Vault System Administrator Guide

Optegra® Release 6

DOC35032-010

Parametric Technology Corporation

Parametric Technology Corporation, 140 Kendrick Street, Needham, MA 02494-2714
8 January 2001

Table of Contents

Preface

Introduction to System Administration

Working with Storage Pools

Changing Storage Pool Selection

Performing a Backup of Files

Using Exabyte and DAT Tapes

Setting Up a Rulebase

System Administration Tasks For UNIX/NT

Summary of Network Services Commands

Editing the Vault Configuration Files

Vault Domain Management

Problem Solving

Troubleshooting Common Problems

Work Example on Customizing Storage Pool Selection

Command Abbreviations

Preface

Vault System Administrator Guide contains information for performing system administration tasks related to Vault on the Vault host system. It also provides information that is specific to UNIX systems and gives a summary of Network Services commands used to administer and control vault server processes. This guide also gives an overview of the vault processes, explaining their function and interdependence.

Related Documents

The following documents may be helpful as you use *Vault System Administrator Guide*:

- *Vault Command Reference*
- *Vault Programmer Guide*
- *Vault Manager Guide*
- *Vault Interactive Query Facility Guide*

Book Conventions

The following table illustrates and explains conventions used in writing about Optegra applications.

Convention	Example	Explanation
EPD_HOME	`cd $EPD_HOME/install` (UNIX) `cd %EPD_HOME%\install` (Windows)	Represents the default path where the current version of the product is installed.
Menu selections	Vault > Check Out > Lock	Indicates a command that you can choose from a menu.
Command buttons and options	`Mandatory` check box, `Add` button, `Description` text box	Names selectable items from dialog boxes: options, buttons, toggles, text boxes, and switches.

User input and code	`Wheel_Assy_details`
	
	`-xvf /dev/rst0`
	
	`Enter command>` **`plot_config`**
		Enter the text in a text box or on a command line.
		
		Where system output and user input are mixed, user input is in bold.

System output	`CT_struct.aename`	Indicates system responses.
Parameter and variable names	`tar -cvf /dev/rst0` filename	Supply an appropriate substitute for each parameter or variable; for example, replace filename with an actual file name.
Commands and keywords	The `ciaddobj` command creates an instance of a binder.	Shows command syntax.
Text string	`"SRFGROUPA"` or `'SRFGROUPA'`	Shows text strings. Enclose text strings with single or double quotation marks.
Integer	n	Supply an integer for *n*.
Real number	x	Supply a real number for *x*.
#	`# mkdir /cdrom`	Indicates the root (superuser) prompt on command lines.
%	% `rlogin` remote_system_name `-l root`	Indicates the C shell prompt on command lines.
$	$ `rlogin` remote_system_name `-l root`	Indicates the Bourne shell prompt on command lines.
>	`> copy` filename	Indicates the MS-DOS prompt on command lines.
Keystrokes	`Return` or `Control-g`	Indicates the keys to press on a keyboard.

Online User Documentation

Online documentation for each Optegra book is provided in HTML if the documentation CD-ROM is installed. You can view the online documentation from an HTML browser or from the HELP command.

You can also view the online documentation directly from the CD-ROM without installing it.

From an HTML Browser:

1. Navigate to the directory where the documents are installed. For example,

   $EPD_HOME/data/html/htmldoc/ (UNIX)

   %EPD_HOME%\data\html\htmldoc\ (Windows NT)
2. Click mainmenu.html. A list of available Optegra documentation appears.
3. Click the book title you want to view.

From the HELP Command:

To view the online documentation for your specific application, click HELP. (Consult the documentation specific to your application for more information.)

From the Documentation CD-ROM:

1. Mount the documentation CD-ROM.
2. Point your browser to:

   CDROM_mount_point/htmldoc/mainmenu.html (UNIX)

   CDROM_Drive:\htmldoc\mainmenu.html (Windows NT)

Printing Documentation

A PDF (Portable Document Format) file is included on the CD-ROM for each online book. See the first page of each online book for the document number referenced in the PDF file name. Check with your system administrator if you need more information.

You must have Acrobat Reader installed to view and print PDF files.

The default documentation directories are:

- $EPD_HOME/data/html/pdf/doc_number.pdf (UNIX)
- %EPD_HOME%\data\html\pdf\doc_number.pdf (Windows NT)

Resources and Services

For resources and services to help you with PTC (Parametric Technology Corporation) software products, see the *PTC Customer Service Guide*. It includes instructions for using the World Wide Web or fax transmissions for customer support.

Documentation Comments

PTC welcomes your suggestions and comments. You can send feedback in the following ways:

- Send comments electronically to `doc-webhelp@ptc.com`.
- Fill out and mail the PTC Documentation Survey located in the *PTC Customer Service Guide.*

Chapter 1

Introduction to System Administration

This chapter introduces Optegra documentation for the System Administrator and lists platforms on which Optegra runs.

- Overview of System Documentation
- Other Documentation
- Client and Server Machines
- Version Numbers in Configurations
- Setting Up Defaults
- OBJECT-CLASS Parameter
- Keywords and the Corresponding Commands

Overview of System Documentation

*Vault System Administrator Guid*e is the basic guide for the Vault System Administrator. It contains both portable and platform-specific system management information. The portable information covers storage pools, local file rulebases, and some aspects of performing backups. The platform-specific information is about executing tape commands, running backups, and recovering data.

Basic Books for the System Administrator

These books contain the primary information needed to perform system administration for Vault.

1. *Optegra Release Notes*

   The *Optegra Release Notes* is the release bulletin that accompanies each software shipment of Optegra software. It provides information that allows you to do the following at your site:

   - Determine if you have the hardware, software, and documentation for your installation
   - Become aware of any special operating considerations
   - Determine problem reporting and resolution procedures

2. *Installing Vault and Locator*

   This book explains software installation procedures for company representatives and on-site personnel responsible for initial software installation on a UNIX-based machine. It contains installation instructions for Vault, Distributed Vault, and Programming.

Other Documentation

As a system administrator, you need to know the platform that hosts Vault and the interface between Vault and the host system.

Vault Managers

As the Vault manager, you need to know Vault thoroughly to implement it in the best way possible for the site. Consequently, you need books on the following topics:

- Planning
- Security and Setup
- Interactive Query Facility
- Database Tables

You also need to use the reference books listed on the previous page. There are usually around one or two Vault managers at each site.

Client and Server Machines

Vault is based on the client/server model. This is an application architecture that allows the separation of functions. In Vault, the servers manage storage pools and write to an RDBMS file and the clients present the information to the users. Server machines provide data management services, while client machines provide access to database information.

Servers

The machine on which the Vault software and Vault database resides is called the Vault server. Servers provide these data management services:

- Administrative functions
- File transfer facilities
- Multiple levels of security
- Electronic voting capability
- Reporting and querying functions

You can have any number of Vault servers. You can also have more than one Vault server on a physical node.

Clients

The machines on which the Vault Client software resides are known as Vault clients. Client systems can be local or remote to the server. Vault Client software allows you to access data under the control of Vault.

Any system running Vault Client software can be networked to access a Vault of the same or greater Vault version. However, the usage of the Vault commands and features are limited according to the platform used.

Version Numbers in Configurations

The version number of a Vault release is the first number before the period/dot. For example, the “3” in Vault 3.0 is a version number. When a version number increases, it means that a Vault release has gained new commands or other major features.

Point numbers indicate that the same set of features is now available on a new platform or that enhancements have been added to existing commands/features. For example, the .0.1 in Vault 3.0.1 are the point numbers.

Setting Up Defaults

The Vault.DEFAULTS file can specify defaults to be used for various Vault functions. You can find or create the EDM.DEFAULTS file in the /$EPD_HOME/data directory. You can also have an individual copy in your current working and/or home directory.

Please note: The /$EPD_HOME/data/EDM.DEFAULTS file contains all the applications, even though these are not installed or available on your system. This file lists all applications that are supported by EPD.Connect.

Alternative to the EDM.DEFAULTS File

To set a default in the EDM.DEFAULTS file, prefix the EDM variable with EDM and execute the OS command as you would. EDM treats any variable defined this way as though it is defined in the EDM.DEFAULTS file. For example, on a UNIX system you can enter

```
% setenv EDM_EDMTEMPDIR /user/snoopy/tempfiles
```

Creating the EDM.DEFAULTS File

Vault creates the EDM.DEFAULTS file when you install a new version of Vault by running edminstall. The edmrefresh script can also recreate the EDM.DEFAULT file for the new rulebase format.

Location of Rulebase Executables

The CADDS variable specifies the path name of the CADDS rulebase. The LOCAL variable specifies the path name of the LOCAL rulebase. When you install Vault, the system creates the EDM.DEFAULTS file and includes entries for CADDS and LOCAL.

Please note: Do not change these entries.

If you define a new rulebase for use at your site, you must specify its name in the EDM.DEFAULTS file. The name of the rulebase is the value you enter for the environment parameter when executing the STORE command. Set the name of the rulebase equal to the path name of the rulebase executable.

The ENV variable is used directly by the STORE command. It is used indirectly in conjunction with the OBJECT-CLASS parameter by the following commands: GET, READ, REPLACE, UPDATE.

The following syntax in the EDM.DEFAULTS file sets the ENV parameter for each rulebase:

```
ENV(rulebase_name)=fully_qualified_name_of_executable
```

This syntax is the pattern for the following entries:

```
ENV(LOCAL)=$EPD_HOME/bin/caddsif
ENV(CADDS)=$EPD_HOME/bin/ddcdsrb
```

Please note: Any user-written rulebase must have an ENV entry similar to the preceding ones.

The SELSCOPE parameter is used by the STORE command. The following syntax in the EDM.DEFAULTS file sets the allowed selection scopes for the STORE parameter for each rulebase:

```
SELSCOPE(rulebase_name)=selscope_values
```

This syntax is the pattern for the following entries:

```
SELSCOPE (CADDS)=F,P
SELSCOPE (LOCAL)=F,D
```

Please note: Any user-written rulebase must have a SELSCOPE entry similar to the preceding ones.

Case Sensitivity

The EDMCASE variable determines whether Vault converts lowercase selection name input to uppercase or accepts lowercase selection name input as it is. Set EDMCASE equal to MIXED to recognize and preserve lowercase input for selecting the names.

EDMCASE has precedence over any case rules in the rulebase associated with the file or part, except for CADDS. CADDS files and parts are converted to uppercase when stored. EDMCASE affects execution of the GET, READ, REPLACE, and UPDATE commands. You must specify uppercase when performing the GET, READ, UPDATE, or REPLACE command on a CADDS file or part.

Location of Command Audit Files, Menu Profile, and Other Files

The EDMTEMPDIR variable specifies a path name for the directory in which you want Vault to place your command audit files, Vault menu profile, and other files produced by Vault.

When you do not specify a value for EDMTEMPDIR in the EDM.DEFAULTS file, Vault places command audit files, your Vault menu profile, and any other files produced in your current working directory.

For example, if EDMTEMPDIR is set to /users/owright, the path name of the audit file resulting from the STORE command is:

```
/users/owright/STORE.EDMAUDIT
```

Tape Device Names

Use the EDM.DEFAULTS file to specify device names for the Vault logical tape units TAPE1, TAPE2, TAPE3, and TAPE4. For example, on Solaris, you might include the entry

```
TAPE1=/dev/rmt/5
```

Printers

Use the EDM.DEFAULTS file to specify the printer used by the Interactive Query Facility (IQF). The PRINTER statement has the following format.

```
PRINTER=command_for_printer
```

The command for printer value is an operating system-specific command, identical to a command you would enter at your OS prompt. For example, on a Solaris machine you can direct IQF output to a printer called lpeddie with the following statement in the EDM.DEFAULTS file.

```
PRINTER=lp -d lpeddie
```

There can be only one PRINTER statement in an EDM.DEFAULTS file. If it does not exist, IQF sends printed output to the system's default printer.

Settings for Export Command

Specify one or more of the following values in the $EPD_HOME/data/EDM.DEFAULTS file, to export files and parts belonging to the other rulebases,

- EXPORT-PREFIX
- EXPORT-SUFFIX
- EXPORT-FILEDIR

To set the parameters for each rulebase, use the following syntax in the EDM.DEFAULTS file:

- EXPORT-PREFIX(rulebase_name)=suffix_values
- EXPORT-SUFFIX(rulebase_name)=prefix_values (if any)
- EXPORT-FILEDIR(rulebase_name)=OS level representation of the files and parts (if any)

  The values are:

  - F — If the object is stored as a file.
  - D — If the object is stored as a directory.

