights: NO
    SYSPRIV Rights: YES
Application Rights: MDMS_ASSIST
                    MDMS_LOAD_SCRATCH
                    MDMS_ALLOCATE_OWN
                    MDMS_ALLOCATE_POOL
                    MDMS_BIND_OWN
                    MDMS_CANCEL_OWN
                    MDMS_CREATE_POOL
                    MDMS_DEALLOCATE_OWN
                    MDMS_DELETE_POOL
                    MDMS_LOAD_OWN
                    MDMS_MOVE_OWN
                    MDMS_SET_OWN
                    MDMS_SHOW_OWN
                    MDMS_SHOW_POOL
                    MDMS_UNBIND_OWN
                    MDMS_UNLOAD_OWN
   Default Rights:
  Operator Rights: MDMS_ALLOCATE_ALL
                    MDMS_ASSIST
                    MDMS_BIND_ALL
                    MDMS_CANCEL_ALL
                    MDMS_DEALLOCATE_ALL
                    MDMS_INITIALIZE_ALL
                    MDMS_INVENTORY_ALL
                    MDMS_LOAD_ALL
                    MDMS_MOVE_ALL
                    MDMS_SHOW_ALL
                    MDMS_SHOW_RIGHTS
                    MDMS_UNBIND_ALL
                    MDMS_UNLOAD_ALL
                    MDMS_CREATE_POOL
                    MDMS_DELETE_POOL
                    MDMS_SET_OWN
                    MDMS_SET_POOL
      User Rights: MDMS_ASSIST
                    MDMS_ALLOCATE_OWN
                    MDMS_ALLOCATE_POOL
                    MDMS_BIND_OWN
                    MDMS_CANCEL_OWN
                    MDMS_DEALLOCATE_OWN
                    MDMS_LOAD_OWN
                    MDMS_SHOW_OWN
                    MDMS_SHOW_POOL
                    MDMS_UNBIND_OWN
                    MDMS_UNLOAD_OWN
```

L.1.4 Configuration Step 4 Example - Defining MDMS Database Nodes

This example shows the MDMS commands for defining the three MDMS database nodes of the cluster ACCOUN. This cluster is configured to use DECnet-PLUS.

Note that a node is defined using the DECnet node name as the name of the node.

- If the node has DECnet-PLUS installed, the DECnet Fullname attribute must be the DECnet-PLUS full name.
- If the node uses TCP/IP, the TCP/IP attribute should be defined.
- If you use the GUI, you must define the TCP/IP attribute and include TCPIP in the Transports attribute.

```
$ !
$ !   create nodes
$ !   database node
$ MDMS CREATE NODE SMITH1 -            ! DECnet node name
  /DESCRIPTION="ALPHA node on cluster ACCOUN" -
  /DATABASE_SERVER -                   ! this node is a database server
  /DECNET_FULLNAME=SMI:.BLD.SMITH1 -   ! DECnet-Plus name
  /LOCATION=BLD1_COMPUTER_ROOM -
  /TCPIP_FULLNAME=SMITH1.SMI.BLD.COM - ! TCP/IP name
$ MDMS SHOW NODE SMITH1
           Node: SMITH1
    Description: ALPHA node on cluster ACCOUN
DECnet Fullname: SMI:.BLD.SMITH1
TCP/IP Fullname: SMITH1.SMI.BLD.COM:2501-2510
       Disabled: NO
Database Server: YES
       Location: BLD1_COMPUTER_ROOM
  Opcom Classes: TAPES
     Transports: DECNET,TCPIP
$ MDMS CREATE NODE SMITH2  -           ! DECnet node name
  /DESCRIPTION="ALPHA node on cluster ACCOUN" -
  /DATABASE_SERVER -                   ! this node is a database server
  /DECNET_FULLNAME=SMI:.BLD.SMITH2 -   ! DECnet-Plus name
  /LOCATION=BLD1_COMPUTER_ROOM -
  /TCPIP_FULLNAME=SMITH2.SMI.BLD.COM -  ! TCP/IP name
  /TRANSPORT=(DECNET,TCPIP)            ! TCPIP used by JAVA GUI and JONES
$ MDMS SHOW NODE SMITH2
           Node: SMITH2
    Description: ALPHA node on cluster ACCOUN
DECnet Fullname: SMI:.BLD.SMITH2
TCP/IP Fullname: SMITH2.SMI.BLD.COM:2501-2510
       Disabled: NO
Database Server: YES
       Location: BLD1_COMPUTER_ROOM
  Opcom Classes: TAPES
     Transports: DECNET,TCPIP
$ MDMS CREATE NODE SMITH3 -            ! DECnet node name
  /DESCRIPTION="VAX node on cluster ACCOUN" -
  /DATABASE_SERVER -                   ! this node is a database server
  /DECNET_FULLNAME=SMI:.BLD.SMITH3  -   ! DECnet-Plus name
  /LOCATION=BLD1_COMPUTER_ROOM -
  /TCPIP_FULLNAME=CROP.SMI.BLD.COM -    ! TCP/IP name
  /TRANSPORT=(DECNET,TCPIP)            ! TCPIP used by JAVA GUI and JONES
$ MDMS SHOW NODE SMITH3
           Node: SMITH3
    Description: VAX node on cluster ACCOUN
DECnet Fullname: SMI:.BLD.SMITH3
TCP/IP Fullname: CROP.SMI.BLD.COM:2501-2510
       Disabled: NO
Database Server: YES
       Location: BLD1_COMPUTER_ROOM
  Opcom Classes: TAPES
     Transports: DECNET,TCPIP
```

L.1.5 Configuration Step 5 Example - Defining a Client Node

This example shows the MDMS command for creating a client node. TCP/IP is the only transport on this node.

```
$ !
$ !    client node
$ !    only has TCP/IP
$ MDMS CREATE NODE JONES -
  /DESCRIPTION="ALPHA client node, standalone" -
  /NODATABASE_SERVER -                 ! not a database server
  /LOCATION=BLD1_COMPUTER_ROOM -
  /TCPIP_FULLNAME=JONES.SMI.BLD.COM -  ! TCP/IP name
  /TRANSPORT=(TCPIP)                   ! TCPIP is used by JAVA GUI
$ MDMS SHOW NODE JONES
           Node: JONES
    Description: ALPHA client node, standalone
DECnet Fullname:
TCP/IP Fullname: JONES.SMI.BLD.COM:2501-2510
       Disabled: NO
Database Server: NO
       Location: BLD1_COMPUTER_ROOM
  Opcom Classes: TAPES
     Transports: TCPIP
```

L.1.6 Configuration Step 6 Example - Creating a Jukebox

This example shows the MDMS command for creating a jukebox

```
$ !
$ !   create jukebox
$ !
$ MDMS CREATE JUKEBOX TL826_JUKE -
  /DESCRIPTION="TL826 Jukebox in Building 1" -
  /ACCESS=ALL -           ! local + remote for JONES
  /AUTOMATIC_REPLY -      ! MDMS automatically replies to OPCOM requests
  /CONTROL=MRD -          ! controled by MRD robot control
  /NODES=(SMITH1,SMITH2,SMITH3) - ! nodes the can control the robot
  /ROBOT=$1$DUA560 -      ! the robot device
  /SLOT_COUNT=176         ! 176 slots in the library
$ MDMS SHOW JUKEBOX TL826_JUKE
     Jukebox: TL826_JUKE
 Description: TL826 Jukebox in Building 1
       Nodes: SMITH1,SMITH2,SMITH3
      Groups:
    Location: BLD1_COMPUTER_ROOM
    Disabled: NO
      Shared: NO
  Auto Reply: YES
      Access: ALL
       State: AVAILABLE
     Control: MRD
       Robot: $1$DUA560
  Slot Count: 176
       Usage: NOMAGAZINE
```

L.1.7 Configuration Step 7 Example - Defining a Drive

This example shows the MDMS commands for creating the six drives for the jukebox.
This example is a command procedure that uses a counter to create the six drives. In this example it is easy to do this because of the drive name and device name. You may want to have the drive name the same as the device name. For example:

```
$ MDMS CREATE DRIVE $1$MUA560/DEVICE=$1$MUA560
```

This works fine if you do not have two devices in your domain with the same name.

```
$ COUNT = COUNT + 1
$ IF COUNT .LT. 6 THEN GOTO DRIVE_LOOP
$DRIVE_LOOP:
$ MDMS CREATE DRIVE TL826_D1 -
  /DESCRIPTION="Drive 1 in the TL826 JUKEBOX" -
  /ACCESS=ALL -           ! local + remote for JONES
  /AUTOMATIC_REPLY -      ! MDMS automatically replies to OPCOM requests
  /DEVICE=$1$MUA561 - ! physical device
  /DRIVE_NUMBER=1 - ! the drive number according to the robot
  /JUKEBOX=TL826_JUKE -   ! jukebox the drives are in
  /MEDIA_TYPE=TK88K -     ! media type to allocate drive and volume for
  /NODES=(SMITH1,SMITH2,SMITH3)! nodes that have access to drive
$ MDMS SHOW DRIVE TL826_D1
           Drive: TL826_D1
     Description: Drive 1 in the TL826 JUKEBOX
          Device: $1$MUA561
           Nodes: SMITH1,SMITH2,SMITH3
          Groups:
          Volume:
        Disabled: NO
          Shared: NO
       Available: NO
           State: EMPTY
         Stacker: NO
 Automatic Reply: YES
  RW Media Types: TK88K
  RO Media Types:
          Access: ALL
         Jukebox: TL826_JUKE
    Drive Number: 1
       Allocated: NO
             :
             :
             :
$ MDMS CREATE DRIVE TL826_D5 -
  /DESCRIPTION="Drive 5 in the TL826 JUKEBOX" -
  /ACCESS=ALL -           ! local + remote for JONES
  /AUTOMATIC_REPLY -      ! MDMS automatically replies to OPCOM requests
  /DEVICE=$1$MUA565 - ! physical device
  /DRIVE_NUMBER=5 - ! the drive number according to the robot
  /JUKEBOX=TL826_JUKE -   ! jukebox the drives are in
  /MEDIA_TYPE=TK88K -     ! media type to allocate drive and volume for
  /NODES=(SMITH1,SMITH2,SMITH3)! nodes that have access to drive
$ MDMS SHOW DRIVE TL826_D5
           Drive: TL826_D5
     Description: Drive 5 in the TL826 JUKEBOX
          Device: $1$MUA565
           Nodes: SMITH1,SMITH2,SMITH3
          Groups:
          Volume:
        Disabled: NO
          Shared: NO
       Available: NO
           State: EMPTY
         Stacker: NO
 Automatic Reply: YES
  RW Media Types: TK88K
  RO Media Types:
          Access: ALL
         Jukebox: TL826_JUKE
    Drive Number: 5
       Allocated: NO
$ COUNT = COUNT + 1
$ IF COUNT .LT. 6  THEN GOTO DRIVE_LOOP
```

L.1.8 Configuration Step 8 Example - Defining Pools

This example shows the MDMS commands to define two pools: ABS and HSM. The pools need to have the authorized users defined.

```
$ !
$ !   create pools
$ !
$ mdms del pool abs
$ MDMS CREATE POOL ABS -
  /DESCRIPTION="Pool for ABS" -
  /AUTHORIZED=(SMITH1::ABS,SMITH2::ABS,SMITH3::ABS,JONES::ABS)
$ MDMS SHOW POOL ABS
            Pool: ABS
     Description: Pool for ABS
Authorized Users: SMITH1::ABS,SMITH2::ABS,SMITH3::ABS,JONES::ABS
   Default Users:
$ mdms del pool hsm
$ MDMS CREATE POOL HSM -
  /DESCRIPTION="Pool for HSM" -
  /AUTHORIZED=(SMITH1::HSM,SMITH2::HSM,SMITH3::HSM)
$ MDMS SHOW POOL HSM
            Pool: HSM
     Description: Pool for HSM
Authorized Users: SMITH1::HSM,SMITH2::HSM,SMITH3::HSM
   Default Users:
```

L.1.9 Configuration Step 9 Example - Defining Volumes using the /VISION qualifier

This example shows the MDMS commands to define the 176 volumes in the TL826 using the /VISION qualifier. The volumes have the BARCODES on them and have been placed in the jukebox. Notice that the volumes are created in the UNINITIALIZED state. The last command in the example initializes the volumes and changes the state to FREE.