Please note: You need not do these settings in the EDM.DEFAULTS for the files and parts with representations similar to CADDS and LOCAL.

- If both SUFFIX and PREFIX exists for an object, specify both the values in the EDM.DEFAULTS file.
- If SELSCOPE=F and the OS level representation of the object is file, then do not specify the EXPORT-FILEDIR value in the EDM.DEFAULTS file.
- If SELSCOPE=P and the OS level representation of the object is directory, then do not specify the EXPORT-FILEDIR value in the EDM.DEFAULTS file.

Exporting PS File

To export the PS file, set the following in the EDM.DEFAULTS file:

On UNIX:

```
EXPORT-SUFFIX(PS)=/_ps
EXPORT-FILEDIR(PS)=D
```

ON WINDOWS NT:

```
EXPORT-SUFFIX(PS)=\_ps
EXPORT-FILEDIR(PS)=D
```

When you store a PS file in Vault, it requires Ifname as directory name, followed by suffix value (_ps) for EXPORT-SUFFIX(rulebase_name) variable.

When you store a PS file in Vault, it requires Ifname as directory name, followed by suffix value (_ps for Unix and _ps for Windows NT) for EXPORT-SUFFIX(rulebase_name) variable.

lfname is the localFileName which is name of the local file or directory in your local area. The filename length can be up to 80 characters.

Please note: Do not set the EXPORT_PREFIX variable in the EDM.DEFAULTS file, if the prefix does not exist.

Exporting MEDSHT File

To export the MEDSHT file, set the following in the EDM.DEFAULTS file:

On UNIX:

```
EXPORT-PREFIX(MEDSHT)=s.
EXPORT-FILEDIR(MEDSHT)=F
```

ON WINDOWS NT:

```
EXPORT-SUFFIX(MEDSHT)=s.
EXPORT-FILEDIR(MEDSHT)=D
```

MEDSHT part is represented at the OS level as a file starting with s.partname.

When a MEDSHT file is stored in Vault, it requires Ifname as part name, with .s as the prefix value for EXPORT-PREFIX(rulebase_name) variable. In this case SUFFIX does not exist.

Please note: Do not set the EXPORT-SUFFIX variable in the EDM.DEFAULTS file, if the suffix does not exist.

OBJECT-CLASS Parameter

The OBJECT-CLASS parameter maps the part-type in the database to the rulebase with which the client software processes the part-type. The two part types that must be mapped are CADDS and LOCAL. The following lines show the mappings:

```
OBJECT-CLASS(CADDS)=CADDS
OBJECT-CLASS(LOCAL)=LOCAL
```

Example: The installed EDM.DEFAULTS file has the following appearance.

```
VAULTID=local
ENV(LOCAL)=$EPD_HOME/bin/ddlclrb
ENV(CADDS)=$EPD_HOME/bin/caddsif
ENV(ADRAW)=$EPD_HOME/bin/caddsif
SELSCOPE(CADDS)=F,P
SELSCOPE(LOCAL)=F,D
SELSCOPE(ADRAW)=P
OBJECT-CLASS(CADDS)=CADDS
OBJECT-CLASS(LOCAL)=LOCAL
OBJECT-CLASS(ADRAW)=CADDS
SELSCOPE=F
ENV=LOCAL
EDMCASE=MIXED
```

Please note: The cont keyword is not recognized as valid in the EDM.DEFAULTS file.

Environment Variables

Any default that you set in the EDM.DEFAULTS file, can also be set with the operating system command that sets environment variables. To do so, add the prefix EDM to the Vault variable and execute the OS command. Vault treats any variables defined this way as though they were defined in the EDM.DEFAULTS file. For example, on a UNIX system you can enter:

```
setenv EDM_EDMTEMPDIR /user/snoopy/tempfiles.
```

Example: The EDM.DEFAULTS file might look like the one below if a site-defined rulebase named JET existed.

```
ENV(CADDS)=$EPD_HOME/bin/caddsif
ENV(LOCAL)=$EPD_HOME/bin/ddlclrb
EDMCASE=MIXED
EDMTEMPDIR=/users/snoopy/tempfiles
ENV(JET)=$EPD_HOME/bin/ddjetrb
TAPE2=/dev/rmt/1
```

Multiple Copies of EDM.DEFAULTS File

Multiple copies of the EDM.DEFAULTS file can exist to provide two levels of variables:

- Site-wide variables
- Individualized user variables.

When a Vault command is issued, Vault first checks the EDM.DEFAULTS file in the data directory. It then checks the current working directory. If there is no defaults file, Vault checks the home directory. For variables that are common to the EDM.DEFAULTS file in the data directory and the home directory, Vault uses the values defined in your directory.

Use the EDM.DEFAULTS file in the data directory to specify values for variables that you want your entire site to use. For example, you must specify values for the LOCAL and CADDS rulebases in the data directory version of the EDM.DEFAULTS file. Also, define tape logicals in the EDM.DEFAULTS file that is in the data directory.

In your current working directory, use the EDM.DEFAULTS file to specify values for such things as selection scope, append option, and environment.

Command Parameter Values

If the system defaults are not useful for you, you can set a default for some command parameters by including a line in the EDM.DEFAULTS file in the following format.

```
keyword=value
```

For example, CLASS=PRO makes sense if you always work on project files. The ENV parameter requires a value, set either explicitly on the command line or in the EDM.DEFAULTS file. Any rulebase can be made the default with this syntax:

```
ENV=envvalue
```

The SELSCOPE parameter requires a value set, either explicitly on the command line or in the EDM.DEFAULTS file. In releases prior to 1.1 the value of SELSCOPE was F, and you can retain that default value by including the following line in the EDM.DEFAULTS file:

```
SELSCOPE=F
```

Any rulebase can be made the default with this syntax:

```
SELSCOPE=selscope
```

The VAULTID parameter specifies which commands refers to the vault. Its value should remain LOCAL unless Distributed Vault is installed.

The following table lists the parameters for which you can set defaults:

Table 1-1 List of Commands and their Parameters

Command	Parameters
ADDT	CLASS
ADDTU	ADMIN READA WRITEA PROTGREP
ADDUSA	REJCNT TERMCNT
ADDMUL	ULMTYPE
ADDMFS	MEMTYPE
ARCHIVE	APPEND OUTPUT TAPEUNIT TAPENUM TAPPEND
ADDUP	ADMIN READA WRITEA
CHGFA	APPEND OUTPUT SELSCOPE
CHGFCL	APPEND OUTPUT SELSCOPE
CHGFPW	APPEND OUTPUT SELSCOPE
CHGFREV	SELSCOPE
CHGFSC	APPEND OUTPUT SELSCOPE
COPY	APPEND OUTPUT SELSCOPE
DELETE	APPEND OUTPUT
GET	APPEND LDIRNAME OUTPUT SELSCOPE DATECRIT
IBKUP	TAPEUNIT TAPENUM TAPPEND
LOAD	APPEND CLASS DATECRIT LOADTYPE OUTPUT SELSCOPE TAPEUNIT
LISTDIR	APPEND DIRFORM DIRNAME OUTPUT SELSCOPE
MARKA	APPEND OUTPUT SELSCOPE
MARKD	APPEND OUTPUT SELSCOPE
MARKR	APPEND OUTPUT SELSCOPE
PURGE	APPEND OUTPUT SELSCOPE
READ	APPEND DATECRIT LDIRNAME OUTPUT SELSCOPE
READMSG	APPEND
RECSF	TAPEUNIT
RECSP	TAPEUNIT APPEND

Table 1-1 List of Commands and their Parameters

Command	Parameters
REPLACE	APPEND OUTPUT SELSCOPE
RESTORE	APPEND OUTPUT TAPEUNIT
REMMUL	ULMTYPE
REMMFS	MEMTYPE
RESET	APPEND OUTPUT SELSCOPE
RESERVE	CLASS SIGNOUT
RSVP	SELSCOPE
REQRVW	APPEND OUTPUT SELSCOPE
SCANTAPE	APPEND DATECRIT OUTPUT SELSCOPE TAPEUNIT
SIGNON	DMSERVER
SIGNOUT	APPEND DATECRIT OUTPUT SELSCOPE
STORE	APPEND CLASS ENV OUTPUT SELSCOPE
UBKUP	CLASS CYCLES TAPEUNIT
UNMARK	APPEND OUTPUT SELSCOPE
UNLOAD	APPEND DATECRIT OUTPUT SELSCOPE TAPEUNIT UNLDTYPE
UPDATE	APPEND OUTPUT SELSCOPE

Keywords and the Corresponding Commands

Table 1-2 Keywords and the corresponding commands

Keyword	Commands
ADMIN	ADDU, ADDUP
APPEND	ARCHIVE, CHGFA, CHGFCL, CHGFPW, CHGFSC, COPY, DELETE, GET, LISTDIR, LOAD, MARKA, MARKD, MARKR, PURGE, READ, READMSG, RECSP, REPLACE, REQRVW, RESET, RESTORE, SCANTAPE, SIGNOUT, STORE, UNLOAD, UNMARK, UPDATE
CLASS	LOAD, RESERVE, STORE
CLEAR	UBKUP
CYCLES	UBKUP
DATECRIT	GET, LOAD, READ, SCANTAPE, SIGNOUT, UNLOAD
DIRFORM	LISTDIR
DIRNAME	LISTDIR
ENV	STORE
LDIRNAME	GET, READ
LOADTYPE	LOAD
MEMTYPE	ADDMFS, REMMFS
OUTPUT	ARCHIVE, CHGFA, CHGCL, CHGFPW, CHGFSC, COPY, DELETE, GET, LISTDIR, LOAD, MARKA, MARKD, MARKR, PURGE, READ, REPLACE, REQRVW, RESET, RESTORE, SCANTAPE, SIGNOUT, STORE, UNLOAD, UNMARK, UPDATE
PROTGRP	ADDU
READA	ADDU, ADDUP
RESENT	ADDUSA
SELSCOPE	CHGFA, CHGFCL, CHGFPW, CHGFREV, CHGFSC,COPY, GET, LISTDIR, LOAD, MARKA, MARKD, MARKR, PURGE, READ, REPLACE, REQRVW, RESET, RSVP, SCANTAPE, SIGNOUT, STORE, UNLOAD, UNMARK, UPDATE
SIGNOUT	RESERVE
TAPEUNIT	ARCHIVE, IBKUP, LOAD, RECSP, RESTORE, SCANTAPE, UBKUP, UNLOAD
TERMCNT	ADDUSA
UNLDTYPE	UNLOAD
ULMTYPE	ADDMUL, REMMUL
WRITEA	ADDU, ADDUP
TAPENUM	ARCHIVE, IBKUP
WRITEA	ARCHIVE, IBKUP
DMSERVER	SIGNON

Chapter 2

Working with Storage Pools

This chapter provides information and instructions for working with Vault storage pools. Storage pools hold the data you store in Vault.

- Introduction
- Adding a Storage Pool
- Changing the Status of a Storage Pool
- Changing the Type of a Storage Pool
- Recovering a Storage Pool
- Using Local Copies to Rebuild Latest Versions
- Selecting a Storage Pool for a File
- Changing Storage Pool Selection

Introduction

Storage pools are mounted file systems in which Vault stores files. The storage pool is selected by the Vault. The process of selecting the storage pool is called storage pool selection. Vault makes this selection according to SQL logic which you can customize. Storage pools can also be NFS-mounted file systems.

When you install Vault, assign a minimum of two storage pools but preferably seven or more storage pools. After installation you can add more storage pools. The Vault determines the storage pool to place the file. This occurs when you execute the `STORE, LOAD, UPDATE, REPLACE, COPY, RECSF`, or `RESTORE` commands.

Selecting Storage Pool

Storage pool selection is based on SQL queries executed at run time. Vault selects a storage pool by examining the following:

- Characteristics of the file you are storing
- Current status of the set of storage pools

Customizing Storage Pool Selection

The sequence of SQL queries that select a storage pool is stored in the Vault database. You can customize storage pool selection by changing these queries.

Vault includes a default storage pool selection algorithm. You have the option of changing this default. You are not required to modify the selection logic if the default storage pool selection meets your needs. There are two ways of customizing storage pool selection.