```
$ !
$ !  create volumes
$ !
$ !  create 120 volumeS for ABS
$ !    the media type, offsite location, and onsite location
$ !    values are taken from the DOMAIN object
$ !
$ MDMS CREATE VOLUME -
    /DESCRIPTION="Volumes for ABS" -
    /JUKEBOX=TL826_JUKE -
    /POOL=ABS -
    /SLOTS=(0-119) -
    /VISION
$ MDMS SHOW VOLUME BEB000
      Volume: BEB000
 Description: Volumes for ABS
   Placement: ONSITE BLD1_COMPUTER_ROOM
 Media Types: TK88K                              Username:
        Pool: ABS                              Owner UIC: NONE
 Error Count:          0                         Account:
 Mount Count:          0                        Job Name:
       State: UNINITIALIZED                     Magazine:
 Avail State: UNINITIALIZED                      Jukebox: TL826_JUKE
Previous Vol:                                       Slot: 0
    Next Vol:                                      Drive:
      Format: NONE                           Offsite Loc: ANDYS_STORAGE
  Protection: S:RW,O:RW,G:RW,W              Offsite Date: NONE
    Purchase:  1-FEB-1999 08:19:00             Onsite Loc: BLD1_COMPUTER_ROOM
    Creation:  1-FEB-1999 08:19:00                   Space:
        Init:  1-FEB-1999 08:19:00            Onsite Date: NONE
```

```
  Allocation: NONE                                Brand:
     Scratch: NONE                    Last Cleaned:  1-FEB-1999 08:19:00
Deallocation: NONE                     Times Cleaned: 0
  Trans Time:   14 00:00:00               Rec Length: 0
       Freed: NONE                      Block Factor: 0
 Last Access: NONE
$ !
$ ! create 56 volumes for HSM
$ !
$ MDMS CREATE VOLUME -
    /DESCRIPTION="Volumes for HSM" -
    /JUKEBOX=TL826_JUKE -
    /POOL=HSM -
    /SLOTS=(120-175) -
    /VISION
$ MDMS SHOW VOL BEB120
      Volume: BEB120
 Description: Volumes for HSM
   Placement: ONSITE BLD1_COMPUTER_ROOM
 Media Types: TK88K                         Username:
        Pool: HSM                          Owner UIC: NONE
 Error Count:          0                     Account:
 Mount Count:          0                    Job Name:
       State: UNINITIALIZED                 Magazine:
 Avail State: UNINITIALIZED                  Jukebox: TL826_JUKE
Previous Vol:                                   Slot: 120
    Next Vol:                                  Drive:
      Format: NONE                       Offsite Loc: ANDYS_STORAGE
  Protection: S:RW,O:RW,G:RW,W           Offsite Date: NONE
    Purchase:  1-FEB-1999 08:22:16          Onsite Loc: BLD1_COMPUTER_ROOM
    Creation:  1-FEB-1999 08:22:16               Space:
        Init:  1-FEB-1999 08:22:16         Onsite Date: NONE
  Allocation: NONE                              Brand:
     Scratch: NONE                    Last Cleaned:  1-FEB-1999 08:22:16
Deallocation: NONE                     Times Cleaned: 0
  Trans Time:   14 00:00:00               Rec Length: 0
       Freed: NONE                      Block Factor: 0
 Last Access: NONE
$ !
$ ! initialize all of the volumes
$ !
$ MDMS INITIALIZE VOLUME -
    /JUKEBOX=TL826_JUKE -
    /SLOTS=(0-175)
$ MDMS SHOW VOL BEB000
      Volume: BEB000
 Description: Volumes for ABS
   Placement: ONSITE BLD1_COMPUTER_ROOM
 Media Types: TK88K                         Username:
        Pool: ABS                          Owner UIC: NONE
 Error Count:          0                     Account:
 Mount Count:          0                    Job Name:
       State: FREE                          Magazine:
 Avail State: FREE                           Jukebox: TL826_JUKE
Previous Vol:                                   Slot: 0
    Next Vol:                                  Drive:
      Format: NONE                       Offsite Loc: ANDYS_STORAGE
  Protection: S:RW,O:RW,G:RW,W           Offsite Date: NONE
    Purchase:  1-FEB-1999 08:19:00          Onsite Loc: BLD1_COMPUTER_ROOM
    Creation:  1-FEB-1999 08:19:00               Space:
        Init:  1-FEB-1999 08:19:00         Onsite Date: NONE
  Allocation: NONE                              Brand:
     Scratch: NONE                    Last Cleaned:  1-FEB-1999 08:19:00
Deallocation: NONE                     Times Cleaned: 0
  Trans Time:   14 00:00:00               Rec Length: 0
       Freed: NONE                      Block Factor: 0
 Last Access: NONE
```

M

Converting SLS/MDMS V2.X to MDMS V3

M.1 Operational Differences Between SLS/MDMS V2 & MDMS V3

This appendix discusses the main operational differences in the new version of MDMS from previous versions. In some cases, there are conceptual differences in approach, while others are more changes of the 'nuts and bolts' kind. This appendix is designed to acquaint you with the changes, including why some of them were made, in order to make the upgrade as smooth as possible. It will also enable you to use the new features to optimize your configuration and usage of the products.

M.1.1 Architecture

The media manager used for previous versions of ABS and HSM was embedded within the SLS product. The MDMS portion of SLS was implemented in the same requester (SLS$TAPMGRRQ), database (SLS$TAPMGRDB) and OPCOM (SLS$OPCOM) processes used for SLS.

The STORAGE DCL interface contained both SLS and MDMS commands, as did the forms interface and the configuration file TAPESTART.COM. All media management status and error messages used the SLS prefix. All in all, it was quite difficult to determine where MDMS left off and SLS began. In addition, SLS contained many restrictions in its design that inhibited optimal use of ABS and HSM in a modern environment.

Compaq reviewed the SLS/MDMS design and the many requests for enhancements and decided to completely redesign the media manager for ABS and HSM. The result is MDMS V3 (V3.0 and later), which is included as the preferred media manager for both ABS and HSM V3.0 and later. The main functional differences between MDMS V3 and previous versions include:

- An object oriented design that begins at the user interfaces and is propagated throughout the product. You will become familiar with the ten classes of objects and use a consistent interface to manipulate them. A multi-threaded design that allows any number of concurrent operations throughout the MDMS domain.
- Complete separation from SLS. MDMS now has its own distinct user interfaces and error messages. Its two fully functional interfaces (DCL and GUI) can be used interchangeably at your preference. It is no longer necessary to switch interfaces to perform certain functions. The GUI is usable on OpenVMS and Windows-based PCs.
- A simplified design that utilizes only one server process on a node. This process performs all MDMS operations on a node. Support of modern network protocols including TCP/IP and DECnet-Plus with fullname support.
- New features that allow lights-out operations and enhance ease of use.
- A non-device-specific approach to jukebox handling that should allow support of new devices without code modifications. Flexible logging and auditing capabilities that allow you to see what MDMS is working on and has completed.

- While MDMS V3 has been completely re-engineered, a great effort was made to ensure compatibility and upgradability with the previous version.
- Important attributes and functions that you may be using are retained, albeit in a slightly different form.

The following sections will guide you through the changes one by one.

M.1.2 MDMS Interfaces

The previous SLS/MDMS contained several "interfaces" that you used to configure and run operations. These were:

- The file TAPESTART.COM - used for configuration of drives, jukeboxes, media types and other related parameters. Changes to the configuration required SLS/MDMS to be restarted.
- DCL STORAGE commands - used for day-to-day operations and manipulation of volumes and magazines
- A forms interface - used for more complex operations and certain operations not supported by DCL.
- Utilities like SLS$VOLUME to repair the database after an error

While these interfaces together provided a fully functional product, their inconsistent syntax and coverage made them hard to use.

With MDMS V3, a radical new approach was taken. Two interfaces were chosen for implementation, each of which is fully functional:

A modern DCL interface –this interface was designed with a consistent syntax which is easier to remember. It is also functionally complete so that all MDMS operations can be initiated without manipulating files or forms. This interface can be used by batch jobs and command procedures, as well as by users.

A modern GUI interface –based on Java technology, is provided for those users who prefer graphical interfaces. Like the DCL interface, it is functionally complete and all operations can be initiated from it (with necessary exceptions).

In addition, it contains a number of wizards that can be used to guide you through complex operations such as configuration and volume rotation. The GUI is usable on both OpenVMS Alpha (V7.1-2 and later) systems and Windows-based PC systems.

Note

The GUI requires TCP/IP to be running on the OpenVMS MDMS server node and the node on which the GUI is running.

There are also a limited number of logical names used for tailoring the functionality of the product and initial startup (when the database is not available).The forms interface, TAPESTART and the utilities have been eliminated. When you install MDMS V3 you will be asked about converting TAPESTART and the old databases to the new format. This is discussed in the Appendix of the Guide to Operations.

Both the DCL and GUI take a forgiving approach to creating, modifying and deleting objects, in that they allow you to perform the operation even if it creates an inconsistency in the database, as follows:

- You can create or modify objects by referencing objects that have not yet been defined. This allows you to enter commands "out-of-order". A warning message is displayed if an object contains undefined references to other objects.

- You can delete objects that have references to other objects. The GUI delete wizard will help go through procedures to clean up references in order.
- One other global feature has been added to MDMS V3 when creating objects. This is the INHERIT option that allows you to create an object using most of the attributes of an existing object. All fields except the object name and protected fields may be inherited. The Command Reference Guide lists fields that cannot be inherited for any particular object.

M.1.3 Rights and Privileges

Both the DCL interface and the GUI require privileges to execute commands. These privileges apply to all commands, including defining objects and attributes that used to reside in TAPESTART.

With MDMS V3, privileges are obtained by defining MDMS rights in users' UAF definitions. There are three high-level rights, one each for an MDMS user, application and operator. There are also a large set of low-level rights, several for each command, that relate to high level rights by a mapping defined in the domain object.

In addition, a guru right is enabled which allows any command, and the OpenVMS privilege SYSPRV can optionally be used instead of the guru right. This mechanism replaces the six SLS/MDMS V2 rights defined in TAPESTART and the OPER privilege.

A full description of rights can be found in the Appendix of the ABS/HSM Command Reference Guide.

M.1.4 The MDMS Domain

There was no real concept of a domain with SLS/MDMS V2. The scope of operations within SLS varied according to what was being considered.

For example, attributes defined in TAPESTART were applicable to all nodes using that version of the file - normally from one node to a cluster. By contrast, volumes, magazines and pools had scope across clusters and were administered by a single database process running somewhere in the environment.

MDMS V3 formally defines a domain object, which contains default attribute values that can be applied to any object which does not have them specifically defined. MDMS formally supports a single domain, which supports a single database. All objects (jukeboxes, drives, volumes, nodes, magazines etc.) are defined within the domain.

This introduces some level of incompatibility with the previous version, especially regarding parameters stored in TAPESTART. Since TAPESTART could potentially be different on every node, default parameters like MAXSCRATCH could potentially have different values on each node (although there seemed to be no particularly good reason for this). MDMS V3 has taken the approach of defining default attribute values at the domain level, but also allowing you to override some of these at the a specific object level (for example, OPCOM classes for nodes). In other cases, values such at LOC and VAULT defined in TAPESTART are now separate objects in their own right.

After installing MDMS V3, you will need to perform conversions on each TAPESTART that you have in your domain. If your TAPESTART files on every node were compatible (not necessarily identical, but not conflicting) this conversion will be automatic. However, if there were conflicts, these are flagged in a separate conversion log file, and need to be manually resolved. For example, if there are two drives called 1MUA500 on different nodes, then one or both need to be renamed for use in the new MDMS.

It is possible to support multiple domains with MDMS V3, but when you do this you need to ensure that no objects span more than one domain.

Each domain contains its own database, which has no relationship to any database in another domain.

For example, your company may have two autonomous groups which have their own computer resources, labs and personnel. It is reasonable for each group to operate within their own domain, but realize that nodes, jukeboxes and volumes cannot be shared among the two groups. If there is a need to share certain resources (e.g. jukeboxes) it is also possible to utilize a single domain, and separate certain resources in other ways.

M.1.5 Drives

The drive object in MDMS is similar in concept to a drive in SLS/MDMS V2. However, the naming convention for drives in MDMS V3 is different.

In V2, drives were named after the OpenVMS device name, optionally qualified by a node.

In MDMS V3, drives are named like most other objects - they may be any name up to 31 characters in length, but they must be unique within the domain. This allows you to give drives names like DRIVE_1 rather than 1MUA510 if you wish, and specify the OpenVMS device name with the DEVICE_NAME attribute. It is also equally valid to name the drive after the OpenVMS device name as long as it is unique within the domain.

Nodes for drives are specified by the NODES or GROUPS attributes. You should specify all nodes or groups that have direct access to the drive.

Do not specify a node or group name in the drive name or OpenVMS device name.

Consider two drives named 1MUA500, one on cluster BOSTON, the other on cluster HUSTON, and you wish to use a single MDMS domain.

Here's how you might set up the drives

```
$ MDMS CREATE DRIVE BOS_MUA500/DEVICE=$1$MUA500/GROUP=BOSTON
$ MDMS CREATE DRIVE HUS_MUA500/DEVICE=$1$MUA500/GROUP=HUSTON
```

The new ACCESS attribute can limit use of the drive to local or remote access. Local access is defined as access by any of the nodes in the NODES attribute, or any of the nodes defined in the group object defined in the GROUP attributes. Remote access is any other node. By default, both local and remote access are allowed.

With MDMS V3, drives may be defined as being as jukebox controlled, stacker controlled or stand-alone as follows:

A drive is jukebox controlled when it resides in a jukebox, and you wish random-access loads/unloads of any volume in the jukebox. Define a jukebox name, a control mechanism (MRD or DCSC), and a drive number for an MRD jukebox. The drive number is the number MRD uses to refer to the drive, and starts from zero.

A drive may be defined as a stacker when it resides in a jukebox and you wish sequential loading of volumes, or if the drive supports a stacker loading system. In this case, do not define a jukebox name, but set the STACKER attribute.

If the drive is stand-alone (loadable only by an operator), do not define a jukebox and clear the STACKER attribute.

Set the AUTOMATIC_REPLY attribute if you wish OPCOM requests on the drive to be completed without an operator reply. This enables a polling scheme which will automatically cancel the request when the requested condition has been satisfied.

M.1.6 Jukeboxes

In previous SLS/MDMS versions, jukeboxes were differentiated as libraries, loaders and ACS devices, each with their own commands and functions. With MDMS V3, all automatic loading devices are brought together under the concept of a jukebox object.

Jukeboxes are named like other objects as a unique name up to 31 characters. Each jukebox may be controlled by one of two subsystems:

- MRD - for most SCSI jukeboxes, including some StorageTek silos
- DCSC - for most existing and older StorageTek silos

The new ACCESS attribute can limit use of the jukebox to local or remote access. Local access is defined as access by any of the nodes in the NODES attribute, or any of the nodes defined in the group object defined in the GROUP attributes. Remote access is any other node. By default, both local and remote access is allowed.

For MRD jukeboxes, the robot name is the name of the device that MRD accesses for jukebox control, and is equivalent to the device name listed first in the old TAPE_JUKEBOXES definition in TAPESTART, but without the node name. As with drives, nodes for the jukebox must be specified using the NODES or GROUPS attributes.

Jukeboxes now have a LOCATION attribute, which is used in OPCOM messages related to moving volumes into and out of the jukebox. When moving volumes into a jukebox, you may first be prompted to move them to the jukebox location (if they are not already in that location). Likewise, when moving volumes out of the jukebox they will first be moved to the jukebox location. The reason for this is practical; it is more efficient to move all the volumes from wherever they were to the jukebox location, then move all the volumes to the final destination.

One of the more important aspects of jukeboxes is whether you will be using the jukebox with magazines. As described in the magazine section below, MDMS V3 treats magazines as a set of volumes within a physical magazine that share a common placement and move schedule. Unlike SMS/MDMS V2, it is not necessary to relate volumes to magazines just because they reside in a physical magazine, although you can. It is equally valid for volumes to be moved directly and individually in and out of jukeboxes regardless of whether they reside in a magazine within the jukebox.

This is the preferred method when it is expected that the volumes will be moved independently in and out of the jukebox.

If you decide to formally use magazines, you should set the jukebox usage to magazine. In addition, if the jukebox can potentially hold multiple magazines at once (for example, a TL820style jukebox), you can optionally define a topology field that represents the physical topology of the jukebox (i.e. towers, faces, levels and slots). If you define a topology field, OPCOM messages relating to moving magazines in and out of the jukebox will contain a magazine position in the jukebox, rather than a start slot for the magazine. Use of topology and position is optional, but makes it easier for operators to identify the appropriate magazine to move.

Importing and exporting volumes (or magazines) into and out of a jukebox has been replaced by a common MOVE command, that specifies a destination parameter. Depending on whether the destination is a jukebox, a location or a magazine, the direction of movement is determined. Unlike previous versions, you can move multiple volumes in a single command, and the OPCOM messages contain all the volumes to move that have a common source and destination location. If the jukebox supports ports or caps, all available ports and caps will be used. The move is flexible in that you can stuff volumes into the ports/caps in any order when importing, and all ports will be used on export. All port/cap oriented jukeboxes support automatic reply on

OPCOM messages meaning that the messages do not have to be acknowledged for the move to complete.

M.1.7 Locations

The concept of locations has been greatly expanded from SLS/MDMS V2, where a copy of TAPESTART had a single "onsite" location defined in the LOC symbol, and a single "offsite" location defined in the "VAULT" symbol.

With MDMS V3, locations are now separate objects with the usual object name of up to 31 characters. Locations can be arranged in a hierarchy, allowing locations to be within other locations. For example, you can define BOSTON_CAMPUS as a location, with BUILDING_1, BUILDING_2 located in BOSTON_CAMPUS, and ROOM_100, ROOM_200 located within BUILDING_1. Locations that have common roots are regarded as compatible locations, which are used for allocating drives and volumes. For example, when allocating a volume currently located in ROOM_200 but specifying a location of BUILDING_1, these two locations are considered compatible. However, if BUILDING_2 was specified, they are not considered compatible since ROOM_200 is in BUILDING_1.

Locations are not officially designated as ONSITE or OFFSITE, as they could be both in some circumstances. However, each volume and magazine have offsite and onsite location attributes that should be set to valid location objects. This allows for any number of onsite or offsite locations to be defined across the domain.

You can optionally associate "spaces" with locations: spaces are subdivisions within a location in which volumes or magazines can be stored. The term "space" replaces the term "slot" in SLS/MDMS V2 as that term was overloaded. In MDMS V3, "slot" is reserved for a numeric slot number in a jukebox or magazine, whereas a space can consist of up to 8 alphanumeric characters.

M.1.8 Media Types

In SLS/MDMS V2, media type, density, length and capacity were attributes of drives and volumes, defined both in TAPESTART and in volume records. With MDMS V3, media types are objects that contain the attributes of density, compaction, length, and capacity; drives and volumes reference media types only; the other attributes are defined within the media type object.

If you formerly had media types defined in TAPESTART with different attributes, you need to define multiple media types with MDMS V3. For example, consider the following TAPESTART definitions:

```
MTYPE_1  := TK85K
DENS_1   :=
DRIVES_1 := $1$MUA510:, $1$MUA520:
MTYPE_2  := TK85K
DENS_2   := COMP
DRIVES_2 := $1$MUA510:, $1$MUA520:
```

This definition contains two media type definitions, but with the same name. In MDMS V3, you need to define two distinct media types and allow both drives to support both media types. The equivalent commands in MDMS V3 would be:

```
$ MDMS CREATE MEDIA_TYPE TK85K_N /NOCOMPACTION
$ MDMS CREATE MEDIA_TYPE TK85K_C /COMPACTION
$ MDMS CREATE DRIVE $1$MUA510:/MEDIA_TYPES=(TK85K_N,TK85K_C)
$ MDMS CREATE DRIVE $1$MUA520:/MEDIA_TYPES=(TK85K_N,TK85K_C)
```

M.1.9 Magazines

As discussed in the jukebox section, the concept of magazine is defined as set of volumes sharing common placement and move schedules, rather than simply being volumes loaded in a physical magazine. With the previous SLS/MDMS V2, all volumes physically located in magazines had to be bound to slots in the magazine for both DLT-loader jukeboxes, and TL820 style bin-packs (if moved as a whole).

When converting from SLS/MDMS V2 to MDMS V3, the automatic conversion utility will take existing magazine definitions and create magazines for MDMS V3. It is recommended that you continue to use magazines in this manner until you feel comfortable eliminating them. If you do eliminate them, you remove the dependency of moving all volumes in the magazine as a whole. For TL820 style jukeboxes, volumes will move via the ports.

For DLT-loader style jukeboxes, OPCOM requests will refer to individual volumes for movement. In this case, the operator should remove the magazine from the jukebox, remove or insert volumes into it and reload the magazine into the jukebox.

If you utilize magazines with TL820-style jukeboxes, movement of magazines into the jukebox can optionally be performed using jukebox positions (i.e. the magazine should be placed in tower n, face n, level n) instead of a start slot. For this to be supported, the jukebox should be specified with a topology as explained in the jukebox section. For single-magazine jukeboxes like the TZ887, the magazine can only be placed in one position (i.e. start slot 0).

Like individual volumes, magazines can be set up for automatic movement to/from an offsite location by specifying an offsite/onsite location and date for the magazine. All volumes in the magazine will be moved. An automatic procedure is executed daily at a time specified by logical name

MDMS$SCHEDULED_ACTIVITIES_START_HOUR, or at 01:00 by default. However, MDMS V3 also allows these movements to be initiated manually using a /SCHEDULE qualifier as follows:

```
$ MDMS MOVE MAGAZINE */SCHEDULE=OFFSITE ! Scheduled moves to offsite
$ MDMS MOVE MAGAZINE */SCHEDULE=ONSITE  ! Scheduled moves to onsite
$ MDMS MOVE MAGAZINE */SCHEDULE         ! All scheduled moves
```

M.1.10Nodes

A node is an OpenVMS computer system capable of running MDMS V3, and a node object must be created for each node running ABS or HSM in the domain. Each node object has a node name, which must be the same as the DECnet Phase IV name of the system (i.e. SYS$NODE) if the node runs DECnet, otherwise it can be any unique name up to 31 characters in length.

If you wish the node to support either or both DECnet-Plus (Phase V) or TCP/IP, then you need to define the appropriate fullnames for the node as attributes of the node. Do not specify the fullnames as the node name. For example, the following command specifies a node capable of supporting all three network protocols:

```
$ MDMS CREATE NODE BOSTON -
$_ /DECNET_FULLNAME=CAP:BOSTON.AYO.CAP.COM -
$_ /TCPIP_FULLNAME=BOSTON.AYO.CAP.COM
```

A node can be designated as supporting a database server or not. A node supporting a database server must have direct access to the database files in the domain (DFS/NFS access is not recommended). The first node you install MDMS V3 on should be designated as a database server.

Subsequent nodes may or may not be designated as database servers. Only one node at a time actually performs as the database server, but if that node fails or is shut down, another designated database server node will take over.

M.1.11Groups

MDMS V3 introduces the group object as a convenient mechanism for describing a group of nodes that have something in common. In a typical environment, you may wish to designate a cluster alias as a group, with the constituent nodes defined as attributes. However, the group concept may be applied to other groups of nodes rather than just those in a cluster. You may define as many groups as you wish, and individual nodes may be defined in any number of groups. However, you may not specify groups within groups, only nodes.

You would typically define groups as a set of nodes that have direct access to drives and jukeboxes, then simply relate the group to the drive or jukebox using the GROUPS attribute. Other uses for groups may be for the definition of users. For example, user SMITH may be the same person for both the BOSTON and HUSTON clusters, so you might define a group containing constituent nodes from the BOSTON and HUSTON clusters. You might then utilize this group as part of an authorized user for a volume pool.

M.1.12 Pools

Pools retain the same purpose for MDMS V3 as for SLS/MDMS V2. They are used to validate users for allocating free volumes. Pool authorization used to be defined through the old forms interface. With MDMS V3, pool authorization is through the pool object. A pool object needs to be created for each pool in the domain.

Pool objects have two main attributes: authorized users and default users. Both sets of users must be in the form NODE::USERNAME or GROUP::USERNAME, and a pool can support up to 1024 characters of authorized and default users. An authorized user is simply a user that is allowed to allocate free volumes from the pool. A default user also allows that user to allocate free volumes from the pool, but in addition it specifies that the pool is to be used when a user does not specify a pool on allocation. As such, each default user should be specified in only one pool, whereas users can be authorized for any number of pools.

M.1.13 Volumes

The volume object is the most critical object for both MDMS V3 and SLS/MDMS V2. Nearly all of the attributes from V2 have been retained, although a few have been renamed. When converting from SLS/MDMS V2.X to MDMS V3, all volumes in the old volume database are created in the new MDMS V3 database. Support for the following attributes has been changed or is unsupported:

Table M–1 Volume Attributes

Old Name	New Name/Support
Density	Unsupported - included in media type object
Flag	State
Length	Unsupported - included in media type object
Location	Onsite Location
Notes	Description
Offsite	Offsite Date

Table M–1 Volume Attributes

Onsite	Onsite Date
Other Side	Unsupported - obsolete feature with RV64 only
Side	Unsupported - obsolete feature with RV64 only
Slot	Space
Zero	Unsupported - can set counters individually

You can create volumes in the MDMS V3 database in one of three ways:

- • By using the CREATE VOLUME command (or GUI equivalent) - this explicitly creates volumes in the database, and gives you the most flexibility in specifying volume attributes.
- By physically inserting volumes into a jukebox, then performing an INVENTORY JUKEBOX/CREATE command referencing a jukebox/slot range (MRD only), or a volume range (DCSC only). Volume attributes can be set from an inherited volume, or media type can be specified. You can later use SET VOLUME to customize other attributes.
- By performing scratch loads in non-jukebox drives with the LOAD DRIVE/CREATE command. Volume attributes can be set from an inherited volume, or media type can be specified. You can later use SET VOLUME to customize other attributes.

Once a volume is created and initial attributes are set, it is not normally necessary to use the SET VOLUME commands to change attributes. Rather, the attributes are automatically modified as a result of some action on the volume, such as ALLOCATE or LOAD. However, in some cases, the volume database and physical reality may get out of synchronization and in these cases you can use SET VOLUME to correct the database.

Note that several fields in the volume object are designated as "protected". These fields are used by MDMS to control the volume's operations within MDMS. You need a special privilege to change protected fields, and in the GUI you need to "Enable Protected" to make these fields writable. When changing a protected field you should ensure that its new value is consistent with other attributes. For example, if manually setting the volume's placement to jukebox, you should ensure that a jukebox name is defined.

Two key attributes in the volume object are "state" and "placement". The volumes states are:

- Uninitialized - This is a new state which is the default when a volume is created. A volume cannot be allocated in this state, and you should either initialize the volume using the MDMS INITIALIZE command, or set the volume to /PREINITIALIZED, which puts it in the Free state.
- Free - This is equivalent to the SLS/MDMS V2 Free state, from which volumes can be initialized.
- Allocated - This is equivalent to the SLS/MDMS V2 Allocated state. Allocated volumes cannot be deleted or otherwise re-used until they are deallocated.
- Transition - This is equivalent to the SLS/MDMS V2 Transition state, that disallows reallocation for a period of time called the Transition Time. Deallocating volumes will either place them in the Transition state or the Free state, depending on the Transition time.
- Unavailable - This is equivalent to the SLS/MDMS V2 Down state, which removes a volume from use.

The placement attribute is a new attribute for MDMS V3, and describes a volume's current placement: in a drive, jukebox, magazine or onsite or offsite location. The placement may also

be "moving", meaning that it is changing placements but the change has not completed. No load, unload or move commands may be issued to a volume that is moving. While a volume is moving, it is sometimes necessary for an operator to determine to where it is moving: for example, moving from a jukebox to a onsite location and space. The operator can issue a SHOW VOL-UME command for moving volumes that shows exactly to where the volume is supposed to be moved.

The new MDMS V3 CREATE VOLUME command replaces the old "Add Volume" storage command. Note that most attributes are supported for both the create volume and set volume commands for consistency purposes.

Volumes can be set up for automatic movement to/from an offsite location by specifying an offsite/onsite location and date, similar to MDMS/SLS V2. Similarly, volumes can be set up for automatic recycling using the scratch date (to go from the allocated to transition states) and free dates (to go from the transition to free states). An automatic procedure is executed daily at a time specified by logical name MDMS$SCHEDULED_ACTIVITIES_START_HOUR, or at 01:00 by default. However, MDMS V3 also allows these movements/state changes to be initiated manually using a /SCHEDULE qualifier as follows:

```
$ MDMS MOVE VOLUME */SCHEDULE=OFFSITE   ! Scheduled moves to offsite
$ MDMS MOVE VOLUME */SCHEDULE=ONSITE    ! Scheduled moves to onsite
$ MDMS MOVE VOLUME */SCHEDULE           ! All scheduled moves
$ MDMS DEALLOCATE VOLUME /SCHEDULE      ! All scheduled deallocations
```

MDMS V3 continues to support the ABS volume set objects (those objects whose volume IDs begins with "&+"). These are normally hidden objects, but they may be displayed in SHOW VOLUME and REPORT VOLUME commands with the ABS_VOLSET option.

In all other respects, the MDMS V3 volume object is equivalent to the SLS/MDMS V2 volume object.

M.1.14Remote Devices

In MDMS V3, support for remote devices is handled through the Remote Device Facility (RDF) in the same manner that was supported for SLS/MDMS V2. DECnet support on both the client and target nodes is required when using RDF.

M.2 Converting SLS/MDMS V2.X Symbols and Database

This section describes how to convert the SLS/MDMS V2.X symbols and database to Media and Device Management Services Version 3 (MDMS). The conversion is automated as much as possible, however, you will need to make some corrections or add attributes to the objects that were not present in SLS/MDMS V2.X.

Before doing the conversion, you should read Chapter 16 - MDMS Configuration in this Guide to Operations to become familiar with configuration requirements.

All phases of the conversion process should be done on the first database node on which you installed MDMS V3. During this process you will go through all phases of the conversion:

1. Convert the symbols in SYS$STARTUP:TAPESTART.COM into the following objects:

- Locations
- Domain
- Nodes
- Media types
- Jukeboxes

- Drives

2. Convert the database authorization file, VALIDATE.DAT, into node objects.

- Convert the rest of the database files:
- Pool Authorization file (POOLAUTH.DAT)
- Slot Definition file (SLOTMAST.DAT)
- Volume Database file (TAPEMAST.DAT)
- Magazine Database file (SLS$MAGAZINE_MASTER_FILE.DAT)

When you install on any other node that does not use the same TAPESTART.COM as the database node, you only do the conversion of TAPESTART.COM

.

M.2.1 Executing the Conversion Command Procedure

To execute the conversion command procedure, type in the following command:

```
$ @MDMS$SYSTEM:MDMS$CONVERT_V2_TO_V3
```

The command procedure will introduce itself and then ask what parts of the SLS/MDMS V2.x you would like to convert.

During the conversion, the conversion program will allow you to start and stop the MDMS server. The MDMS server needs to be running when converting TAPESTART.COM and the database authorization file. The MDMS should not be running during the conversion of the other database files.

During the conversion of TAPESTART.COM the conversion program generates the following file:

```
$ MDMS$SYSTEM:MDMS$LOAD_DB_nodename.COM
```

This file contains the MDMS commands to create the objects in the database. You have the choice to execute this command procedure or not during the conversion.

The conversion of the database files are done by reading the SLS/MDMS V2.x database file and creating objects in the MDMS V3 database files.

You must have the SLS/MDMS V2.x DB server shutdown during the conversion process. Use the following command to shut down the SLS/MDMS V2.x DB server:

```
$ @SLS$SYSTEM:SLS$SHUTDOWN
```

M.2.2 Resolving Conflicts During the Conversion

Because of the difference between SLS/MDMS V2.x and MDMS V3 there will be conflicts during the conversion. Instead of stopping the conversion program and asking you about each conflict, the conversion program generates the following file during each conversion:

$ MDMS$MDMS$LOAD_DB_CONFLICTS_nodename.COM

Where nodename is the name of the node you ran the conversion on. This file is not meant to be executed, it is there for you to look at and see what commands executed and caused a change in the database. This change is flagged because there was already an object in the database or this command changed an attribute of the object.

An example could be that you had two media types of the same name but one specified compressed and one other specified non compressed. This would cause a conflict. MDMS V3

does not allow two media types with the same name but different attributes. What you see in the conflict file would be the command that tried to create the same media type. You will have to create a new media type.

Table M–2 shows the symbols in TAPESTART.COM file and what conflicts they may cause.

At the completion of the conversion of the database files, you will see a message that notes the objects that where in an object but not defined in the database. For example: the conversion program found a pool in a volume record that was not a pool object.

Table M–2 Symbols in TAPESTART.COM

TAPESTART.COM Symbol	MDMS V3 Attribute or Object	Possible conflict
ALLOCSCRATCH	If defined, adds the SCRATCH_TIME attribute to the domain object.	If the ALLOCSCRATCH symbol is different in different TAPESTART.COM files a line is added to the conflict file.
DB_NODES	If defined, creates a node object for the nodes in the DB_NODES list.	A conflict can be generated if the node exists and an attribute changed with a different TAPESTART.COM file. Every drive and jukebox definition in the TAPESTART.COM can cause a node to be created with a /NODATABASE_SERVER qualifier. A DB node will change the attribute to database server, this can cause a line to be added to the conflict file.
DCSC_n_NODES	If defined, creates a node object and adds the node attribute to the DCSC jukebox.	All adds of nodes to jukeboxes cause a line to be added to the conflict file.
DCSC_DRIVES	If defined, creates a drive object for DCSC.	If an attribute is different when adding attributes a line is added to the conflict file.
DENS_x	If defined, adds the density or compaction attribute to a media type. If the value is COMP or NOCOMP then the compaction attribute is define: YES or NO. If the density is anything other than COMP or NOCOMP then the value is placed in the density attribute.	A line is added to the conflict file if the DENS_x is different.
FRESTA	If defined, adds the deallocate state attribute to the domain object.	If the FRESTA symbol is different in different TAPESTART.COM files a line is added to the conflict file.
LOC	Creates a location object and also sets the ONSITE_LOCATION attribute in domain object.	If the object exists or is different than the onsite location attribute in the domain object a line to be added to the conflict file. This can happen when you have different LOC symbol in two TAPESTART.COM files.

Table M–2 Symbols in TAPESTART.COM

MAXSCRATCH	If defined, adds the maximum scratch time attribute to the domain object.	If the MAXSCRATCH symbol is different in different TAPESTART.COM files a line is added to the conflict file.
MTYPE_x	Creates a media type object for each MTYPE_x.	A line is added to the conflict file if a media type is already in the database and another one has the same name. In SLS/MDMS V2.x you could have the same media type name with compaction and nocompaction. In MDMS you cannot have two media types with the same name. You need to change the name of one of the media type and enter it into the database. You will also have to change ABS or HSM to reflect this. Also, you may have to change volume and drive objects.
NET_REQUEST_TIMEOUT	If defined, adds the NETWORK_TIMEOUT attribute to the domain object.	If the NET_REQUEST_TIMEOUT is different in different TAPESTART.COM files a line is added to the conflict file.
PROTECTION	Adds the default protection to the domain object.	A line is added to the conflict file if the protection is changed.
QUICKLOAD	When drives are created, this attribute is added as automatic reply.	A line is added to the conflict file if a drive's automatic reply is changed.
TAPE_JUKEBOXES	Creates a jukebox object for each jukebox in the list.	A line is added to the conflict file if a jukebox is already defined and any of the attributes change.
TAPEPURGE_MAIL	If defined, adds the mail attribute to the domain object.	If the TAPEPURGE_MAIL is different in different TAPESTART.COM files a line is added to the conflict file.
TOPERS	If defined, adds the Opcom class attribute to the domain object.	If the TOPERS symbol is different in different TAPESTART.COM files a line is added to the conflict file.
TRANS_AGE	If defined, adds the transition time attribute to the domain object.	If the TRANS_AGE symbol is different in different TAPESTART.COM files a line is added to the conflict file.
VLT	Creates a location object and also sets the OFFSITE_LOCATION attribute in domain object.	If the object exists or is different than the offsite location attribute in the domain object, a line is added to the conflict file. This can happen when you have different VLT symbol in two TAPESTART.COM files.

M.3 Things to Look for After the Conversion

Because of the differences between SLS/MDMS V2.x and MDMS V3 you should go through the objects and check the attributes and make sure that the objects have the attributes that you want. Table M–3 shows the attributes of objects that you may want to check after the conversion.

Table M–3 Things to Look for After the Conversion

Object	Attribute	Description
Drive	Drive	Make sure you have all of the drives defined. In the MDMS V3 domain, you can only have one drive with a given name. In SLS/MDMS V2.x you could have two drives with the same name if they were in different TAPESTART.COM files. You should make sure that all drives in your domain are in the database. You may have to create one drive with a name of say, DRIVE1 with a device name of 1MUA520 and a node of NODE1. Then create another drive, DRIVE2, with a device name of 1MUA520 and a node of NODE2. A line is added to the conflict file every time a node is added to a drive. This flags you to check that the node really belongs to this drive of if you need to create another drive.
	Description	Make sure this is the description you want for this drive. This attribute is not filled in during the conversion program.
	Device	Make sure this device name does not have a node name as part of it.
	Nodes	Make sure this list of nodes contains the nodes that can reach this drive.
	Disabled	The conversion program enables all drives. If you want this drive disabled, then set this attribute to YES.
	Shared	The conversion program sets this attribute to NO. NO means that MDMS does not have to compete with other applications for this device. If MDMS is supposed to share this device with other applications set this attribute to YES.
	State	Make sure this drive is in the right state. If the drive is not in the right state, you can set this attribute to the right state or issue the following command: $ MDMS SET DRIVE drive/CHECK
	Automatic reply	The conversion program sets this attribute from the QUICK-LOAD symbol. Make sure this is the way you want the drive to react.
	RW media types	The conversion program added media types to this drive as it found them. Make sure these are the correct read-write media types for this drive.
	RO Media Types	There are no read-only media types in SLS/MDMS V2.x so none is added to the drives during conversion. You may want to add some read-only media types to the drive object.
	Access	The conversion program has no way of knowing what the access should be, therefore, it sets the access attribute to ALL. Make sure this is the access you want for this drive.
	Jukebox	Make sure this is the jukebox that this drive is in.

Table M–3 Things to Look for After the Conversion

	Drive Number	Make sure this is the drive number for robot commands.
Domain	Description	Make sure this is the description you want for your domain. The default is: Default MDMS Domain.
	Mail	Make sure this is where you want mail sent when a volume reaches its scratch data and MDMS dellocates it. If you do not want mail sent, make the value blank. The default is: SYSTEM.
	Offsite location	Make sure this is the offsite location that you want for the default when you create objects. This was set to the value of VLT from TAPESTART.COM file. This could be different in each TAPESTART.COM file.
	Onsite location	Make sure this is the onsite location that you want for the default when you create objects.
	Default media type	Make sure this is the media type you want assigned to volumes that you do not specify a media type for, while creating.
	Deallocate state	Make sure this is the default state you want volumes to go to after they have reached their scratch date. This could be changed each time that you convert the TAPESTART.COM file on a new node.
	Opcom classes	Make sure these are the Opcom classes where you want MDMS OPCOM messages directed. This could be changed each time you convert the TAPESTART.COM file on a new node.
	Protection	Make sure this is the default protection that you want assigned to volumes that you do not specify a protection for.
	Maximum scratch time	Make sure this is the default maximum scratch time you want for volumes in your domain. This could be changed each time that you convert the TAPESTART.COM file on a new node.
	Scratch time	Make sure this is the default scratch time you want for volumes in your domain. This could be changed each time that you convert the TAPESTART.COM file on a new node.
	Transition time	Make sure this is the default transition time you want for volumes in your domain. This could be changed each time that you convert the TAPESTART.COM file on a new node.
	Network time-out	Make sure this is the network timeout you want. This could be changed each time that you convert the TAPESTART.COM file on a new node.
Location	Description	Make sure this is the description you want for this location. This attribute is not filled in during the conversion program.
	Spaces	The conversion program cannot fill in spaces so make sure you set the spaces attribute.
	In location	The conversion program cannot fill in this attribute so make sure if this location is in a higher level location you set this attribute.

Table M–3 Things to Look for After the Conversion

Media type	Media type	Make sure you have all the media types that you had before. In the MDMS V3 you can only have on media type with the same name. If you had two media types in SLS/MDMS V2.x with the same name, the second one is not created in the MDMS V3 database.
	Description	The conversion program does not add a description to this attribute. Type in a description for this attribute.
	Density	The density attribute is only changed when the DENS_x symbol in the TAPESTART.COM file is something other than COMP or NOCOMP. Check to make sure this is correct.
	Compaction	This attribute is set to YES if the DENS_x symbol in the TAPESTART.COM file is COMP. It is set to NO if the symbol is NOCOMP. Check to make sure this is right.
	Capacity	This attribute is set to the value of DENS_X from the TAPESTART.COM file if it is not defined as COMP or NOCOMP. Check to make sure this right.
Jukebox	Description	The conversion program does not put a description for this attribute. Type in a description to this attribute.
	Nodes	Make sure this list of nodes contains the nodes that can reach the robot.
	Location	Make sure this is the location where this jukebox is located.
	Disabled	The conversion program enables all jukeboxes. If you want this jukebox disabled, set this attribute to YES.
	Shared	The conversion program sets this attribute to NO. NO means that MDMS does not expect to compete with other applications for this jukebox. If MDMS is supposed to share this jukebox with other applications set this attribute to YES.
	Auto reply	The conversion program sets this attribute to NO. Make sure this is the way you want the jukebox to react.
	Access	The conversion program has no way of knowing what the access should be, therefore, it sets the access attribute to ALL. Make sure this is the access you want for this jukebox.
	Control	Make sure that the attribute is set to MRD if MRD controls the robot. If the robot is controlled by DCSC, this attribute should be set to DCSC.
	Robot	Make sure this is the robot for this jukebox.
	Slot count	You need to set the slot count. The conversion program has no way of finding out the slot count.
	Usage	Make sure the usage is correct for the type of jukebox you have. The conversion program has no way of finding out if the jukebox uses magazines or not. If this jukebox uses magazines, you will need to configure it.
Magazine	Description	The conversion program does not add a description to this attribute. Type a description for this attribute.
	Offsite location	The old magazine record had no offsite location, so you need to add this attribute.

Table M–3 Things to Look for After the Conversion

	Offsite date	The old magazine record had no offsite date, so you need to add this attribute.
	Onsite location	The old magazine record had no onsite location, so you need to add this attribute.
	Offsite date	The old magazine record had no onsite date, so you need to add this attribute.
Node	Description	The conversion program does not put a description in this attribute. Type a description for this attribute.
	DECnet-Plus fullname	TAPESTART.COM does not support DECnet-Plus, therefore the conversion program cannot put in the DECnet-Plus fullname attribute. If this node uses DECnet-Plus, you should set this attribute.
	TCP/IP fullname	TAPESTART.COM does not support TCP/IP, therefore the conversion program cannot put in the TCP/IP fullname attribute. If this node uses TCP/IP, you should set this attribute.
	Disabled	The conversion program sets the enabled attribute. Make sure you want this node enabled.
	Database server	If this attribute is set to YES, this node has the potential to become a database server. The logical MDMS$DATABASE_SERVERS must have this node name in is definition of nodes in the domain. This definition is defined in the SYS$STARTUP:MDMS$SYSTARTUP.COM file
	Location	Make sure this is the location that this node is located in. During the conversion it could have been changed depending on the TAPESTART.COM file or what the default was in the domain object at the time of creation.
	Opcom classes	This attribute is defined as the Opcom class in the domain object when the node was created. Make sure this is the Opcom class for this node.
	Transports	Make sure this is the transport you want. The conversion program has no way of knowing the transports you want so it takes the defaults.
POOL	Description	Make sure this is the description you want for this pool. This attribute is not filled in during the conversion program.
	Authorized users	Make sure that the comma separated list contains all of the authorized users for the pool. The users must be specified as NODE::user
	Default users	You need to set this attribute because conversion program does not set this attribute. The users must be specified as node::user.
VOLUME		The conversion program fills in all needed attributes from the old record. This is included so you will not think the volume object was forgotten.

M.4 Using SLS/MDMS V2.x Clients With the MDMS V3 Database

This section describes how older versions of SLS/MDMS can coexist with the new version of MDMS for the purpose of upgrading your MDMS domain. You may have versions of ABS, or HSM or SLS which are using SLS/MDMS V2 and which cannot be upgraded or replaced immediately. MDMS V3 provides limited support for older SLS/MDMS clients to make upgrading your MDMS domain to the new version as smooth as possible. This limited support allows rolling upgrade of all SLS/MDMS V2 nodes to MDMS V3. Also ABS and HSM version 3.0 and later have been modified to support either SLS/MDMS V2 or MDMS V2 to make it easy to switch over to the new MDMS. The upgrade procedure has been completed as soon as all nodes in your domain are running the new MDMS V3 version exclusively.

M.4.1 Limited Support for SLS/MDMS V2 during Rolling Upgrade

The major difference between SLS/MDMS V2 and MDMS V3 is the way information about objects and configuration is stored. To support the old version the new server can be set up to accept requests for DECnet object SLS$DB which was serving the database before. Any database request sent to SLS$DB will be executed and data returned compatible with old database client requests. This allows SLS/MDMS V2 database clients to still send their database requests to the new server without any change.

The SLS$DB function in the new MDMS serves and shares information for the following objects to a V2 database client:

- Volume information - previously stored in TAPEMAST.DAT
- Pool information - previously stored in POOLMAST.DAT
- Magazine information - previously stored in MAGAZINE.DAT
- Object information - not shared between the old and new MDMS:
- Drive information - previously stored in TAPESTART.COM
- Jukebox information - previously stored in TAPESTART.COM
- Media type information - previously stored in TAPESTART.COM
- Slot information - previously stored in SLOTMAST.DAT
- Node information - previously stored in NODE_VALIDATE.DAT

The new MDMS server keeps all its information in a per object database. The MDMS V3 installation process propagates definitions of the objects from the old database to the new V3 database. However, any changes made after the installation of V3 have to be carefully entered by the user in the old and new databases. Operational problems are possible if the databases diverge. Therefore it is recommended to complete the upgrade process as quickly as possible.

M.4.2 Upgrading the Domain to MDMS V3

Upgrading your SLS/MDMS V2 domain starts with the nodes, which have been defined as database servers in symbol DB_NODES in file TAPESTART.COM. Refer to the Installation Guide for details on how to perform the following steps.

Step 1. Shut down all SLS/MDMS database servers in your SLS/MDMS domain.

Step 2. Install version MDMS V3 on nodes, which have been acting as database servers before.

Step 3. When the new servers are up-and-running check and possibly change the configuration and database entries so that it matches your previous SLS/MDMS V2 setup

Step 4. Edit SYS$MANAGER:MDMS$SYSTARTUP.COM and make sure that:

- Logical name MDMS$DATABASE_SERVERS include this nodes DECnet (Phase IV) node name
- Logical name MDMS$PREV3_SUPPORT is set to TRUE to enable the SLS/MDMS V2 support function in the new server
- Logical name MDMS$VERSION3 is set to TRUE to direct ABS and/or HSM to use the new MDMS V3 interface

If you had to change any of the logical name settings above you have to restart the server using '@SYS$STARTUP:MDMS$STARTUP RESTART'. You can type the server's logfile to verify that the DECnet listener for object SLS$DB has been successfully started.

Step 5. To support load, unload and operator requests from old SLS/MDMS clients you have to edit SYS$MANAGER:TAPESTART.COM and change the line which defines DB_NODES to read like this:

```
$ DB_NODES = ""
```

This prevents a SLS/MDMS V2 server from starting the old database server process SLS$TAPMGRDB.

Step 6. Start SLS/MDMS V2 with @SYS$STARTUP:SLS$STARTUP.

Use a "STORAGE VOLUME" command to test that you can access the new MDMS V3 database.

Step 7. Now you are ready to start up ABS, HSM or SLS.

Note that no change is necessary for nodes running SLS/MDMS V2 as a database client. For any old SLS/MDMS client in your domain you have to add its node object to the MDMS V3 database. In V3 all nodes of an MDMS domain have to be registered (see command MDMS CREATE NODE). These clients can connect to a new MDMS DB server as soon as the new server is up and running and has been added to the new database.

A node with either local tape drives or local jukeboxes which are accessed from new MDMS V3 servers need to have MDMS V3 installed and running.

A node with either local tape drives or local jukeboxes, which are accessed from old SLS/MDMS V2 servers, need to have SLS/MDMS V3 running.

If access is required from both old and new servers then both versions need to be started on that node. But in all cases DB_NODES in all TAPESTART.COM needs to be empty.

M.4.3 Reverting to SLS/MDMS V2

MDMS V3 allows you to convert the MDMS V3 volume database back to the SLS/MDMS V2 TAPEMAST.DAT file. Any changes you did under MDMS V3 for pool and magazine objects need to be entered manually into V2 database. Any changes you did under MDMS V3 for drive, jukebox or media type objects need to be updated in file TAPESTART.COM.

The following steps need to be performed to revert back to a SLS/MDMS V2 only domain:

Step 1. Shut down all applications using MDMS (i.e., ABS, HSM and SLS)

Step 2. Shut down all MDMS V3 servers in the domain and deassign system logical name MDMS$VERSION3 on all nodes.

Step 3. .Convert the new database back to the old database files. Refer to section “Converting SLS/MDMS V2 Symbols and Database” for instructions.

Step 4. Edit TAPESTART.COM on all SLS/MDMS nodes, which should be database servers again. Add the node’s DECnet name to the symbol DB_NODES.

Step 5. Remove the call to MDMS$STARTUP.COM from your SYSTARTUP_VMS.COM.

Step 6. Make sure a call to SLS$STARTUP.COM is included in your SYSTARTUP_VMS.COM.

Step 7. Start up SLS/MDMS V2 and all applications using it.

M.4.4 Restrictions

During the rolling upgrade period, the following restrictions apply:

- Only the first media type of a volume object can be used by a SLS/MDMS V2 client.
- Node names must be exactly the nodes' DECnet (Phase IV) names.
- Some functions of old V2 utilities will not work. All updates to pools, slots, magazines and volumes should be preformed on a MDMS V3 node.

M.5 Convert from MDMS Version 3 to a V2.X Volume Database

This section describes how to convert the MDMS V3 volume database back to a SLS/MDMS V2.X volume database.

If for some reason, you need to convert back to SLS/MDMS V2.X a conversion command procedure is provided. This conversion procedure does not convert anything other than the volume database. If you have added new objects, you will have to add these to TAPESTART.COM or to the following SLS/MDMS V2.X database files:

- database authorization file (VALIDATE.DAT)
- Pool Authorization file (POOLAUTH.DAT)
- Slot Definition file (SLOTMAST.DAT)
- Volume Database file (TAPEMAST.DAT)
- Magazine Database file (SLS$MAGAZINE_MASTER_FILE.DAT)

To execute the conversion command procedure, type in the following command:

```
$ @MDMS$SYSTEM:MDMS$CONVERT_V3_TO_V2
```

After introductory information, this command procedure will ask you questions to complete the conversion.

N

Using ABS to Backup Oracle Databases

This appendix describes how to use ABS to backup Oracle databases using the Oracle Server Manager. When doing a backup, you can either do a closed database backup or an open database backup. This appendix contains an example database and examples of ABS environment policies, ABS storage policies, and ABS save requests that are used to backup the example database. The examples uses a jukebox that has three tape drives. Be sure to read the *Performing Operating System Backup and Recovery* section in the *Oracle Backup and Recovery Guide*.

The following sections are included in this appendix:

- Example Oracle Database
- Backing up a Closed Database (offline or cold backup)
- Backup up an Open Database (online or hot backup)

N.1 Example Oracle Database

Before getting started with describing how to use ABS to backup the Oracle database, let us first look at the example database so we know the name of the files to backup.

The following shows the tablespaces and datafiles:

```
SQL> select t.name "Tablespace", f.name "Datafile"
from v$tablespace t, v$datafile f where T.ts# = f.ts#
order by t.name;

Tablespace      Datafile
------------------------------------------------------------------

SYSTEM          DISK$ALPHA:[ORACLE.DB_PAYROLL]ORA_SYSTEM.DBS
TBS1            DISK$ORACLE1:[PAYROLL_TBS1]PAYROLL_TBS1.DF
TBS2            DISK$ORACLE2:[PAYROLL_TBS2]PAYROLL_TBS2.DF
TBS3            DISK$ORACLE3:[PAYROLL_TBS3]PAYROLL_TBS3.DF
TBS4            DISK$ORACLE4:[PAYROLL_TBS4]PAYROLL_TBS4.DF
TBS5            DISK$ORACLE5:[PAYROLL_TBS5]PAYROLL_TBS5.DF
TBS6            DISK$ORACLE6:[PAYROLL_TBS6]PAYROLL_TBS6.DF

7 rows selected.
```

The following shows the online redo log files:

```
SQL> select member from v$logfile;

MEMBER
-------------------------------------------------

DISK$ALPHA:[ORACLE.DB_PAYROLL]ORA_LOG1.RDO
DISK$ALPHA:[ORACLE.DB_PAYROLL]ORA_LOG2.RDO
```

The following shows the control files:

```
SQL>  select value from v$parameter where name = 'control_files';

VALUE
------------------------------

ora_control1, ora_control2
```

All of the files used to create the database are located in the following directory which is also a backup:

```
DISK$ALPHA:[ORACLE.DB_PAYROLL]
```

N.2 Backing up a Closed Database

When backing up a closed database, the database must be shutdown. After shutting down the database, all of the database files can be backed up before starting up the database again. To accomplish this task, the following example creates an environment policy that has a prologue file that shutsdown the database. The environment policy also has an epilogue file that restarts the database. This example has three save requests. Each save request will execute on a separate tape drive.

The following sections show the environment policy, storage policy, and save requests for the closed database backup.

N.2.1 Creating ABS Environment and Storage Policies for a Closed Database Backup

The first thing to create is an environment policy, ORA_CLOSED_BACKUP_ENV. The environment has a prologue file and epilogue file which is only run once for each save request. In a save request, the prologue and epilogue files are run for each save object. That would not work, so the prologue and epilogue files are put in the environment file. These prologue and epilogue files have if statements that execute different code for each save request.

The following shows the creation of the ORA_CLOSED_BACKUP_ENV environment:

```
$ ABS CREATE ENVIRONMENT ORA_CLOSED_BACKUP_ENV -
/PROLOGUE=@DISK$ALPHA:[ORACLE.COM]ORA_CLOSED_BACKUP_PROLOG -
/EPILOGUE=@DISK$ALPHA:[ORACLE.COM]ORA_CLOSED_BACKUP_EPILOG

$ ABS SHOW ENV ORA_CLOSED_BACKUP_ENV/FULL

Execution Environment

Name      - ORA_CLOSED_BACKUP_ENV
Version   - 1
UID       - 883D0028-226A-11D4-AD9F-474F4749524C
Data Safety Options - CRC_VERIFICATION
Listing Option - NO_LISTING
Span Filesystem Options - NO FILESYSTEM SPAN
Symbolic Links Option - LINKS_ONLY
Compression Options - None
Original Object Action - NO_CHANGE

User Profile

Node - CURLEY
Cluster -
User - ABS
Privs - SETPRV,TMPMBX,OPER,NETMBX
Access Right - ORA_DBA
Owner     - CURLEY::DBA
```

```
Access Right - CURLEY::DBA
Access Granted - READ, WRITE, SET, SHOW, DELETE, CONTROL

Notification Method   - NO_NOTIFICATION
Notification Receiver - TAPES
Notification When     - FATAL
Notification Type     - BRIEF

Notification Method - MAIL
Notification List   - DBA
Notification Reason - COMPLETE
Notification Type   - BRIEF

Locking Options  - None
Number of Drives - 1
Retry Count      - 3
Retry Interval   - 15

Prologue Command - @DISK$ALPHA:[ORACLE.COM]ORA_CLOSED_BACKUP_PROLOG
Epilogue Command - @DISK$ALPHA:[ORACLE.COM]ORA_CLOSED_BACKUP_EPILOG
```

The prologue and epilogue files have logic in them that only executes when the save request is ORA_CLOSED_BACKUP1. Because the environment is executed for each save request, the database should not be shutdown or started up for each save request. The shutdown and startup of the database is only going to happen when the save request is named ORA_CLOSED_BACKUP1.

The logic also starts the other two save requests:

- ORA_CLOSED_BACKUP2
- ORA_CLOSED_BACKUP3

The following is the prologue file, ORA_CLOSED_BACKUP_PROLOG.COM:

```
$!
$! THIS COMMAND PROCEDURE SHUTSDOWN THE ORACLE DATABASE
$! SO THAT A CLOSED DATABASE BACKUP CAN BE COMPLETED
$!
$! IT THEN STARTS ALL OF THE BACKUPS OF THE DATABASE AND
$! THEN SYNCHRONIZES ON THEM TO WAIT UNTIL THEY ARE FINISHED
$!
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .NES. "ORA_CLOSED_BACKUP1")
$ THEN
$   EXIT
$ ENDIF
$ @DISK$ALPHA:[ORACLE.DB_PAYROLL]ORAUSER_PAYROLL J3944
$ SVRMGRL @SYS$INPUT
SET ECHO ON
CONNECT INTERNAL AS SYSDBA
SHUTDOWN IMMEDIATE
EXIT
$!
$! START THE BACKUP OF DATABASE
$ ABS SET SAVE/START ORA_CLOSED_BACKUP2
$ ABS SET SAVE/START ORA_CLOSED_BACKUP3
$ EXIT
```

The epilogue file, ORA_CLOSED_BACKUP_EPILOG.COM, also has logic defined to wait for the other two save requests to finish before the database is started. The following is the epilogue file, ORA_CLOSED_BACKUP_EPILOG.COM:

```
$!
$! THIS COMMAND PROCEDURE STARTS UP THE ORACLE DATABASE
$! AT THE COMPLETION OF THE ORACLE DATABASE BACKUP
$!
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .NES. "ORA_CLOSED_BACKUP1")
$ THEN
```