- Use one of the storage pool selection templates provided with Vault.
- Design your own storage pool selection.

Use Vault to assign types to storage pools when you customize storage pool selection.

Customizing Storage Pool Selection

The files stored on Vault have varying characteristics. For example, many files can be classified under a project or private owner. Many have an in-work status while others are released. Files vary in size. Customizing storage pool selection allows you to group files by these and other characteristics.

The installation includes different types of disk media such as optical disk, slow and fast drives. Customize storage pool selection to assign files to various media.

Some storage pools can be taken offline automatically by Vault or by the system administrator, while Vault continues to operate using the remaining online storage pools. By grouping files according to specific characteristics, it is easy to control operations when storage pools need to be taken offline.

Using storage Pool Commands

After installation you can:

- Assign another storage pool to Vault. Use the Vault command `ADDSP` (Add Storage Pool)
- Change the status of a storage pool, for example, from online to offline. Use the Vault command `CHGSPS` (Change Storage Pool Status)
- Change the type of a storage pool. This is useful if you plan to customize how Vault selects a storage pool in which to place a file. Use the Vault command `CHGSPT` (Change Storage Pool Type)
- Recover a destroyed or defective storage pool. Use the Vault command `RECSP` (Recover Storage Pool)

Instructions for using the `ADDSP, CHGSPS, CHGSPT, and RECSP` commands are explained later in this chapter. Sign on to Vault before you can execute any of these commands.

Determining Storage Pool Needs

The number of storage pools you need, and the space allotted to each storage pool varies from site-to-site. When determining storage pool needs, it is important to understand that Vault stores each version of a file as it is updated. For example, suppose the `wing` file uses 2MB. Sign the file and modify it. During the day, you update the file in Vault three times and then you sign it back in to Vault at the end of the day. Vault now stores five copies of the `wing` file - original, three update and the signed in version. This means that `wing` file needs 10MB.

Although Vault includes five copies of the `wing` file, you can only access the latest version. The other versions are not available to you, but Vault maintains them until you perform an incremental backup followed by deleting the older version. Vault maintains old versions so that they are available in case the latest version is corrupted.

Consequently, you should consider the following when determining the storage pool capacity:

- Average size of files
- Number of files
- Frequency of file transfers, that is how often files are updated and replaced
- Length of work day
- Number of users
- Frequency of backup and delete old version procedures
- Frequency of archive procedures

Additional Information about Storage Pools

Information pertaining to storage pools touches many phases of Vault use.

- Before storage pools can be used by Vault, you need to analyze disk space requirements. If required, purchase the equipment.
- *Installing Vault and Locator* provides detailed information about partitioning disks into Vault storage pools at installation time.
- Chapter 3, “Changing Storage Pool Selection” discusses how to select and install one of the alternate selection templates. It also outlines the steps involved in designing and installing custom logic.
- Appendix A, “Work Example on Customizing Storage Pool Selection” provides detailed examples on customizing your storage pool selection.

Adding a Storage Pool

Use the ADDSP command to add a storage pool to the Vault database. You can execute this command only when you are locally logged in to the Vault server. The ADDSP command requires information specific to your operating system.

On UNIX systems, a storage pool is a mounted file system. When you add a storage pool, the status is available online. Vault does not assign a type to the storage pool when you add it. You must execute the CHGSPT (change storage pool type) to assign a type to the new storage pool.

ADDSP Command Parameters

Table 2-1 Parameters for ADDSP command

Keyword=Prompt	Parameter Description
poolname=Storage pool name	Unique storage area name in Vault. Enter 8 or fewer letters and/or numbers.
poolinfo=Storage pool information	Operating system-specific information needed to access the physical storage area, 240 or fewer alphanumeric characters (including blank spaces). See *Installing Vault and Locator.*

Using the Command-Line Format

In this example, a storage pool named POOL2 is added to Vault.

```
ciaddsp
EDM> POOLNAME=POOL2 POOLINFO=/EDMPOOLS/POOL2
```

Changing the Status of a Storage Pool

Use the CHGSPS command to change the status of a storage pool.

CHGSPS Command Parameters

Table 2-2 Parameters for CHGSPS command

Keyword=Prompt	Parameter Description
poolname=Storage pool name	Unique name of an existing storage area in Vault. Enter 8 or fewer letters and/or numbers.
poolstat=Storage pool status	A one-character status value. The following values are allowed: 0 Available and online 1 Locked in write-mode by Vault 2 Read-only 3 Offline 4 Unavailable as detected by Vault 5 Unavailable due to media failure 6 Tagged for roll forward by the RECSP command 7 Recovered by RECSP 8-9 Reserved for future use A-Z Unavailable with user-defined reason Values 0-9 are defined by Vault and stored in a two-column database table. These columns are for the actual status value and a description. Values A-Z are user-defined values that are loaded into the database with a file. Treatment of values A-Z is functionally equivalent to the status value of 3 (offline).

The Command-Line Format

In this example, the status of a storage pool named EDMA202 is changed to 3.

```
cichgsps
EDM> POOLNAME=EDMA202 POOLSTAT=3
```

Changing the Type of a Storage Pool

Use the CHGSPT command to add or remove a storage pool type.

CHGSPT Command Parameters

Table 2-3 Parameters for the CHGSPT command

Keyword=Prompt	Parameter Description
poolname=Storage pool name	Unique name of an existing storage area in Vault. Enter 8 or fewer letters and/or numbers.
pooltype=Storage pool type	A 32-character string without embedded blanks that contains the categorization value.
chgtype=Action	A one-character string that indicates the action to be performed. Enter A for Add or R for Remove.

Using the Command-Line Format

In this example, the type of a storage pool named EDMA202 is changed.

```
cichgspt
EDM> POOLNAME=EDMA202 POOLTYPE=cadds CHGTYPE=a
```

Recovering a Storage Pool

Use the RECSP command to recover the contents of a destroyed or defective storage pool. You can execute this command only when you are locally logged in to the server.

Before you execute the RECSP command, issue the CHGSPS command to change the status of the storage pool to 6. After you execute RECSP, the recovered storage pool has a status of 7.

To execute the RECSP command, it must be in your PUBLIC command list. Locator on a system that is remote from the Vault database does not include the RECSP command. If the files to be recovered are associated with user-defined attributes, the attribute server must be running during execution of the RECSP command. If the attribute server is not running, the association between the files and/or parts and the attributes is lost during execution of the RECSP command.

Files That Cannot Be Recovered

Any files stored within a corrupt or destroyed storage pool area that have not been previously backed up cannot be recovered. Normally, these can be files stored in Vault after the last incremental or universal backup was performed.

Regenerating the Storage Pool

Before you execute the RECSP command, regenerate the storage pool with the files from the last universal backup tape created with the UBKUP command.

During the recovery process, you might be asked to mount a specific incremental backup tape that contains the most current file versions.

You cannot change media (tape types) during execution of the RECSP command. Consequently, when performing an incremental backup, always use the same type of tape media as used by any other incremental backup since the last universal backup. This is important when you want to recover a storage pool. The RECSP command might need to read data from several tapes during its operation.

When Vault recovers a storage pool, the recovered data includes any user-defined attributes that were associated with the files.

Audit Information

The audit information generated by this command provides information for each file showing whether it was:

- recovered with the current version
- recovered with a previous version
- not recovered

To recover more than one storage pool at a time, set the Replace/append audit file parameter to A. Otherwise, the next storage pool recovery audit information overwrites the previous information. The default value is A.

Audit information created by the `RECSP` command is written in the `RECSP.EDMAUDIT` file in your current working directory.

Table 2-4 Parameters for the RECSP Command

Keyword=Prompt	Parameter Description
`tapeunit`=Tape unit	Name to represent the tape device. Enter: **TAPE1** or **TAPE2** or **TAPE3** or **TAPE4**. the default is TAPE1
`poolname`=Storage pool name	Existing storage area name in Vault. Enter: 8 or fewer letters and/or numbers.
`append`=Replace or append audit file	Indicates whether to replace or append to (or create) the audit file. Enter: **A** (append) or **R** (replace). Default is A

Using the Command-Line Format

In this example, the EDMA202 storage pool contents from TAPE2 are recovered. The audit information generated by this command is appended to the previously saved audit information (the default).

```
CIRECSP
EDM> TAPEUNIT=TAPE2 POOLNAME=EDMA202
```

Categories of Files after Executing RECSP

After you execute the RECSP command, all files residing in the storage pool fall into one of the following categories.

- Restored to the latest version (an exact copy of what was lost) from the latest universal backup tape. The date and time stamp for these files precedes the latest universal backup.
- Restored to latest version from an incremental backup tape. The date and time stamp for these files precedes one of the incremental backups taken since the last universal backup.
- Restored to an old version from either an incremental backup tape or another storage pool. The date and time stamp for the latest version of these files is later than the latest incremental backup, but a previous version is available. Vault's default storage pool selection logic assigns a new version of a file to a different storage pool than the one most recently used so that a recent version of a file is often available when a storage pool is lost.
- Lost permanently. Files stored for the first time since the latest universal or incremental backup cannot be recovered.

Latest Version Recovered

Files recovered to their latest version retain the Vault state they had before the storage pool was lost. Files signed out or marked for archive or delete, remain signed out or marked. Files previously in review remain in review as long as the selection in review to which they belong has not been partially destroyed. Recovered files that belonged to a part continue to belong to that part.

Old Version Recovered

Files that were recovered to an old version usually drop any special Vault state they might have had. Files marked for archive or delete are no longer marked. Files previously in review are no longer in review. The files that were signed out remain signed out. Files recovered from old versions assume the attributes of the old version (class, owner, and status code).

Lost Files

Files that were lost altogether also lose any special Vault status they might have had such as marked for archive, marked for delete, or in review, but they remain in the Vault file directory as placeholders. If the file they replace was signed out, the placeholder representing them is signed out. The placeholder for a lost file no longer belongs to any part.

Using Local Copies to Rebuild Latest Versions

If you have local copies of the files that were lost or recovered to old versions, you can rebuild the latest version in Vault using the `EDM REPLACE` or `STORE` commands.

- Lost file now shown as a placeholder in Vault.
  - Standalone file: Store the file again.
  - Part member file: Store the file again and specify that the file is to be added to the particular part.
- File recovered to an old version, file is signed out.
  - Standalone file: Replace or update the file.
  - Part member file: Replace or update the part.
- File recovered to an old version, file not signed out.
  - Get and replace file: This is convenient for in-work files, but be careful not to overlay the local, good copy with the `GET` command. Or, you can read, purge, and then store the file. This is best when the file is released.
  - Part member file: Get and replace part or read, purge, and then store file using the option for adding the file to the part.

If you no longer want the old versions or placeholders, you can remove them with the `PURGE` or `DELETE` commands or, for placeholders, the `RESET` command.

Selecting a Storage Pool for a File

The overview of how storage pool selection operates can help you to understand the steps necessary to customize storage pool selection.

To select the storage pools, Vault:

- examines the file and the current database state for the existence of various conditions
- selects the most important criteria for storing the file
- finds all the storage pools satisfying that criteria
- selects one in which to store the file

The SQL code used in performing these steps is stored in Vault database tables. Customizing the selection process involves changing this code.

Examining the File and Database: Selection Queries

Vault examines the database record containing all characteristics of the file such as its unique name, project, status, size, and storage pool where the last version of the file was stored.

The selection process uses SQL queries to test for the existence of various conditions in the database. Any Vault-controlled table can be queried. Each query results in the determination of true or false, where true means that the condition tested exists in the database.

Since a single selection query can have either of two outcomes, each query allows a branch in the selection logic. The next step on each branch can be another selection query or the pool filter. The pool filter identifies the storage pools that meet the criteria.

The results of the queries determine the most important criteria for storing the file. The sequence of selection query steps is critical for identifying the criteria and thus choosing the proper pool filter. More than one of the tested database conditions or file states can be true. If the first true condition selects the pool filter, the order of the tests can change the outcome.

Finding Storage Pools: The Pool Filter

The storage pools chosen are associated with the SQL code that finds all applicable storage pools and ranks them according to characteristics of the storage pools. Vault selects the highest-ranked storage pool in the list based on the following tests:

- The storage pool must be online.
- It must have enough space left to hold the file.

Controlling the Selection Process

The SQL code used in the selection process is stored in the Vault database tables. Customizing the selection process includes changing the data stored in these tables.

- One table contains the selection queries. Each entry has a unique name for its SQL code.
- Another table stores the pool filter. Each entry has a unique name for its SQL code.
- A third table contains control logic that:
  - identifies by name the selection query to execute in each step
  - identifies the next selection query or decision criteria to use for each of the two possible query outcomes
- Two additional tables store descriptions of the SQL code. These tables are not used during processing but can be of used in understanding the function of each SQL query without reading the SQL code itself.

Role of Storage Pool Types

The default storage pool selection logic does not require storage pool types. But most alternate pool filters do select storage pools by their type. For example, if you base storage pool selection on project ID, you need to type storage pools with the IDs of each project.

Storage pools can be assigned multiple types and values and the same value can be assigned to multiple storage pools. In the project example, one storage pool can be typed with multiple project IDs and one project ID can be the type for multiple storage pools.

Storage pools can carry types for multiple purposes. For example, if selection is sometimes based upon user ID and other times based upon project ID, each storage pool can be assigned user IDs and project IDs. Selection based on one criteria is

not impacted by selection based on the other criteria because, in the above example, the types are independent.

The `CHGSPT` command adds and removes types from storage pools.

Examples of Storage Pool Selection

Chapter 3, “Changing Storage Pool Selection” contains descriptions of each alternate storage pool selection template available with Vault. It also describes the storage pool types needed for each alternate template. See Appendix A, “Work Example on Customizing Storage Pool Selection” for examples of customizing storage pool selection.

Changing Storage Pool Selection

You customize storage pool selection by changing the SQL and control data stored in the five Vault tables described. You change the data by running SQL load programs provided with Vault installation procedures. These programs replace existing data in all five tables with new data.

When you execute the load programs, Vault subjects the new storage pool logic to several consistency checks to guard against errors.

Selecting and Installing an Alternate Template

You might find that one of the supplied alternate templates satisfies your needs for customizing storage pool selection. If you can use one of the alternate templates, you do not need to alter any of the SQL code that must be loaded. However, you might need to assign types to the storage pools allocated to Vault for the alternate templates to work properly. Use the `CHGSPT` command to add or remove types from each of your storage pools.

Designing and Installing Custom Logic

If none of the alternate templates suits your purposes, you can design your own selection logic. Chapter 3, “Changing Storage Pool Selection” outlines the steps involved in installing an alternate template for storage pool selection.

Recovering from Errors

Customizing storage pool selection requires that your SQL code works correctly and that you have typed storage pools properly. You can test your custom SQL code before installing it. But in either case, you must qualify your alternate logic by installing and testing it with your Vault database.

Preventing Installation of Bad Logic

During the load process, Vault tests new storage pool selection code for consistency. The checks include syntax checking, verification that logic steps named or numbered in the control steps have been provided, and that there is no recursion. If Vault detects any errors, it does not install the new logic.

Vault does not verify that you have correctly typed the storage pools. By testing the new logic can you assure that it is functioning properly.

Logic Errors That Can Occur

Logic failure usually means that Vault cannot find an appropriate storage pool for a file. When Vault cannot find a pool, the transaction fails with an error message that specifies which pool filter Vault was using. The command can execute successfully after you identify and correct the selection logic.

New Logic Stores Files in the Wrong Storage Pool

A logic failure can result in a file stored in the wrong storage pool. If this happens, the sequence of selection queries can be incorrect or storage pools might not be properly typed. You can move files from the wrong storage pool to the correct one using the `CHGFSP` command described earlier in this chapter.

Bad Logic Is Installed

The installation procedure saves the most recent production logic in backup tables. If the newly installed code does not work, it can be replaced quickly with the most recent production logic.

Chapter 3

Changing Storage Pool Selection

This chapter provides information about changing the storage pool logic used by Vault. This logic determines which storage pool to use for storing a file.

- Installing a Selection Template
- Designing and Installing Custom Selection Logic
- Selection Flow Diagram
- Creating the Control File
- Creating the Selection Query File
- Creating the Pool Filter File
- Default Selection
- Selecting by File Classification (PUB, PRO, PRI)
- Selecting by Project ID
- Selecting by File Status
- Selecting by Part Number
- Selecting by User ID
- Selecting by User-defined File Type
- Selecting by Released Status
- Selecting by File Classification and Owner

Installing a Selection Template

Each alternate storage pool template provided with Vault has its own requirements for assigning types to storage pools. The instructions to install an alternate storage pool template is as follows:

1. Select one of the alternate templates. The last half of this chapter provides a detailed description of each alternate template and its prerequisite storage pool types.
2. Assign types to the storage pools as needed for the selection template you chose. For instance, if you chose to assign storage pools by user ID, you must assign each of your user IDs to one or more storage pools.

   Use the Vault command `CHGSPT` to add or remove types on your storage pools. Do not remove old storage pool types before adding new ones if they are independent. For instance, if you are changing storage pool selection from one based on user ID to one based on project ID, you can have storage pools typed by both user and project IDs without causing problems.
3. Using any standard text editor at your site, edit the script file `ldedmspl` found in directory `$EPD_HOME/install.` Uncomment the line containing the name of the template you want to install. For all other lines containing template names preceded by the label FD1=, indicate that they are comments.
4. Load the selected template into temporary EDM SQL tables by running the script file `ldedmspl` that you just edited.
5. Run the script file `edmispl`, if the load is successful and you want to make the code just loaded the production version. This script checks the temporary tables for logic errors. If none are found, it swaps the current production logic with that stored in the temporary tables.
6. Test the new production logic. Try storing files that fit every category of storage pool so that each pool filter is exercised.
7. If you find errors, you can back out the selection logic most recently installed by running the script `edmbspl` which restores the system to its status prior to the step above.

Designing and Installing Custom Selection Logic

You can design your own storage pool selection logic based upon the SQL queries. The major steps are to plan the selection process, code the three control files, and install the new selection logic. Follow the instructions below to design your own template for storage pool selection. See Appendix A, “Tutorial on Customizing Storage Pool Selection” for a tutorial on doing this.

Planning the Selection Process

1. Design the selection flow.
   a. Order the queries to select the most important criteria when more than one criteria matches the data.
   b. Terminate each set of branches with a pool filter.
   c. Be sure to include a catch-all category for files that meet none of your selection queries.
   d. Assign unique names to each selection query and pool filter.
   e. Number the steps.
   f. The blank Selection Flow Diagram at the end of this chapter can help you visualize the logic.
2. Assign types to the storage pools as appropriate for each of the pool filters you design.
   a. Use the Vault command `CHGSPT` to type your storage pools.
   b. You do not have to remove old storage pool types before adding new ones if they are orthogonal.

Coding the Three Control Files

1. Code the control logic. Each capsule in the selection flow diagram corresponds to one entry of control logic. Indicate the following:
   - The name of the selection query described in the capsule
   - The name of the next step when the condition tests true
   - The name when the condition tests false
   - Whether the next step is another query or a pool filter
2. Code the selection queries corresponding to the tests described in each capsule on the Selection Flow Diagram. You need one entry for each unique selection. If the same test is used in more than one capsule, you need to code it only once.

Use the Selection Flow Diagram to format the data correctly. This is to specify the SQL SELECT COUNT (*) queries correctly. Vault Database Tables is indispensable to this task. Code the pool filters corresponding to the logic described in each terminating box on the Selection Flow Diagram. You need one entry for each unique selection. If the same logic is used in more than one box, you need to code it only once.

See Appendix A, “Work Example on Customizing Storage Pool Selection”of this document.

Installing the New Selection Logic

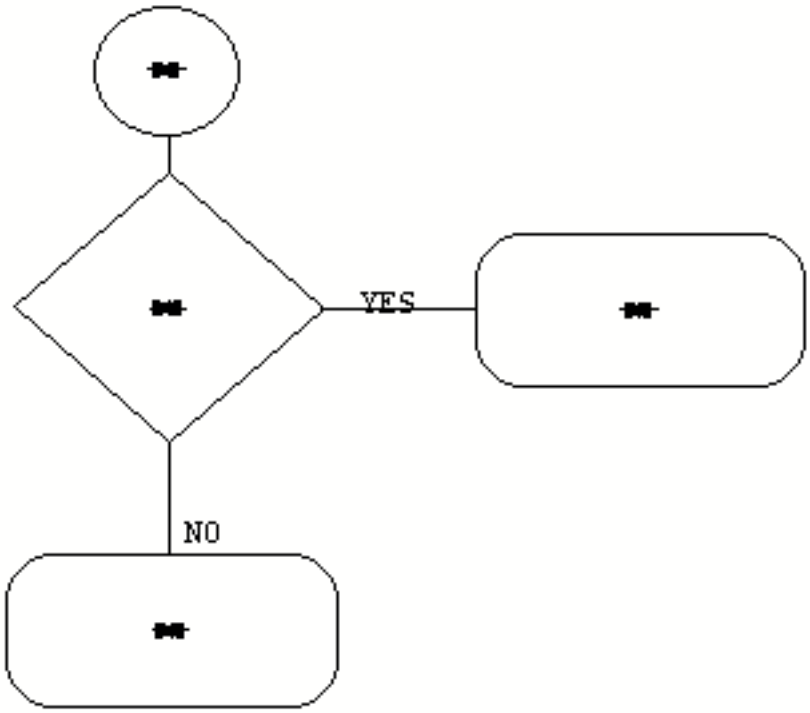
1. Using any standard text editor, edit the script file `ldedmspl` found in directory `/$EPD_HOME/install.`
   a. For all lines that contain template names preceded by the labels `FD1=`, `FD2=`, and `FD3=`, indicate that they are comments.
   b. Add a line for each file you have prepared.
   - **C**ontrol file
   - Selection queries
   - Pool filters
2. Load the selected template into temporary EDM SQL tables by running the script file `ldedmspl` that you just edited. If the load fails, check for coding errors such as character data entered into numeric-only fields. Correct the errors and re-run.
3. If you want to make the code just loaded the production version, run script file `edmispl.` This script checks the temporary tables for logic errors. If none are found, it swaps the current production logic with the logic in the temporary tables. If the install fails, return to steps 3, 4, and 5 to correct the error. If you want to check the new code for logic errors but not install it, run the script `edmespl.`
4. Test the new production logic. Store files that fit every category of storage pool to exercise all pool filters.
5. If you find errors, exit the selection logic most recently installed. Do so by running the script `edmbspl.` It restores the system to its original status.

Selection Flow Diagram

Use the Blank Selection Flow Diagram shown below to help visualize selection logic.

- Use the circles to sequence selection queries.
- Use a diamond for each selection query.
- Use a capsule for each pool filter.
- Label each selection query and pool filter on the adjacent line.
- Terminate each branch with a pool filter capsule.

Figure 3-1 Blank Selection Flow Diagram



Creating the Control File

Format the data in the control file according to the table below.

Table 3-1 Control File Column Definitions

Columns	Instructions
1-5	Enter a unique sequence number for the control step. First step uses 00000.
7-38	Enter the unique name of a Selection Query to use for the step.
40-44	Enter next step number to use when query returns a Found condition. Use -1 when next step is a pool filter instead of a selection query.
46-50	Enter next step number or -1 for when query returns a Not Found condition.
52-83	If Found Seq is -1, enter the name of a Pool Filter.
85-116	If Not Found Seq is -1, enter the name of a Pool Filter.
118-357	Optionally, enter a description.

The following is an example of the format of a control file:

```
1...5 7.................38  40.44  46.50  52.....................83
SEQ   SELECTION QUERY NAME  FOUND  NOT      POOLFILTER IF FOUND
NUM                         SEQ   FOUND


00000--------------------  -----  ------  -----------------------



85.....................116 118.................................357
POOL FILTER IF NOT FOUND     DESCRIPTION (optional)
```

Creating the Selection Query File

Format the data in the selection query file according to the table below.

Table 3-2 Selection Query File Column Definitions

Columns	Instructions
1-32	Enter the name of the selection query.
34-35	Enter the sequence number of the selection query.
37-276	When the sequence number is 00, enter a description of the query. For sequence numbers 01 and higher, enter the SQL SELECT COUNT (*) query using up to 240 characters for each entry.