```
$   EXIT
$ ENDIF
$!
$!  NOW SYNCHRONIZE TO MAKE SURE ALL BACKUPS ARE DONE BEFORE RESTARTING
$!  THE DATABASE
$!
$ ABS SYNCHRONIZE ORA_CLOSED_BACKUP2
$ ABS SYNCHRONIZE ORA_CLOSED_BACKUP3
$!
$ @DISK$ALPHA:[ORACLE.DB_PAYROLL]ORAUSER_PAYROLL J3944
$ SVRMGRL
SET ECHO ON
CONNECT INTERNAL AS SYSDBA
STARTUP
EXIT
$ EXIT
```

The storage policy, ORA_CLOSED_BACKUP_SP, supports the use of three drives at one time. The following shows the creation of the storage policy:

```
$ ABS CREATE STORAGE_CLASS ORA_CLOSED_BACKUP_SP -
/MAXIMUM_SAVES=3 -
/TYPE_OF_MEDIA=TK89

$ ABS SHOW STORAGE_CLASS ORA_CLOSED_BACKUP_SP /full

Storage Class

Name      - ORA_CLOSED_BACKUP_SP
Version   - 1
UID       - 889E98D3-226A-11D4-ADE2-474F4749524C
Execution Node Name - CURLEY
Archive File System
Primary Archive Location    -
Staging Location    -
Primary Archive Type         - SLS/MDMS
Owner     - CURLEY::DBA
Access Right - CURLEY::DBA
Access Granted - READ, WRITE, SET, SHOW, DELETE, CONTROL
Tape Pool - None
Volume Set Name     -
Retention Period -  365
Consolidation Criteria
Count    - 0
Size     - 0
Interval -    7
Catalog Name          - ABS_CATALOG
Maximum Saves  - 3
Media Management Info
Media Location        - None
Type of Media  - TK89
Drive List     - None
```

N.2.2 Creating ABS Save Requests for a Closed Database Backup

Since this example is using three tape drives there are also three save requests to allow multiple stream operation. All three save requests backup approximately the same amount of data to keep all three drives in use. The first save request, ORA_CLOSED_BACKUP1, is scheduled daily and backups all of the following files:

- ORA_SYSTEM.DBS
- Redo log files
- Archived redo log files
- Database initialization files

- Table space 1 and table space 2

The ORA_CLOSED_BACKUP1 save request is scheduled each day in this example to start at 23:00. When it starts, the environment policy runs its prologue which shuts down the database and starts the other two save requests. The other two save requests are scheduled on demand.

The other two save requests, ORA_CLOSED_BACKUP2 and ORA_CLOSED_BACKUP3, are scheduled on demand and backup two table spaces each.

The following shows the creation of the three save requests and then the scheduling of ORA_CLOSED_BACKUP1 at 23:00.

```
$!
$! CREATE SAVE REQUESTS
$!
$ ABS SAVE /NOSTART DISK$ALPHA:[ORACLE.DB_PAYROLL]ORA_SYSTEM.DBS -
/NAME=ORA_CLOSED_BACKUP1 -
/ENVIRONMENT=ORA_CLOSED_BACKUP_ENV -
/SCHEDULE_OPTION=DAILY -
/STORAGE_CLASS=ORA_CLOSED_BACKUP_SP

$ ABS SET SAVE ORA_CLOSED_BACKUP1 -
DISK$ALPHA:[ORACLE.DB_PAYROLL]ORA_LOG*.RDO/ADD, -
DISK$ALPHA:[ORACLE.DB_PAYROLL]*.ARC/ADD, -
DISK$ALPHA:[ORACLE.DB_PAYROLL]*.ORA/ADD

$ ABS SET SAVE ORA_CLOSED_BACKUP1 -
DISK$ORACLE1:[PAYROLL_TBS1]PAYROLL_TBS1.DF/ADD,
DISK$ORACLE2:[PAYROLL_TBS2]PAYROLL_TBS2.DF/ADD

$!
$ ABS SAVE /NOSTART DISK$ORACLE3:[PAYROLL_TBS3]PAYROLL_TBS3.DF -
/NAME=ORA_CLOSED_BACKUP2 -
/SCHEDULE_OPTION=ON_DEMAND -
/ENVIRONMENT=ORA_CLOSED_BACKUP_ENV -
/STORAGE_CLASS=ORA_CLOSED_BACKUP_SP


$ ABS SET SAVE ORA_CLOSED_BACKUP2 -
DISK$ORACLE4:[PAYROLL_TBS4]PAYROLL_TBS4.DF/ADD

$!
$ ABS SAVE /NOSTART DISK$ORACLE5:[PAYROLL_TBS5]PAYROLL_TBS5.DF -
/NAME=ORA_CLOSED_BACKUP3 -
/SCHEDULE_OPTION=ON_DEMAND -
/ENVIRONMENT=ORA_CLOSED_BACKUP_ENV -
/STORAGE_CLASS=ORA_CLOSED_BACKUP_SP


$ ABS SET SAVE ORA_CLOSED_BACKUP3 -
DISK$ORACLE6:[PAYROLL_TBS6]PAYROLL_TBS6.DF/ADD

$!
$!  NOW TO START IT
$!
$ ABS SET SAVE ORA_CLOSED_BACKUP1/START="23:00"
```

N.3 Backing Up an Open Database

Backing up an open database allows users to have normal access to all online tablespaces during backup. During the backup of an online tablespace, the Oracle Server Manager must be notified at the beginning of the backup and at the end of the backup. To accomplish this task, the following example creates an environment policy that has a prologue file that notifies the Oracle Server Manager that a backup of tablespace is about to begin. Also, the environment policy has an epilogue file that notifies the Oracle Server Manager that a backup of the tablespace has ended.

This example has the following save requests:

- ORA_OPEN_BACKUP - is the controlling save request. It also backs up tablespaces TBS1 and TBS2.
- ORA_OPEN_BACKUP_SYS - backs up the system tablespace.
- ORA_OPEN_BACKUP_TBS3 - backs up tablespace TBS3.
- ORA_OPEN_BACKUP_TBS4 - backs up tablespace TBS4.
- ORA_OPEN_BACKUP_TBS5 - backs up tablespace TBS5.
- ORA_OPEN_BACKUP_TBS6 - backs up tablespace TBS6.
- ORA_OPEN_BACKUP_RDO - backs up the archived redo logs.
  Before backing up the logs, it archives the redo log. Also, it backs up the control file.

N.3.1 Creating ABS Environment and Storage Policies for an Open Database Backup

The first thing to create is an environment policy, ORA_OPEN_BACKUP_ENV. The environment has a prologue file and epilogue file which is only run once for each save request. These prologue and epilogue files have if statements that execute different code for each save request.

The following shows the creation of the ORA_OPEN_BACKUP_ENV environment:

```
$!
$! CREATE ENVIRONMENT POLICY
$!
$ ABS CREATE ENVIRONMENT ORA_OPEN_BACKUP_ENV -
/PROLOGUE=@DISK$ALPHA:[ORACLE.com]ORA_OPEN_BACKUP_PROLOG -
/EPILOGUE=@DISK$ALPHA:[ORACLE.com]ORA_OPEN_BACKUP_EPILOG

$ ABS SHOW ENV ORA_OPEN_BACKUP_ENV/FULL


Execution Environment

Name      - ORA_OPEN_BACKUP_ENV
Version   - 1
UID       - F253D0EC-2A42-11D4-942B-474F4749524C
Data Safety Options - CRC_VERIFICATION
Listing Option - NO_LISTING
Span Filesystem Options - NO FILESYSTEM SPAN
Symbolic Links Option - LINKS_ONLY
Compression Options - None
Original Object Action - NO_CHANGE
User Profile
Node - CURLEY
Cluster -
User - ABS
Privs - SETPRV,TMPMBX,OPER,NETMBX
Access Right - ORA_DBA
Owner     - CURLEY::DBA
Access Right - CURLEY::DBA
Access Granted - READ, WRITE, SET, SHOW, DELETE, CONTROL
Notification Method   - NO_NOTIFICATION
Notification Receiver - TAPES
Notification When     - FATAL
Notification Type     - BRIEF
Locking Options  - None
Number of Drives - 1
Retry Count      - 3
Retry Interval   - 15
Prologue Command - @DISK$ALPHA:[ORACLE.COM]ORA_OPEN_BACKUP_PROLOG
Epilogue Command - @DISK$ALPHA:[ORACLE.COM]ORA_OPEN_BACKUP_EPILOG
```

The prologue and epilogue files have logic in them that executes different code for each save request. In the save requests that backup tablespaces, the code in the prologue file runs SQL code

that notifies the Oracle Server Manager that a backup of the tablespace is about to begin. In the save requests that backup tablespaces, the code in the epilogue file runs SQL code that notifies the Oracle Server Manager that a backup of the tablespace has ended. For the save, ORA_OPEN_BACKUP, the logic deletes the old copy of the control file and starts all of the other save requests except ORA_OPEN_BACKUP_RDO.

The following is the prologue file, ORA_OPEN_BACKUP_PROLOG.COM:

```
$!
$! THIS COMMAND PROCEDURE STARTS UP ALL OF THE SAVE REQUESTS FOR
$! AN OPEN DATABASE BACKUP IF THIS SAVE REQUEST IS
$! ORA_OPEN_BACKUP.
$!
$! ALSO, IT BEGINS THE DATABASE BACKUP TABLESPACE TBS1 AND TBS2
$! IT THEN STARTS ALL OF THE BACKUPS OF THE DATABASE AND
$! THEN SYNCHRONIZES ON THEM TO WAIT UNTIL THEY ARE FINISHED
$!
$ @DISK$ALPHA:[ORACLE.DB_APITEST]ORAUSER_APITEST J3944
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP")
$ THEN
$   !
$   !  DELETE THE COPY OF THE CONTROL FILE, WE WILL MAKE ANOTHER COPY
$   !  AFTER WE GET THROUGH
$   !
$   IF(F$SEARCH("DISK$ALPHA:[ORACLE.DB_APITEST]COPY_OF_CONTROL_FILE.CON")
.NES. "")
$   THEN
$      DELETE/NOCONFIRM
DISK$ALPHA:[ORALE.DB_APITEST]COPY_OF_CONTROL_FILE.CON;*
$   ENDIF
$   !
$   ! START OTHER SAVE REQUESTS
$   !
$   ABS SET SAVE/START ORA_OPEN_BACKUP_SYS
$   ABS SET SAVE/START ORA_OPEN_BACKUP_TBS3
$   ABS SET SAVE/START ORA_OPEN_BACKUP_TBS4
$   ABS SET SAVE/START ORA_OPEN_BACKUP_TBS5
$   ABS SET SAVE/START ORA_OPEN_BACKUP_TBS6
$   !
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.LOG]ORA_OPEN_BACKUP_PROLOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS1 BEGIN BACKUP;
ALTER TABLESPACE TBS2 BEGIN BACKUP;
EXIT
$   EXIT
$ ENDIF
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_SYS")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.LOG]ORA_OPEN_BACKUP_SYSTEM_PROLOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE SYSTEM BEGIN BACKUP;
EXIT
$   EXIT
$ ENDIF
$ !
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_TBS3")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.LOG]ORA_OPEN_BACKUP_TBS3_PROLOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS3 BEGIN BACKUP;
EXIT
$   EXIT
$ ENDIF
```

```
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_TBS4")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.LOG]ORA_OPEN_BACKUP_TBS4_PROLOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS4 BEGIN BACKUP;
EXIT
$   EXIT
$ ENDIF
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_TBS5")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.LOG]ORA_OPEN_BACKUP_TBS5_PROLOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS5 BEGIN BACKUP;
EXIT
$   EXIT
$ ENDIF
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_TBS6")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.LOG]ORA_OPEN_BACKUP_TBS6_PROLOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS6 BEGIN BACKUP;
EXIT
$   EXIT
$ ENDIF
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_RDO")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.LOG]ORA_OPEN_BACKUP_RDO_PROLOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER SYSTEM ARCHIVE LOG CURRENT;
ALTER SYSTEM SWITCH LOGFILE;
ALTER DATABASE BACKUP CONTROLFILE TO
'DISK$ALPHA:[ORACLE.DB_APITEST]COPY_OF_CONTROL_FILE.CON';
EXIT
$   EXIT
$ ENDIF
$ !
$ EXIT
```

The epilogue file, ORA_OPEN_BACKUP_EPILOG.COM, also has logic defined to execute different code depending on the save request. All save requests that back up tablespaces executes code that notifies the Oracle Server Manager that the backup of the tablespace has ended. If the save request is ORA_OPEN_BACKUP, it waits on all of the jobs to complete and then it starts the ORA_OPEN_BACKUP_RDO save request to archive the redo log files and make a copy of the control file to backup.

The following is the example of ORA_OPEN_BACKUP_EPILOG.COM:

```
$!
$ @DISK$ALPHA:=:[ORACLE.DB_APITEST]ORAUSER_APITEST J3944
$set verify
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.log]ORA_OPEN_BACKUP_EPILOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS1 END BACKUP;
ALTER TABLESPACE TBS2 END BACKUP;
```

```
EXIT
$   ABS SYNCHRONIZE ORA_OPEN_BACKUP_SYS
$   ABS SYNCHRONIZE ORA_OPEN_BACKUP_TBS3
$   ABS SYNCHRONIZE ORA_OPEN_BACKUP_TBS4
$   ABS SYNCHRONIZE ORA_OPEN_BACKUP_TBS5
$   ABS SYNCHRONIZE ORA_OPEN_BACKUP_TBS6
$   !
$   ! NOW FINISH UP WITH ARCHIVING THE REDO FILES
$   !
$   ABS SET SAVE/START ORA_OPEN_BACKUP_RDO
$   WAIT 00:01
$   ABS SYNCHRONIZE ORA_OPEN_BACKUP_RDO
$ set noverify
$   EXIT
$ ENDIF
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_SYS")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.log]ORA_OPEN_BACKUP_SYSTEM_EPILOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE SYSTEM END BACKUP;
EXIT
$ set noverify
$   EXIT
$ ENDIF
$ !
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_TBS3")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.log]ORA_OPEN_BACKUP_TBS3_EPILOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS3 END BACKUP;
EXIT
$ set noverify
$   EXIT
$ ENDIF
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_TBS4")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.log]ORA_OPEN_BACKUP_TBS4_EPILOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS4 END BACKUP;
EXIT
$ set noverify
$   EXIT
$ ENDIF
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_TBS5")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.log]ORA_OPEN_BACKUP_TBS5_EPILOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS5 END BACKUP;
EXIT
$ set noverify
$   EXIT
$ ENDIF
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_TBS6")
$ THEN
$   SVRMGRL @SYS$INPUT
SET ECHO ON
SPOOL DISK$ALPHA:[ORACLE.log]ORA_OPEN_BACKUP_TBS6_EPILOG.LOG
CONNECT INTERNAL AS SYSDBA
ALTER TABLESPACE TBS6 END BACKUP;
EXIT
```