The following is an example of the format of a selection query file:

```
1.....................32
SELECTION QUERY NAME
-------------------------



SEQ DESCRIPTION/QUERY
00    ------------------------------------------------------------
      ------------------------------------------------------------
      ------------------------------------------------------------
01    SELECT COUNT (*) FROM ----------------------------------
      ------------------------------------------------------------
      ------------------------------------------------------------
02    ------------------------------------------------------------
      ------------------------------------------------------------
      ------------------------------------------------------------
```

Creating the Pool Filter File

Format the data in the pool filter file according to the table below.

Table 3-3 Pool Filter File Column Definitions

Columns	Instructions
1-32	Enter the name of the pool filter.
34-35	Enter the sequence number of the pool filter.
37-276	When sequence number is 00, enter a description of the filter. For sequence numbers 01 and higher, enter the WHERE clause to complete the query SELECT FROM DM_STORAGE_POOL WHERE .... Use up to 240 characters for each entry.

The following is an example of the format of a pool filter file:

```
1.....................32
POOL FILTER NAME
------------------------



34-35       37.............................................276
DESCRIPTION/WHERE CLAUSE
00          ------------------------------------------------
            ------------------------------------------------
            ------------------------------------------------
01          WHERE -----------------------------------------
            ------------------------------------------------
            ------------------------------------------------
02          ------------------------------------------------
            ------------------------------------------------
            ------------------------------------------------
```

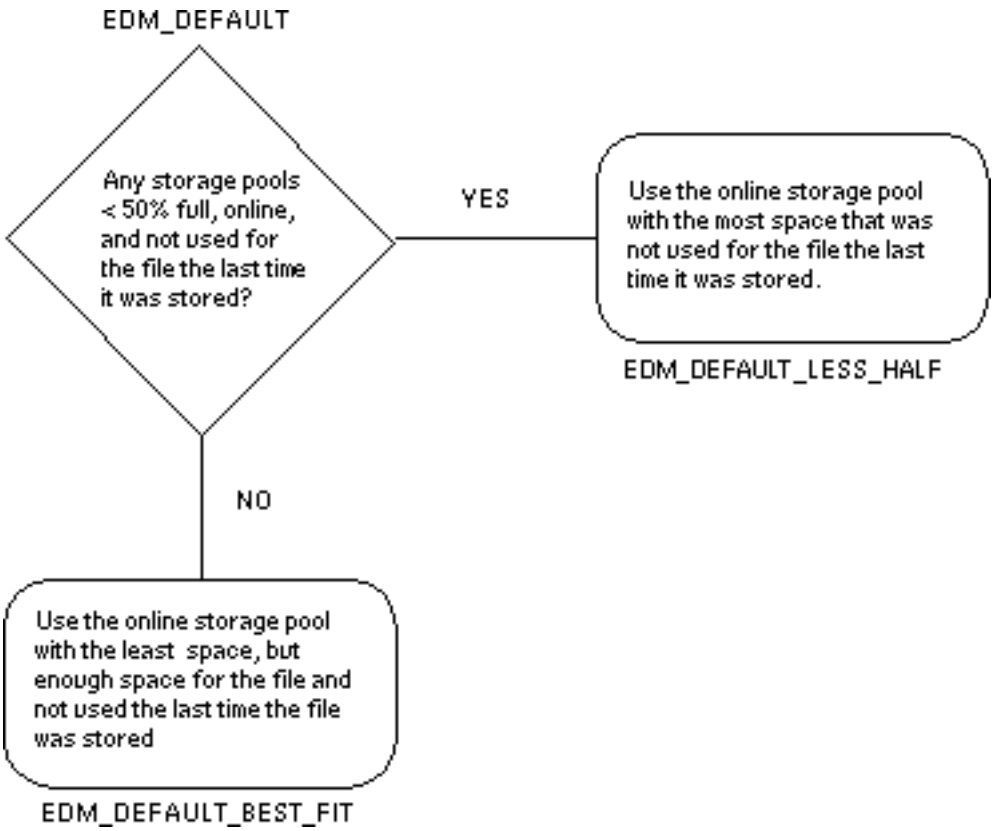
Default Selection

The default storage pool selection is based on characteristics of the set of storage pools. The default selection uses a round-robin algorithm until a pool is 50% full. It then packs the pool to capacity. The database is queried to see if there are any storage pools less than 50% full, online, and not used for the file the last time it was stored.

If this condition is met, the storage pool chosen is the one that has the maximum space available and satisfies the selection criteria. If this condition is not met, the storage pool chosen is the one that is most full, but with still enough space to hold the file, and not used for the file the last time it was stored.

Storage pool types are not needed. If present, they do not affect the algorithm. The default logic is illustrated below.

Figure 3-2 Default Storage Pool Selection Logic



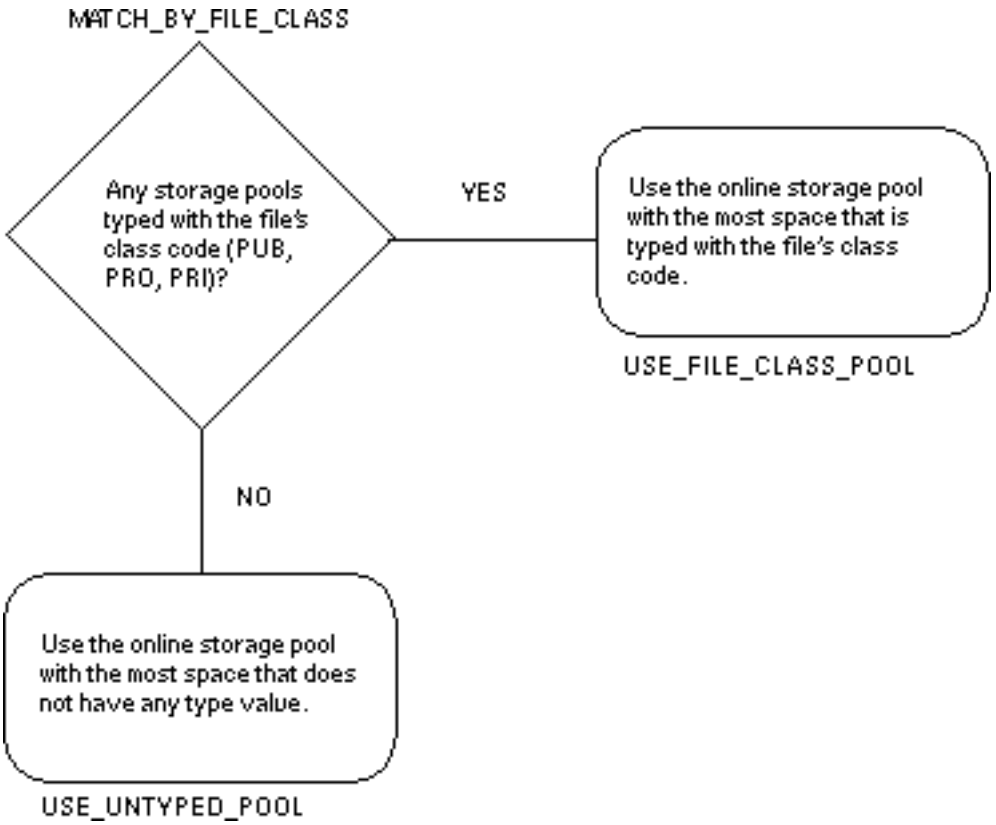
Selecting by File Classification (PUB, PRO, PRI)

This template selects storage pools based on a file's classification.

Algorithm

The set of storage pools are examined to see if any are typed with the classification of the file (PUB, PRO, PRI). If so, one is chosen. If not, an untyped pool is chosen. The logic is shown in the figure below.

Figure 3-3 Logic for Selection by File Classification



Storage Pool Types

Type one or more storage pools with each of the possible file classifications. If any file classification is not represented, then leave one or more pools without any types.

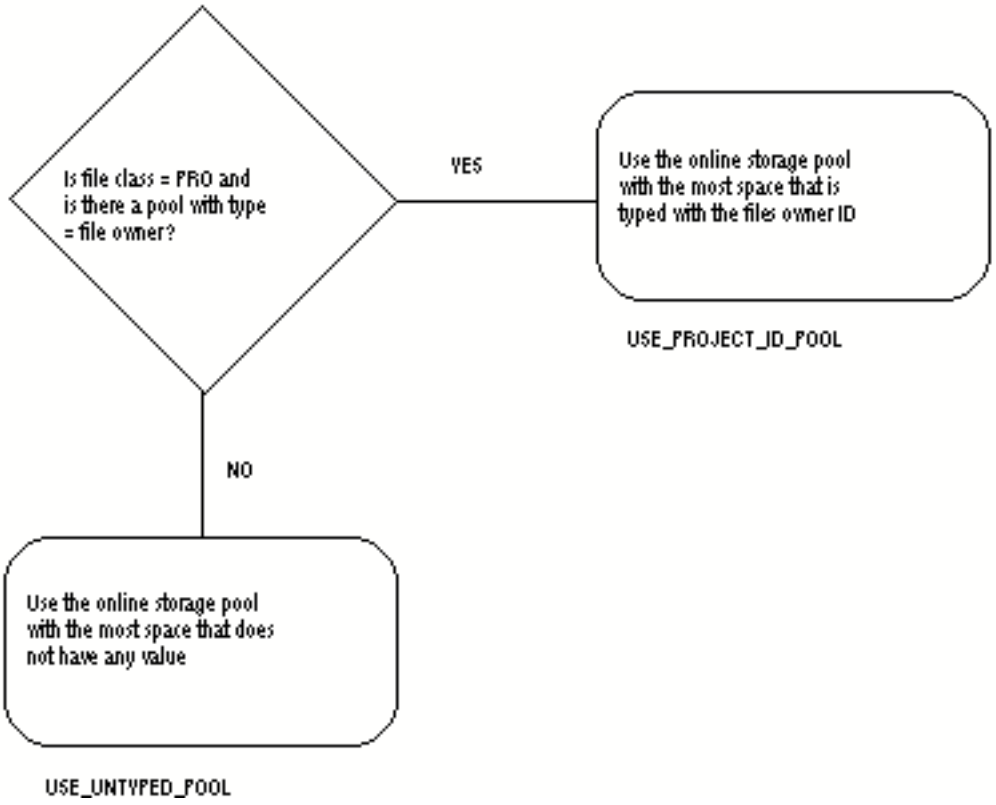
Selecting by Project ID

This template selects storage pools based on a file's project ID.

Algorithm

If the file is a project file and if there are any storage pools typed with the file's Project ID, use one of those pools. If not, use an untyped pool. The logic is shown in the figure below.

Figure 3-4 Logic for Selection by Project ID



Storage Pool Types

Type one or more storage pools with each of the possible project IDs. Any one storage pool can be typed with multiple project IDs. Leave one or more storage pools untyped so that files with project IDs not represented in the set of storage pools can be stored.

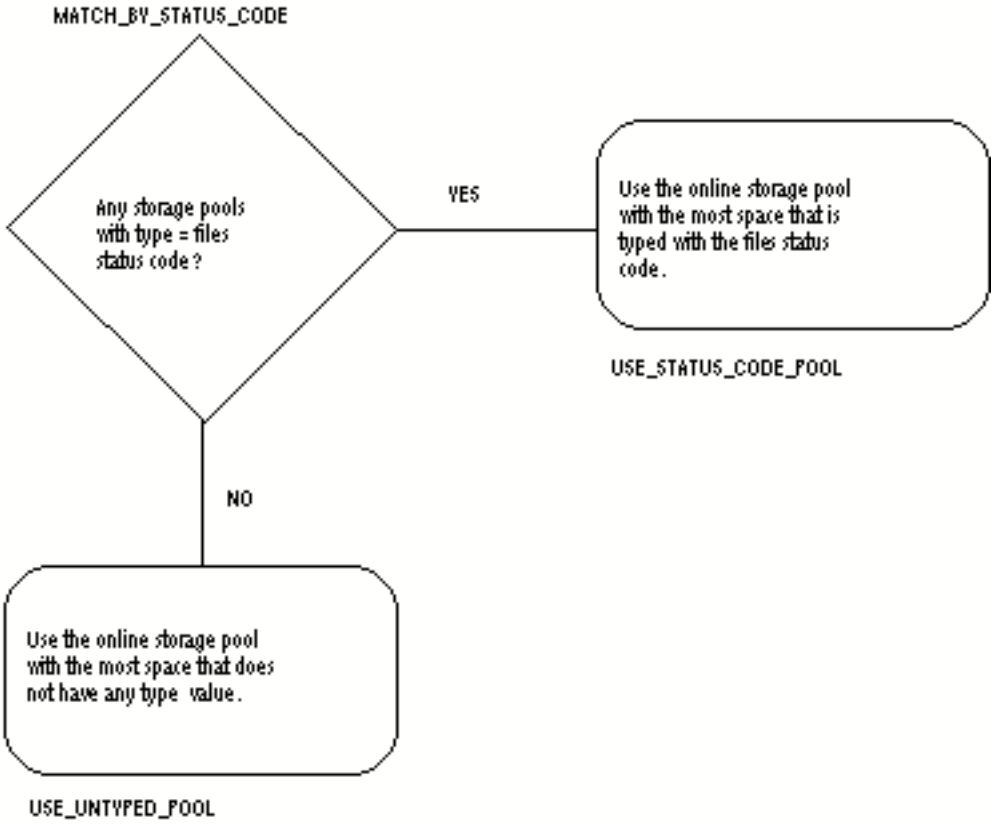
Selecting by File Status

This template selects storage pools based on a file's status.

Algorithm

If there are any storage pools typed with the status code of the file, use one of those pools. If not, use an untyped pool. The logic is shown in the figure below.

Figure 3-5 Logic for Selection by File Status



Storage Pool Types

Type one or more storage pools with each of status codes in the public and project authority schemes. Leave one or more storage pools untyped so that files with status codes not represented in the set of storage pools can be stored.

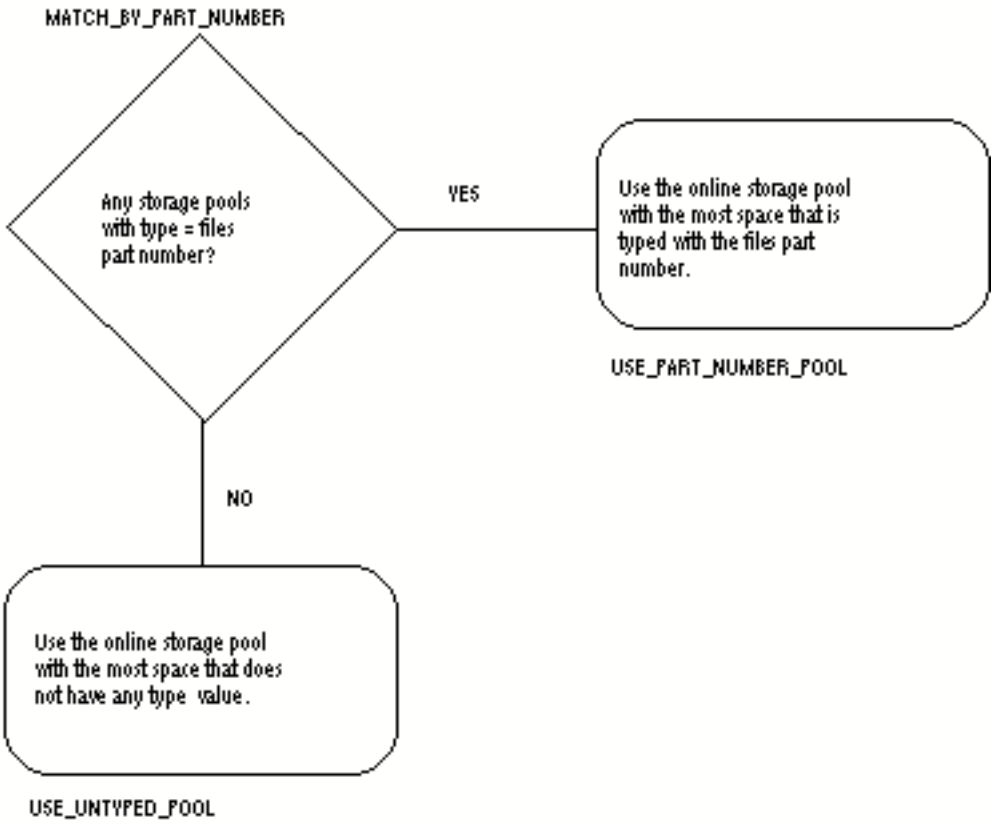
Selecting by Part Number

This template selects storage pools based on a file's part number.

Algorithm

The set of storage pools is examined to see if any are typed with the user-assigned part number. If so, one is chosen. If not, or if no part number is present in the file, Vault selects an untyped pool. The logic is shown in the figure below.

Figure 3-6 Logic for Selection by Part Number



Storage Pool Types

Type one or more storage pools with each of the possible part numbers. Any one storage pool can be typed with multiple part numbers. Leave one or more storage pools untyped so that files with part numbers not represented in the set of storage pools can be stored.

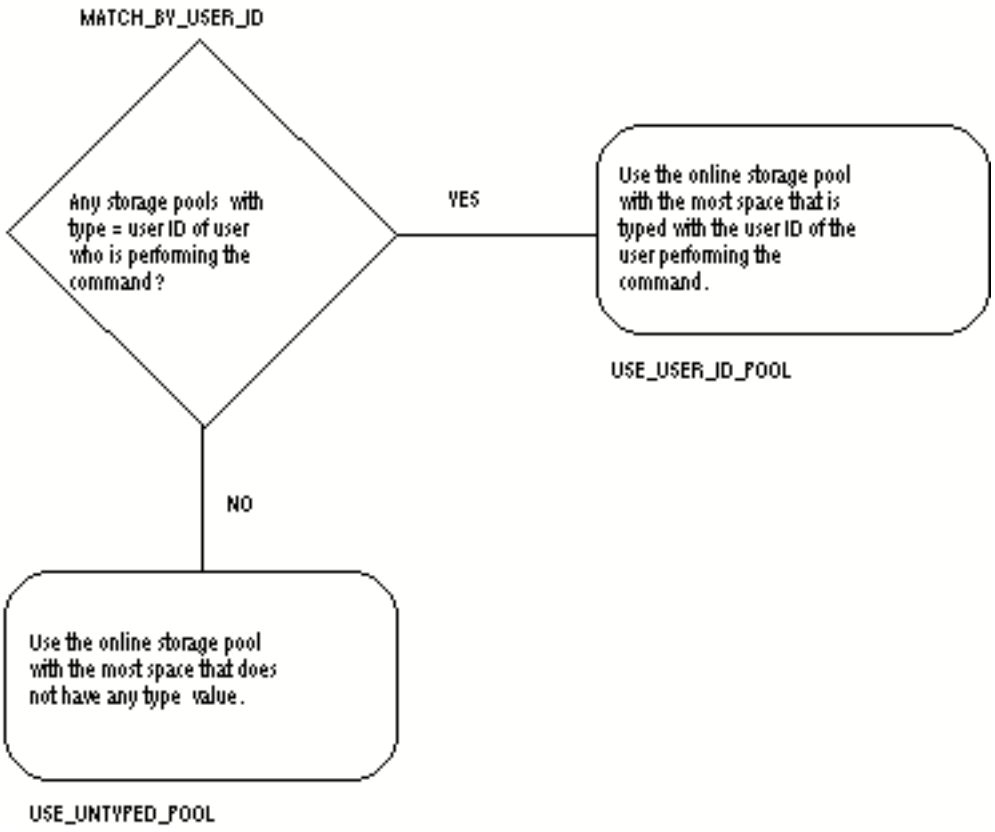
Selecting by User ID

This template selects storage pools based on the user ID of the person performing the command.

Algorithm

If there are any storage pools typed with the user ID of the user performing the command, use one of those pools. If not, use an untyped pool. The logic is shown in the figure below.

Figure 3-7 Logic for Selection by User ID



Storage Pool Types

Type one or more storage pools with each of the possible user IDs. Any storage pool can be typed with multiple user IDs. Leave one or more storage pools untyped so that files with user IDs not represented in the set of storage pools can be stored.

Please note: All Vault users need not have storage pools typed with their IDs. Only those Vault users who are authorized to perform file store commands (`STORE, LOAD, UPDATE, REPLACE, COPY, RECSF, AND RECSP`) need to have storage pools typed with their IDs.

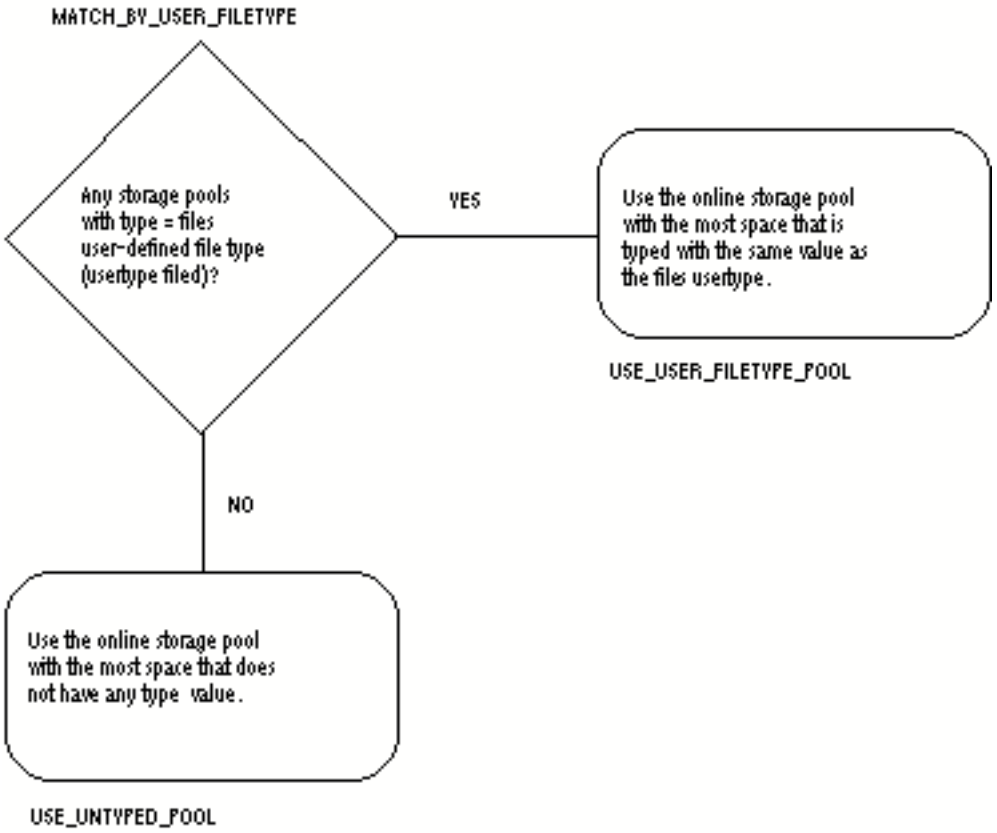
Selecting by User-defined File Type

This template selects storage pools based on a file's user defined file type.

Algorithm

If there are any storage pools typed with the user defined file type, use one of those pools. If not, use an untyped pool. The logic is shown in the figure below.