```
$ set noverify
$   EXIT
$ ENDIF
$ !
$ IF ( F$TRNLNM("ABS_SAVE_REQUEST_NAME") .EQS. "ORA_OPEN_BACKUP_RDO")
$ THEN
$!   DELETE/NOCONFIRM DISK$ALPHA:[ORACLE.DB_APITEST]*.ARC;*
$ set noverify
$    EXIT
$ ENDIF
$ set noverify
$ EXIT
```

The storage policy, ORA_OPEN_BACKUP_SP, supports the use of three drives at one time. The following shows the creation of the storage policy:

```
$!
$! CREATE STORAGE POLICY
$!
$ ABS CREATE STORAGE_CLASS ORA_OPEN_BACKUP_SP -
/MAXIMUM_SAVES=3 -
/TYPE_OF_MEDIA=TK89

$ ABS SHOW STORAGE_CLASS ORA_OPEN_BACKUP_SP/FULL

Storage Class

Name      - ORA_OPEN_BACKUP_SP
Version   - 1
UID       - F28D2E91-2A42-11D4-9453-474F4749524C
Execution Node Name - CURLEY
Archive File System
Primary Archive Location    -
Staging Location    -
Primary Archive Type         - SLS/MDMS
Owner     - CURLEY::DBA
Access Right - CURLEY::DBA
Access Granted - READ, WRITE, SET, SHOW, DELETE, CONTROL
Tape Pool - None
Volume Set Name     -
Retention Period -  365
Consolidation Criteria
Count    - 0
Size     - 0
Interval -    7 00:00:00
Catalog Name          - ABS_CATALOG
Maximum Saves  - 3
Media Management Info
Media Location        - None
Type of Media  - TK89
Drive List     - None
```

N.3.2 Creating ABS Save Requests for an Open Database Backup

This section shows the commands necessary to create the ABS save requests for an open database backup. These save requests are implemented for a jukebox that has three tape drive. The controlling save request, ORA_OPEN_BACKUP, is the only save request that is scheduled, the rest have a scheduling policy of ON_DEMAND. The ORA_OPEN_BACKUP save request backs up tablespaces TBS1 and TBS2. This keeps one tape drive busy while the other two backup the rest of the database. After all tablespaces are backed up, the ORA_OPEN_BACKUP save request starts the ORA_OPEN_BACKUP_RDO save request to archive the redo log files and then backup the archived log files and the control file.

The following shows the creation of the save requests and the scheduling of ORA_OPEN_BACKUP at 23:00.