Figure 3-8 Logic for Selection by User-defined Type



Storage Pool Types

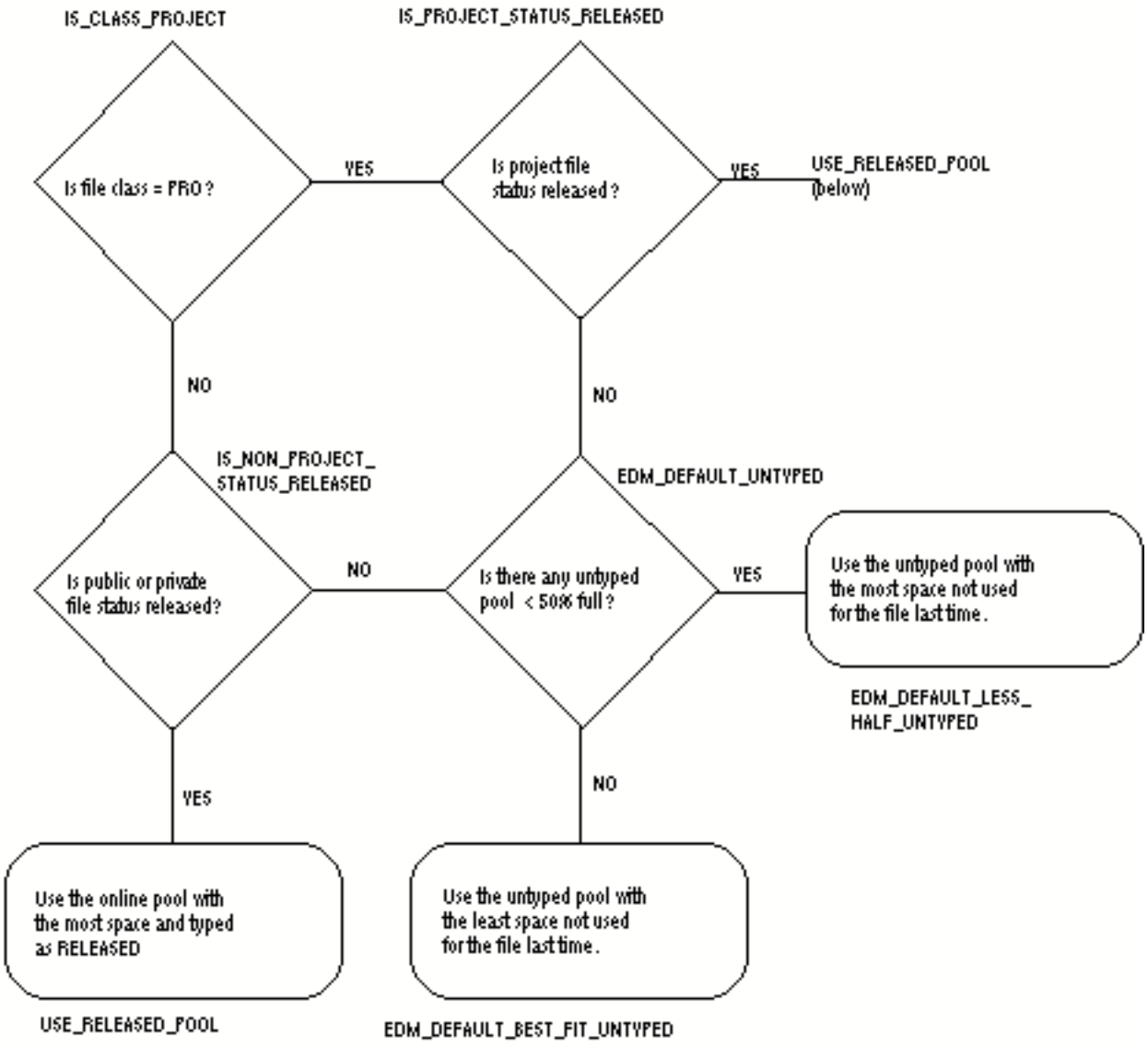
Type one or more storage pools with each of the possible user defined file types. Leave one or more storage pools untyped so that files with user defined file types not represented in the set of storage pools can be stored.

Selecting by Released Status

This template selects storage pools based on a file's released status.

If the file's status code is a released status code, use a storage pool typed as RELEASED. If the status code is not released, use the default algorithm modified for untyped pools. The logic is shown in the figure below.

Figure 3-9 Logic for Selection by Released Status



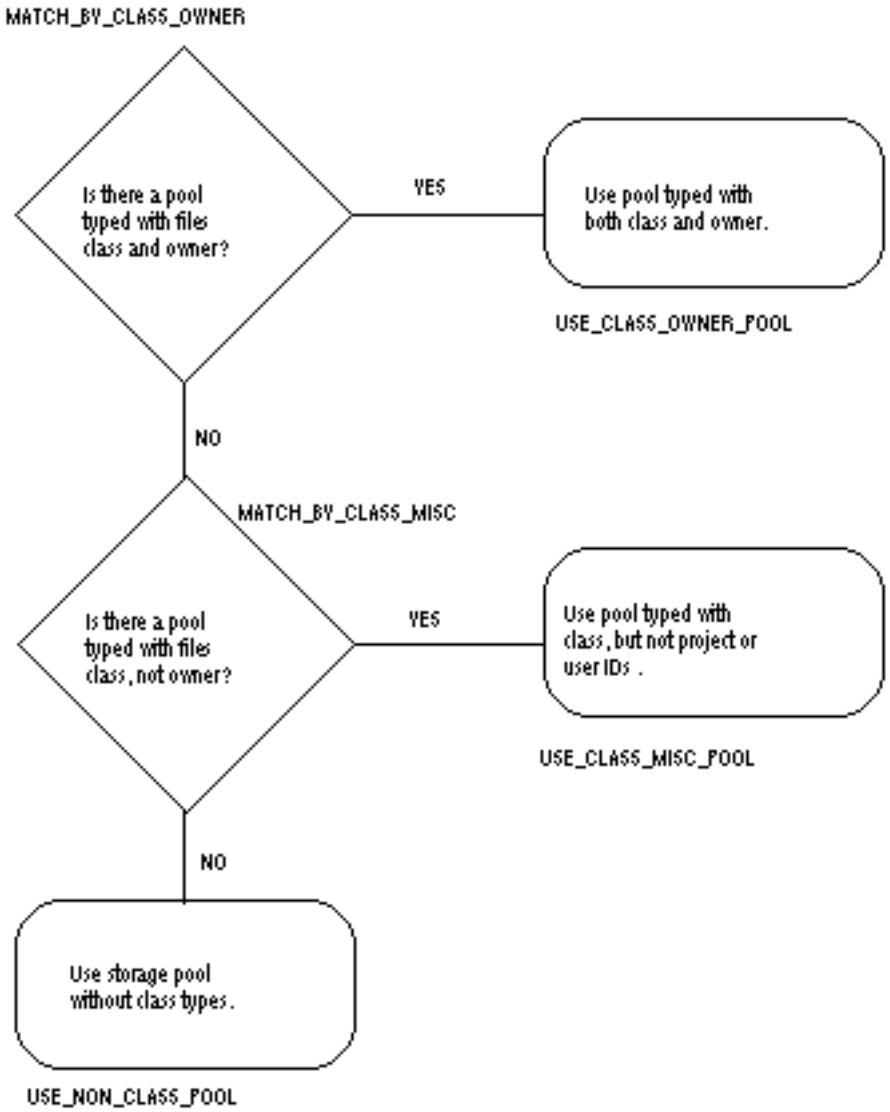
Type one or more storage pools as RELEASED. Leave at least two storage pools untyped.

Selecting by File Classification and Owner

This template selects storage pools based on a file's classification and owner.

Vault assigns files to pools typed with both classification and owner, if possible. When there is no pool typed with both, Vault assigns pools that are typed with the correct classification and without any owner types. When there is no pool typed with the file's classification, Vault assigns a pool without any class type. The logic is shown in the figure below.

Figure 3-10 Logic for Selection by File Classification and Owner



Type one or more storage pools with each of the possible file classifications. Type some or all of these pools also with owner IDs type PRO pools with project IDs and PRI pools with user IDs. If any file classification is not represented, then leave one or more pools without any classification types.

Chapter 4

Performing a Backup of Files

This chapter provides information about backing up of data. The information in this chapter applies to all Vault platforms. For more information, refer to Chapter 7, “System Administration Tasks For UNIX/NT”.

- Performing an Incremental Backup
- Deleting Old File Versions
- Performing a Universal Backup
- Enterprise Backup

Performing an Incremental Backup

To create a Vault incremental backup tape, use the IBKUP command. Execute this command when locally logged into the Vault server. The server need not have a local tape device.

Copying Files

The IBKUP command selectively copies onto the tape, those files that have been added or changed since the previous IBKUP command was executed. The initial execution of the IBKUP command backs up all Vault files.

Making Two Copies

Make two copies on two separate tapes at the same time. The two tape drives in a dual copy operation must be attached to the same type of operating system.

Vault uses magnetic tape to back up and archive its controlled storage areas. To reduce the risk of data loss due to a media or tape drive failure, you can use the Vault dual copy facility to make two backup tapes at the same time. Alternatively, you can make a copy of the backup tape with your OS utilities. The difference between the two methods is that with the Vault dual copy utility the result is two tapes with different volume identifiers and with the OS copy utility the result is two tapes with the same volume IDs.

Warning

Do not Change Tape Media between Universal Backups

While performing an incremental backup, use the same type of tape media as used by any other incremental backup since the last universal backup. This is important when you want to recover a storage pool. The RECSP command might need to read data from several tapes during its operation. You cannot change media during execution of the RECSP command.

Tape Information

During the initial Vault software installation, each tape device that can be used to create incremental backup tapes or perform any tape command must be logically identified to Vault. If you have only one tape device it must be identified with the tape unit name TAPE1. If you have a second tape device it must have the name TAPE2. Third and fourth tape devices must be assigned the names TAPE3 and TAPE4, respectively.

You need to know which name belongs to which tape device. When you execute the `IBKUP` command, enter the tape unit name(s) to identify the tape device(s) on which you are creating the tape(s).

Tape Labels: Label the tape before executing the `IBKUP` command. Each incremental backup tape needs its own unique tape serial number (up to six letters and/or numbers) and Vault uses the label as the tape serial number. When you execute the `IBKUP` command, you can use the default value for the tape serial number. The default is +. This instructs Vault to use the serial number (or label) already on the tape.

Use serial numbers for identification purposes. For example, if you create one incremental backup tape each day, you can use the month, day, and year to identify your tapes (032394 for the tape created on March 23, 1994).

Use operating system commands to initialize (label) the tapes. If your operating system does not have such a utility, then use the utility provided with Vault.

Please note: Do not use the `tapenlabel` command to initialize 1/2-inch tapes. Use the `tapelabel` command instead.

Tape Density: Vault does not control the tape density setting. Use your operating system commands to change the current tape device setting.

Tape Becomes Full: When the tape becomes full, Vault displays a message instructing you to mount an additional tape. You can take one of two actions:

- Attach another tape to the tape device, enter its tape serial name, and press the processing key to continue the backup.
- Or, exit the `IBKUP` command by pressing the F3 or F4 function key.

Appending to Tapes/Two-Copy Backups

You can append to both new and used tapes. When you append to new tapes, Vault simply starts at the beginning of the tape. When you specify the append option for a two-copy backup, both tapes must be new or must be the two copies from an existing two-copy backup set.

You can enter two tape units and two serial numbers to produce two copies of the incremental backup at the same time. You can enter one tape unit and one serial number to produce one copy.

If you instruct Vault to append to a tape that is half of a two-copy set with the second copy not included in the current backup operation, Vault returns an error

message and does not perform the backup. After including the second copy in the two-copy set execute the command again.

Vault performs the same checks on the second tape unit and serial number that it performs on the first unit and number. Vault also verifies that two different units are specified.

When creating a two-copy backup or appending to a two-copy backup, an end-of-tape condition on one tape stops the backup process. In other words, the amount of data that Vault can copy to a two-copy set is determined by the device having the least capacity. Similarly, the speed of the backup process is determined by the speed of the slower tape unit.

Deleting Old File Versions after Incremental Backup

After the `IBKUP` command is executed, use the `DELOV` command to delete old file versions from the database and reclaim storage from the storage pools. For this procedure, see the “Deleting Old File Versions” on page 6 later in this chapter.

The incremental backup command always starts at the beginning of the backup sequence, backing up all new or modified files not previously backed up.

For example, when you execute the `IBKUP` and `DELOV` commands in either of the following sequences, the incremental backup and old file deletion results are the same.

1. `IBKUP`
2. `DELOV`
3. `IBKUP`
4. `DELOV`

1. `IBKUP`
2. `IBKUP`
3. `DELOV`

The backup tapes produced by the `IBKUP` command are required for the `RECSF` (recover a single file) and the `RECSP` (recover a storage pool) commands.

IBKUP Command Parameters

Table 4-1 The parameters for the IBKUP command

Keyword	Parameter Description
TAPEUNIT	Name to represent the tape device(s). Enter: **TAPE1** or **TAPE2** or **TAPE3** or **TAPE4**. Or enter two devices separated by a comma, for example, **TAPE1,TAPE3**, to create two backup tapes at the same time. Spaces are not allowed. Default is TAPE1
TAPENUM	Tape identifier(s). Leave blank to use the identifier already on the tape. Enter 6 or fewer letters and/or numbers to make one backup. Enter two identifiers separated by a comma to make two copies. One or both can use the SCRTCH identifier. SCRTCH instructs Vault to use the identifier already on the tape. Spaces are not allowed.Default is SCRTCH
TAPPEND	Indicates whether to append to the tape. Enter **Y** for Yes or **N** for No. Default is N

Using the Command-Line Format

In this example, all new and modified files can be backed up on the tape attached to the default tape unit (TAPE 1).

```
ciibkup
EDM> TAPENUM=020494
```

In this example, all new and modified files can be backed up on the tapes attached to TAPE2 and TAPE3. Vault can use the serial numbers already on the tapes because nothing is specified for `TAPENUM`. Vault can append the backup data, to data already on the tapes.

```
ciibkup
EDM> TAPEUNIT=(TAPE2,TAPE3) TAPPEND=Y
```

Deleting Old File Versions

To physically delete old versions of Vault files and reclaim storage space in the Vault storage pools, use the DELOV command. This command deletes only those file versions that have previously been backed up on tape with the IBKUP (incremental backup) command.