```
$!
$! CREATE SAVE REQUESTS
$!
$ ABS SAVE /NOSTART DISK$ALPHA:[ORACLE.DB_APITEST]ORA_SYSTEM.DBS -
/AGENT_QUALIFIER="/IGNORE=(INTERLOCK,NOBACKUP)" -
/NAME=ORA_OPEN_BACKUP_SYS -
/ENVIRONMENT=ORA_OPEN_BACKUP_ENV -
/SCHEDULE_OPTION=DAILY -
/STORAGE_CLASS=ORA_OPEN_BACKUP_SP
$!
$ ABS SAVE /NOSTART DISK$ORACLE3:[APITEST_TBS3]APITEST_TBS3.DF -
/AGENT_QUALIFIER="/IGNORE=(INTERLOCK,NOBACKUP)" -
/NAME=ORA_OPEN_BACKUP_TBS3 -
/SCHEDULE_OPTION=ON_DEMAND -
/ENVIRONMENT=ORA_OPEN_BACKUP_ENV -
/STORAGE_CLASS=ORA_OPEN_BACKUP_SP
$!
$ ABS SAVE /NOSTART DISK$ORACLE4:[APITEST_TBS4]APITEST_TBS4.DF -
/AGENT_QUALIFIER="/IGNORE=(INTERLOCK,NOBACKUP)" -
/NAME=ORA_OPEN_BACKUP_TBS4 -
/SCHEDULE_OPTION=ON_DEMAND -
/ENVIRONMENT=ORA_OPEN_BACKUP_ENV -
/STORAGE_CLASS=ORA_OPEN_BACKUP_SP
$!
$ ABS SAVE /NOSTART DISK$ORACLE5:[APITEST_TBS5]APITEST_TBS5.DF -
/AGENT_QUALIFIER="/IGNORE=(INTERLOCK,NOBACKUP)" -
/NAME=ORA_OPEN_BACKUP_TBS5 -
/SCHEDULE_OPTION=ON_DEMAND -
/ENVIRONMENT=ORA_OPEN_BACKUP_ENV -
/STORAGE_CLASS=ORA_OPEN_BACKUP_SP
$!
$ ABS SAVE /NOSTART DISK$ORACLE6:[APITEST_TBS6]APITEST_TBS6.DF -
/AGENT_QUALIFIER="/IGNORE=(INTERLOCK,NOBACKUP)" -
/NAME=ORA_OPEN_BACKUP_TBS6 -
/SCHEDULE_OPTION=ON_DEMAND -
/ENVIRONMENT=ORA_OPEN_BACKUP_ENV -
/STORAGE_CLASS=ORA_OPEN_BACKUP_SP
$!
$ ABS SAVE /NOSTART DISK$ORACLE1:[APITEST_TBS1]APITEST_TBS1.DF -
/NAME=ORA_OPEN_BACKUP -
/AGENT_QUALIFIER="/IGNORE=(INTERLOCK,NOBACKUP)" -
/SCHEDULE_OPTION=DAILY -
/ENVIRONMENT=ORA_OPEN_BACKUP_ENV -
/STORAGE_CLASS=ORA_OPEN_BACKUP_SP
$ ABS SET SAVE ORA_OPEN_BACKUP -
DISK$ORACLE2:[APITEST_TBS2]APITEST_TBS2.DF/ADD
$!
$ ABS SAVE /NOSTART DISK$ALPHA:[ORACLE.DB_APITEST]*.ARC -
/NAME=ORA_OPEN_BACKUP_RDO -
/SCHEDULE_OPTION=ON_DEMAND -
/ENVIRONMENT=ORA_OPEN_BACKUP_ENV -
/STORAGE_CLASS=ORA_OPEN_BACKUP_SP
$ ABS SET SAVE ORA_OPEN_BACKUP_RDO -
DISK$ALPHA:[ORACLE.DB_APITEST]COPY_OF_CONTROL_FILE.CON/ADD
$!
$!  NOW TO START IT
$!
$ ABS SET SAVE ORA_OPEN_BACKUP/START="23:00"
```

Glossary

This glossary contains terms defined for the Archive Backup System for OpenVMS (ABS). It also contains terms associated with the following products when related to ABS:

- Media and Device Management Services for OpenVMS (MDMS)
- Storage Library System for OpenVMS (SLS)

absolute time

A data-entry format for specifying the date or time of day. The format for absolute time is [dd-mmm-yyyy[:]][hh:mm:ss.cc]. You can specify a date and time, or use the keywords TODAY, TOMORROW, or YESTERDAY.

access port

The port on a DCSC-controlled silo where volumes can be inserted into the silo.

active server process

The MDMS server process that is currently active. The active server process responds to requests issued from an MDMS client process.

allocate

To reserve something for private use. In MDMS software, a user is able to allocate volumes or drives.

allocated

The state of a drive or volume when a process is granted exclusive use of that drive or volume. The drive or volume remains allocated until the process gives up the allocation.

allocated state

One of four volume states. Volumes that are reserved for exclusive use by a user (such as ABS) are placed in the allocated state. Allocated volumes are available only to the user name (such as ABS) assigned to that volume.

ANSI

The abbreviation for the American National Standards Institute, an organization that publishes computer industry standards.

ANSI-labeled

A magnetic tape that complies with the ANSI standards for label, data, and record formats. The format of VMS ANSI-labeled magnetic tape volumes is based on Level 3 of the ANSI standard for magnetic tape labels and file structure.

archive

A repository of data that consists of

- Volumes that contains zero or more archive files.
- One or more catalogs that contain information about archived data that is stored on volumes.
- A set of services used to define the storage environment configuration and site policy. These services are also used to move data between the ABS client and the MDMS volume.

archive file system

The file system that contains the archived data.

archive object

The data object that resides in offline storage.

archiving

Saving data for the purpose of long-term storage.

ASCII

The abbreviation for the American Standard Code for Information Interchange.

This code is a set of 8-bit binary numbers representing the alpha- bet, punctuation, numerals, and other special symbols used in text representation and communications protocols.

back up

To make duplicate copies of one or more files, usually onto different media than the original media. This provides the availability to restore the original data if it is lost or corrupted.

backup agent

The client or utility that performs the actual save or restore operation. Examples are the VMS BACKUP Utility and the RMU Backup Utility.

backup engine

The backup engine moves data to and from the storage policy. Examples of backup engines: VMS BACKUP, RMU BACKUP, and UBS.

BACKUP format

Standard OpenVMS BACKUP format. The BACKUP format is the recording format used by the VMS Backup utility to back up data to save sets.

backup management domain

A node or OpenVMS Cluster system that has control over creating save requests. A backup management domain is usually controlled by a single storage administrator.

bind

The act of logically binding volumes into a magazine. This makes the volumes a logical unit that cannot be separated unless an UN- BIND operation is done on the volumes.

blocking factor

The number of records in a physical tape block. The length of a physical block written to magnetic tape is determined by multiplying the record length by the blocking factor. For example, if a record length of 132 and a blocking factor of 20 are specified, the length of each physical block written to tape will be 2640 bytes (or characters).

The blocking factor is only used when MDMS software is writing an EBCDIC tape.

catalog

Contains records of data movement operations. Each time a save request is initiated, the history of the data movement operation is recorded in an associated ABS central security domain: The node or OpenVMS Cluster system where ABS policy server is installed. This domain controls all ABS pol- icy objects, particularly storage and environment policies.

client node

Client nodes send database requests to the server node.

combination time:

A data-entry format for specifying date and time. Combination time consists of an absolute time value plus or minus a delta time value.

Examples:

"TODAY+7-"	Indicates current date plus seven days
"TODAY+7"	Indicates current date plus seven hours
"TOMORROW-1"	Indicates current date at 23:00 hours

command

An instruction, generally an English word, entered by the user at a terminal. The command requests the software to perform a pre- defined function.

CRC

The acronym for cyclic redundancy check. It is a verification process used to ensure data is correct.

consolidation count

The criteria under which ABS creates new volume sets.

consolidation interval

The number of days (in VMS time format) between the creation of new volume sets.

consolidation size

The desired maximum number of volumes allowed in a volume set.

data object

A data object specification, such as an OpenVMS file name or an Rdb/VMS database file name.

data movement request

Either a save or restore request initiated through either the DCL command interface or ABS graphical user interface.

deallocate

To relinquish ownership of a drive, volume, or volume set.

- When a drive is deallocated, it is then available for allocation by other processes.
- When a volume set is deallocated, it is either immediately available for allocation by other users or moved into a transition state.

deassign date

The day on which an allocated volume is scheduled to go into the transition state or the free state.

default

A value or operation automatically included in a command or field unless the user specifies differently.

density

The number of bits per inch (bpi) on magnetic tape. Typical values are 6250 bpi and 1600 bpi.

device

A physical device, such as a tape drive or disk device.

down state

One of four volume states. Volumes that are either damaged, lost, or temporarily removed from the MDMS volume database for cleaning are placed in the down state.

EBCDIC

Extended Binary Coded Decimal Interchange Code. EBCDIC is an unlabeled IBM recording format. Volumes in EBCDIC format do not have records in the MDMS volume database.

environment policy

ABS policy object that defines the environment in which data ABS save and restore requests occur.

expiration

The date and time at which an archived data is no longer considered useful. The archived data can be deleted and its space removed.

format

See recording format.

free state

The volume state that allows volumes to be selected by users or other software applications.

GUI

Graphical User Interface

in port

The physical opening in a jukebox where volumes can be imported.

interface

A shared physical or logical boundary between computing system components. Interfaces are used for sending and/or accepting information and control between programs, machines, and people.

inventory

The act of automatically updating the MDMS database. MDMS can mount each volume located in a magazine and update the MDMS volume database through this process.

I/O station

A jukebox component that enables an operator to manually insert and retrieve volumes. The I/O station consists of an I/O station door on the outside of the jukebox and an I/O station slot on the inside. See also I/O station door and I/O station slot.

I/O station door

An actual door on the outside of the jukebox that can be opened and closed. Behind the I/O station door is the I/O station slot.

I/O station slot

An I/O slot that holds a volume when it is entering or leaving the jukebox.

label

- Information recorded at a fixed location on the volume that identifies the volume to software.
- The physical printed label attached to the outside of the volume to identify it.

labeled

A recording format that includes a volume label.

LEBCDIC

Labeled EBCDIC format. See also EBCDIC.

load

The process which makes a volume physically available to the computer system, such as for read or write operations.

local symbol

A symbol meaningful only to the module or DCL command procedure that defines it.

log file

Any file into which status and error messages are written to reflect the progress of a process.

MDMS server node

The active server node to which all MDMS database requests are sent to be serviced. In a high-availability configuration, when the active server node fails, another node (see MDMS standby server process) in the OpenVMS Cluster system becomes the active server node.

MDMS software

The MDMS software is an OpenVMS software service that enables you to implement media and device management for your storage management operations. MDMS provides services to SLS, ABS, and HSM.

MDMS standby server process

Any MDMS server process that is not currently active. The standby server process waits and becomes active if the active server process fails.

magazine

A physical container that holds from 5 to 11 volumes. The magazine contains a set of logically bound volumes that reside in the MDMS database.

magazine database

The MDMS database that contains the magazine name and the volume names associated with that magazine.

media

A mass storage unit. Media is referred to in this document as a volume. Volumes provide a physical surface on which data is stored. Examples of physical volumes are magnetic tape, tape cartridge, and optical cartridge.

media type

A set of site-specific names associated with volume densities and drives.

nearline storage

Storage in which file headers are accessible through the operating system, but accessing data requires extra intervention.

Nearline storage employs a robotic device to move volumes between drives and volume storage locations. Nearline storage is less costly for each megabyte of data stored. Access times for data in nearline storage may vary. Access to data may be nearly instantaneous when a volume containing the data is already loaded in a drive. The time required for a robotic device to move to the most distant storage location, retrieve a volume, load it into a drive, and position the volume determines the maximum access time.

The devices of nearline storage technology include, but are not limited to, automated tape libraries and optical jukeboxes.

offline storage

Storage in which neither the file headers nor the data is accessible by the operating system and requires extra intervention.

Offline storage requires some type of intervention to move volumes between drives and the volumes' storage location. Offline storage is the least costly for each megabyte of data stored. Access times for data in offline storage vary for the same reasons as described for nearline storage. For archive data stored in a remote vault, access time can take more than a day.

The devices of offline storage technology include, but are not limited to, standalone tape drives, optical disk drives, and sequential stack loader tape drives.

online storage

Storage in which file headers and data can be accessed through the operating system. Online storage is the most costly for each megabyte of data stored.

As a trade off, online storage also offers the highest access performance. Online storage devices offer continuous service. The devices of online storage technology include disk storage and electronic (RAM) storage that uses disk I/O channels.

OPCOM

OpenVMS Operator Communication Manager. An online communication tool that provides a method for users or batch jobs to request assistance from the operator, and allows the operator to send messages to interactive users.

OPER privilege

The level of privilege required by a system operator to suspend an MDMS operation and to perform a variety of maintenance procedures on volumes, as well as archive files and saved system files.

out port

The physical opening in a jukebox where volumes can be exported from the jukebox.

policy

The decisions and methods in which you implement your ABS policy. This includes when and how often you back up or archive data from online to nearline or offline storage.

policy engine

The component in ABS that makes intelligent decisions based upon the implementation of your ABS policy.

policy objects

The method in which ABS enables you to implement your ABS policy. ABS provides the following policy objects:

- Storage policy
- Environment policy
- Save request
- Restore request

policy server

ABS server component. Placement of this component determines the central security domain (CSD).

pool

A set of volumes in the free state. Those volumes can be allocated by users who have access to the volume pool. The storage administrator creates and authorizes user access to pools.

record

A set of related data treated as a unit of information. For example, each volume that is added to the MDMS volume database has a record created that contains information about the volume.

record length

The length of a record in bytes. See also blocking factor.

recorded label

The label recorded on the volume.

recording format

The unique arrangement of data on a volume according to a predetermined standard. Examples of recording format are BACKUP, EBCDIC, and ANSI.

restore process

The method by which the contents of a file or disk are recovered from a volume or volumes that contain the saved data. ABS software will restore data by querying ABS catalog for the file or disk name specified in the restore request, and then locate the BACKUP save sets from one or more volumes, extract the data from those save sets, and place the information onto a Files-11 structured disk where the restored data can be accessed by a user.

restore request

A request to restore data from the archives to either its original location or an alternate location. Restore re- quests are initiated either through the DCL command interface or ABS graphical user interface.

requester

The user who creates a save or restore request.

requester profile

The requester profile is the profile of the user who is creating the save or restore request. This profile is captured at the time the request is created.

restore request

ABS policy object that defines the request for the restoration of data.

robot device

A tape or optical drive that provides automatic loading of volumes, such as a TF867 or a TL820.

save process

The method by which copies of files are made on magnetic or optical volumes for later recovery or for transfer to another site.

For BACKUP formatted volumes, an ABS save operation creates BACKUP save sets on mag- netic tape volume, a system disk, or optical volume.

save request

ABS policy object that defines the request for saving data.

save set

A file created by the VMS Backup Utility on a volume. When the VMS Backup Utility saves data, it creates a file in BACKUP format called a save set on the specified output volume. A sin- gle BACKUP save set can contain numerous files. Only BACKUP can interpret save sets and restore the data stored in the save set.

slot

A vertical storage space for storing a volume. The storage racks and cabinets used in data centers contain multi-row slots that are labeled to easily locate stored volumes.

storage administrator

One or more privileged users responsible for installing, configuring, and maintaining ABS soft- ware. This user has enhanced ABS authorization rights and privileges and controls the central security domain (CSD) by creating and maintaining ABS storage and environment policies.

storage policy

ABS policy object that defines where to store data saved using ABS.

SYSPRV privilege

The level of privilege required to install the software and add user names to the system.

system backup

An ABS system typically saves the system disk, also known as a full disk backup. The system backup can direct ABS software to perform automotive save operations on a predetermined schedule.

tape

See volume.

transition state

Volumes in the transition state are in the process of being deallocated, but are not yet fully deallocated. The transition state provides a grace period during which a volume can be reallocated to the original owner if necessary.

UASCII

Unlabeled ASCII format. See also ASCII.

UIC

User identification code. The pair of numbers assigned to users, files, pools, global sections, common event flag clusters, and mailboxes. The UIC determines the owner of a file or ABS policy object. UIC-based protection determines the type of access available to the object for its owner, members of the same UIC group, system accounts, and other (world) users.

UID

A globally unique identifier for this instance of an object.

unbind

The act of unbinding a volume or volumes from a magazine.

unlabeled

A recording format that does not include a recorded label.

user backup

A save request created by an individual user (not the system) when they would like to make copies of a file or set of files for later recovery or for transfer to another site.

user profile

The set of information about a user that defines the user's right to access data or the user's right to access an ABS policy object. For ABS on OpenVMS, this includes the following information:

- User name
- UIC
- Privileges
- Access right identifiers

vault

An offsite storage location to where volumes are transferred for safekeeping.

VMS Backup Utility

An OpenVMS Operating System utility that performs save and restore operations on files, directories, and disks using the BACKUP recording format.

volume

A *physical* piece of media (volume) that is known *logically* to the MDMS volume database. A volume can be a single magnetic tape or disk, or as in the case of an optical cartridge, can refer to one side of double-sided media. A volume is assigned a logical name, known as the volume label.

volume ID

The volume's internal identification used to verify that the correct volume has been selected. The volume label should be the same as the volume ID.

volume name

Same as volume ID.

volume set

One or more volumes logically connected in a sequence to form a single volume set. A volume set can contain one or more save sets. ABS adds volumes to a volume set until the storage policy's consolidation criteria has been met or exceeded.

volume state

A volume status flag. In MDMS software, volumes are placed in one of the following states:

- Free
- Allocated
- Transition
- Down

wildcard character

A nonnumeric or nonalphanumeric character such as an asterisk (*) or percent sign (%) that is used in a file specification to indicate "ALL" for a given field or portion of a field. Wildcard characters can replace all or part of the file name, file type, directory name, or version number.

Index

A

ABS
  catalogs 1-2
  client
    OpenVMS 2-1
  policy database 1-2
  policy objects 1-2, 1-4, 10-1
ABS policy
  configuring 3-1
  customizing 3-1
ABS Policy Objects J-2
  Catalog J-2
  Execution Environment J-2
  Restore Request J-2
  Save Request J-2
  Storage Class J-2
Access contol
  environment policy 8-7
Access control
  restore request 10-8
  storage policy 7-7
Archive file system 1-2, 1-7, 7-1
  Files-11 1-2, 1-7
  Files–11 3-5
  MDMS 1-2, 1-7, 3-4
Authorizing
  save request users 9-10

B

Backup
  Oracle database
    Closed N-2
    Open N-5
Backup agent 1-2, 1-7
  restore request
    qualifiers 10-6
  save request
    qualifiers 9-6
Backup management 4-3
  domain 4-3
    centralized 4-4
    combined 4-6
    distributed 4-5
Backup strategies 5-1, 5-4
  disaster recovery A-1
    procedure A-4
  OpenVMS clients 5-5, 5-9
  schedules D-1
  system backup process 5-1
  user backup process 5-2

C

Catalog 1-2, 1-6, 15-1
  creating 15-1
    SLS type 15-3
    staging type 15-4
  improving performance 15-5
  Oracle Rdb database 5-13
  Oracle Rdb storage areas 5-13
  showing 15-5
  storage policy 7-6
Catalog Cleanup Utility C-1, C-2
  log file C-3
  shutting down C-3
  starting C-2
Central security domain 4-1
Clients
  NT 2-2
  OpenVMS 2-1
  UNIX 2-2
Compression
  UNIX clients 8-6
Configuring
  NT client 2-2
  NTsystem backups 5-10
  OpenVMS client backups 5-5, 5-9
  OpenVMS client-server 2-1
  system A-1
  system backups 5-5
  UNIX client 2-2
  UNIX system backups 5-10
Converting ABS to SLS J-1
CRC 8-4
CSD
  see Central security domain 4-1
Cyclic Redundancy Check
  see CRC 8-4

D

Data
- deleting 8-5
- finding 13-1
- retaining 7-5

Data object
- disk name 10-1
- file name 10-1

Data Restore J-4
- Full restore J-5
- Selective file restore J-4

Database
- catalog 1-6

Database Cleanup Utility C-1
- log file C-2
- shutting down C-2
- starting up C-1

Deleting
- evironment policies 12-1
- original data 8-5
- storage policies 12-1

Disaster recovery A-1

Drive 3-5
- enviroment policy 8-6
- storage policy 7-4

E

Environment policy 1-6, 8-1
- access control 8-7
- CRC 8-4
- creating 8-2
- customizing 3-7
- data verification 8-4
- deleting 12-1
- modifying 12-1
- naming 8-2
- number of drives 8-6
- processing commands 8-4
- requirements 8-1
- worksheet E-2

Error messages G-1, H-1

F

Files–11
- archive file system 3-5

Full Operation J-15

G

Graphical user interface
- see GUI 6-1

GUI 6-1
- displaying on an NTclient 6-2
- displaying on an OpenVMS client 6-1
- monitoring job status 14-1
- standard buttons 6-3

I

Incremental Operation J-15

Interfaces 1-3
- CLI 1-10
- GUI 1-10

L

Location 3-5
- restored data 10-7
- storage policy 7-4

Lookup
- by archived dates 13-3
- by file type 13-1
- data 13-1
  - by node 13-2
  - storage policy 13-3

M

MDMS
- archive file system 3-4
- drive 3-5
- location 3-5
- pool 3-5

Media type 3-4
- definitions 1-4
- MDMS 3-4
- storage policy 7-3

N

Node
- storage policy 7-6

NTclient
- displaying the GUI 6-2
- large disk considerations F-2
- saving 9-3
- saving data 9-1
- system backups
  - configuring 5-10

NTsystem backups
configuring policy objects 5-10

O

OpenVMS client
displaying the GUI 6-1
recovering A-6
saving 9-2
saving data 9-1
Operations
Full J-15
Incremental J-15
Selective J-15
System Backups J-13
User Requested J-16
Oracle database backup
Closed N-2
Open N-5
Oracle Rdb
databases
catalog entries 5-13
restoring 5-14
saving 9-1, 9-2
system backups 5-11
storage areas 5-12
catalog entries 5-13
finding catalog entries 5-13
restoring 5-14
saving 9-3
system backups 5-11

P

Policy
ABS 3-1
database 1-2
objects 1-2, 1-4
Policy database 1-2
Policy objects 1-2, 1-4
configuring NTclient system backups 5-10
configuring OpenVMS system backups 5-5
configuring user backups 5-7
environment policy 1-6
restore request 1-4, 10-1
save request 1-4, 3-2, 9-1
storage policy 1-5, 3-4
Pool 3-5
storage policy 7-3
Postprocessing command
environment policy 8-4
restore request 10-5
save request 9-4
Preprocessing command
environment policy 8-4
restore request 10-5
save request 9-4
Processing commands
restore request 10-5
save request 9-4

R

Request
restore 1-4
save 1-4
Requirements
storage management 3-1
Restore request 1-4, 10-1
access control 10-8
deleting 12-1
modifying 12-1
naming 10-1
output location 10-7
restrictions 10-3
scheduling 10-7
selection criteria 10-6
storage policy 10-7
time formats B-1
Restoring
Oracle Rdb databases and storage areas 5-14
Restrictions
restore request 10-3
save request 9-3

S

Save request 1-4
access controls 9-10
agent specific qualifiers 9-6
backup agent qualifiers 9-6
configuring NTsystem backups 5-11
configuring OpenVMS system backups 5-6
configuring UNIX system backups 5-11
configuring user backups 5-9
creating 9-1
data syntax 9-2
deleting 12-1
modifying 12-1
name 9-1
policy object 9-1
postprocessing command 9-4
preprocessing command 9-4
restrictions 9-3

saving file versions 9-6
scheduling 9-6
selection criteria 9-6
simultaneous 7-6
time formats B-1
types 3-2
worksheet E-3
Scheduling
backups D-1
Selection criteria
restore request 10-6
save request 9-6
Selective Operation J-15
Shutdown
Catalog Cleanup Utility C-3
Database Cleanup Utility C-2
SLS to ABS J-1
Startup
Catalog Cleanup Utility C-2
Database Cleanup Utility C-1
Storage management
requirements 3-1
Storage policy 1-5, 3-4
access control 7-7
catalog 7-6
creating 7-2
customizing 3-5
deleting 12-1
drive 7-4
execution node 7-6
location 7-4
lookup data 13-3
media type 7-3
modifying 12-1
naming 7-2
pool 7-3
requirements 7-2
restore request 10-7
retaining data 7-5
running simultaneous save requests 7-6
tape options 7-3
worksheet E-1
Symbolic links
environment policy 8-7
System Backups J-13
System backups 5-1
ABS clients F-4
configuring 5-5
NTsave requests 5-11
OpenVMS save requests 5-6
UNIX save requests 5-11
configuring policy objects 5-5
NTclient F-2
Oracle Rdb
databases 5-11
storage areas 5-11
UNIX client F-2

U

UNIX client
compression options 8-6
environment policy 8-6
large disk considerations F-2
saving 9-3
saving data 9-1
symbolic links 8-7
system backups
configuring 5-10
User backup 4-2
restrictions 4-2
User backups 5-2
configuring 5-7
save requests 5-9
configuring policy objects 5-7
process context 5-2
user profile 8-5
user profile process 5-2
User profile
environment policy 8-5
User Requested Operation J-16
Utilities
Catalog Cleanup Utility C-1
Database Cleanup Utility C-1

V

Volume 7-4
usage F-4
Volume sets 7-4
creating 7-4