Preservation of File Attributes

When Vault recognizes a file as either a current, backed up, archived, or old version, then its attributes (both Vault and user-defined attributes) are preserved. In other words, if you can use a Vault command to access a file, then that file's attributes are not deleted by maintenance operations.

When you execute a universal backup and clean out the Backup Table, you can create a situation where an old version has no entries in the Backup Table. If you then execute the DELOV command, the operation deletes file attributes as well as the old versions, for any old versions without entries in the Backup Table.

Executing the IBKUP and DELOV Commands

You can execute the IBKUP and DELOV commands in sequence or you can execute them at different times, depending on your needs. For example:

- You can create incremental backup tapes following each work shift and delete old file versions on weekends only.
- You can create incremental backup tapes and delete old file versions at the end of each work day.

DELOV Messages

After the files have been deleted, the total number of deleted files is displayed. A message is displayed if there are no old file versions to be deleted.

DELOV Parameters

There are no parameters for this command.

Using the Command-Line Format

This deletes all old file versions after you press the processing key.

```
CIDELOV
```

Performing a Universal Backup

Use the UBKUP command to create a backup tape containing copies of all files in the database. You must have the UBKUP command in your public command list to execute this command.

The backup created is a snapshot of the storage pools at the time the command is executed. The universal backup command provides database protection in case the storage medium fails.

The UBKUP command detaches the tape drive, if it is still attached when the universal backup is complete.

Execute the UBKUP command to back up the entire database to tape. Execute this command only when locally logged in to the Vault server. Vault uses an operating system utility to copy the files. Vault can be active while the backup is being done. The information here is general and you cannot execute this command without the platform-specific details. See Chapter 7, “System Administration Tasks For UNIX/NT”if you have Vault on a UNIX system. Use the command-line format to execute the UBKUP command.

Operating System Utility

The contents of a storage pool are backed up on a tape using an operating system utility. Multiple storage pools are placed on a physical tape. This is dependent on the operating system. The storage pools that are copied to tape are grouped by physical disk volume. This minimizes tape mounting operations during the recovery process.

One storage pool is backed up at a time. While a storage pool is being backed up no new files are written to that storage pool. Depending on the local operating system, the backup can be performed while the Vault system is active.

Restarting Universal Backup

If a previous universal backup was aborted during processing, you can restart the process at the point it was aborted. Restarting a universal backup is no longer keyed on a physical volume. It is based on a storage pool.

Pausing Universal Backup

To change a tape between physical volumes, the PAUSE keyword must be set to Yes.

Please note: You cannot use the Vault remote tape facility for a universal backup. Also, during a universal backup you cannot make dual copies of files.

Maintaining Entries in the Backup Table

After all storage pools have been copied to tape, the universal backup utility gives you the opportunity to clean up the Backup Table by deleting entries that you no longer need. If you want to clean up the Backup Table you must indicate the number of cycles you want to save. A cycle is the time between two universal backups.

Designating the number of cycles as one (1) means the entire Backup Table can be cleaned out. Only the current universal backup can remain.

Designating the number of cycles at two (2) means the Backup Table is cleaned out up to the previous universal backup. This leaves only the current and previous universal backups.

Designating the number of cycles as three (3) means the Backup Table is cleaned out up to the backup before the previous universal backup, and so on.

Preservation of File Attributes

When Vault recognizes a file as either a current, backed up, archived, or old version, then its attributes (both Vault and user-defined attributes) are preserved. In other words, if you can use a Vault command to access a file, then that file's attributes are not deleted by maintenance operations.

When you execute a universal backup and clean out the Backup Table, you can create a situation where an old version has no entries in the Backup Table. If you then execute the DELOV command, the operation deletes file attributes as well as the old versions, for any old versions without entries in the Backup Table.

Audit Information

Vault generates an audit trail to indicate the storage pool name, date and time of backup, and other platform specific information. Vault places a copy of the audit information in your local storage area under the name UBKUP.EDMAUDIT. You can append the new audit information to the existing file or you can erase the content of the existing file so that it includes only data from the current universal backup. You use the audit trail to perform the storage pool recovery command (RECSP).

When the restart option is set to yes, Vault appends the new audit trail to the current file. The backup is not considered to be a new backup because it is being restarted. The audit file includes ASA print control characters.

Universal Backup Configuration File

When you issue the UBKUP command you must have a universal backup configuration file in the current directory. On UNIX systems, the file is `ubkup.config`.

The backup configuration file has the following properties.

- It can include any or all of the keyword=value specifications on the command line.
- There should be one keyword=value per line.
- Blank lines are allowed.
- Specifications on the command line take precedence over specifications in the configuration file.
- Comments are delimited by /* and */ as in the example below. Comments cannot span multiple lines. You must have a set of delimiters on each line that includes a comment.

```
RESTART=Y   /* This is a comment. */
/* This is a continuation of the above comment. */
```

- The configuration file can be empty. In this case the UBKUP command uses the default values for parameters not specified on the command line.

The UBKUPNAME Variable: The UBKUPNAME variable in the `ubkup.config` file allows you to specify the name of an executable file to be used for the backup of storage pools. Specify the complete path name of the executable file. For example,

```
UBKUPNAME=$EDM_HOME/bin/ubucpio
```

UBKUP Command Parameters

Table 4-2 The parameters for the UBKUP command

Keyword	Parameter Description
TAPEUNIT	Name assigned to the local physical tape device of the local system. Enter TAPE1 or TAPE2 or TAPE3 or TAPE4. Default is TAPE1.
TAPENUM	Tape identifier name. Enter 6 or fewer alphanumeric characters.
RESTART	Indicates whether or not the backup should start at the point at which it was stopped the last time it was executed. Enter Y for Yes or N for no. Default is N.
CLEAR	Indicates whether or not the Backup Table should be cleared. Enter Y for Yes or N for No. Default is N.
CYCLES	Number that specifies how many universal backup cycles to keep. Enter a number from 01 to 99. Required if CLEAR=Y. Default is 01.
INPUT	A one-character value that indicates if storage pools to be backed up are listed in the command or in a file. Enter L for Listed in the command or F for listed in a File. Default is L.
POOLS	Used in conjunction with the INPUT keyword. Indicates the name of the storage pools to process. Enter * (asterisk) when INPUT=L to indicate all storage pools. Or, enter a list of pool names enclosed in parentheses, not including blanks, and separated by commas to specify those storage pools when INPUT=L. Enter a local file name when INPUT=F to indicate that the file contains the names of the storage pools to be backed up. The file must be a text file and each storage pool name must be on a separate line in the file. The file must be in your current working directory or you must specify the full path name. Default is *.
APPEND	Append new audit information to audit file? Enter Y for Yes or N for No. N means that Vault erases the content of the existing audit file. Default is Yes.
COMPACT	Compress the data as the pools are backed up? Enter Y for Yes or N for No. Default is No.
PARTIAL	Continue the backup even if some of the pools are currently unavailable? Enter Y for Yes or N for No. Default is No.
PAUSE	Query the operator during backup? Enter Y for Yes or N for No. For example, if a pool is busy and PAUSE=Yes, you can instruct Vault to try again. When PAUSE=No, the backup aborts. Default is Yes.

Using the Command-Line Format

```
ciubkup RESTART=Y CLEAR=Y CYCLES=2 INPUT=L,
POOLS=(pool1,pool2,pool3)
```

In this example, Vault can start copying files at the point where it stopped during the previous execution of the UBKUP command. The Backup Table can be cleared except for two cycles. Information for the current universal backup and the previous universal backup remains. The storage pools to be backed up are listed in the command. They are pool1, pool2, and pool3.

Enterprise Backup

Enterprise Backup allows you to customize the procedure for backing up your site's Vault information.

With Vault, you can include extensions to Vault taping commands in your backup procedure. When you enable these extensions, the Vault's taping commands direct the system to produce physical media using these extensions, instead of those traditionally used by default during Vault's incremental, archival, and universal backup and restoration procedures.

Please note: Enabling these extensions provides an alternative taping method, not an additional method.

Customizing Sample Scripts

This release includes a collection of sample scripts and files which you can copy and customize to include this functionality in your Enterprise Backup procedure. These are located in the install directory and are prefixed by the characters ebu_.

You must establish the variable ebuname, which identifies your site-specific Enterprise Backup procedures, before invoking the vault tape command. For incremental, archival, and universal backup and restoration, use the format:

```
% setenv ebuname path
```

The variable ebuname identifies your site-specific Enterprise Backup procedures, and path specifies the full path name of the script to be invoked.

For Universal Backup, use the format:

```
% setenv ebuname_UBKUP path
```

The variable ebuname identifies your site-specific Enterprise Backup procedures and path specifies the full path name of the script to be invoked

In order to invoke the Enterprise Backup extensions, you must set your EDM_TAPEx environment variable using the following syntax:

```
% setenv EDM_TAPEx @<ebuname>
```

The variable x represents a device number from 1 through 4. The special character @ denotes to the Vault tape command that the script invokes backup extensions, and the variable ebuname identifies your site-specific Enterprise Backup procedures.

When the tape command encounters this syntax, it attempts to map the value of the environment variable @ ebuname to that of the variable ebuname previously established.

Please note: The actual EDM_TAPEx device name (represented in the example by @ebuname) cannot exceed 6 characters, including the mandatory @.

For example, if you have included the syntax @ebuname in your backup procedure, the script directs the system to proceed with the Enterprise Backup. The taping command then invokes the script associated with either incremental and/or archival backup, if it encounters the following argument:

```
ebuname
```

A universal backup is invoked if it encounters the following argument:

```
ebuname_UBKUP
```

Using Third-Party Backup Software

Using these extensions, you can backup Vault data with a third-party backup package, such as Legato.

The Vault install directory also provides the following collection of scripts, located in the install directory, each of which provides a sample interface between Vault tape commands and Legato. These items are distinguished by the prefix ebu_. Remove this ebu_ prefix before implementation by copying the file as demonstrated in the following example:

```
cp ebu_LGTO.sh LGTO.sh
```

You can copy and customize each of these scripts to facilitate creating an interface to a different backup system.

Please note: Test these scripts thoroughly before implementing them.

Creating an Enterprise Database Table

You must create a new table, called DM_ENTERPRISE_BACKUP, before you use Enterprise Backup extensions. The install directory contains a file (`ebu_table_definition`) that provides the definition of this table.

To load the DM_ENTERPRISE_BACKUP table, include the following:

```
load_dm_enterprise_backup_LGTO.sh
```

Please note: The file `ebu_ciubkup`, located in the install directory, contains Enterprise Backup extensions for universal backup and incremental archive backup.

Chapter 5

Using Exabyte and DAT Tapes

This chapter provides information about using Exabyte and 4mm DAT (Digital Audio Tape) tapes with Vault. Exabyte tape drives are supported when they are attached to SunOS or Solaris platforms. Four millimeter DAT tapes are supported when they are attached to HP-UX systems.

- Background
- Vault Tape Formats
- Labeling Exabyte or DAT Tapes
- Restrictions and Limitations
- Setting the Capacity Value on Exabyte Tapes

Background

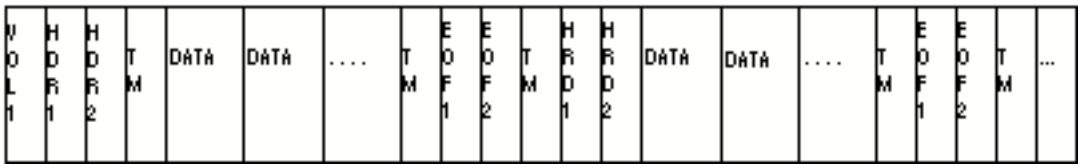
Vault uses an ANSI standard label scheme for Incremental Backup (`IBKUP`) and Archive (`ARCHIVE`) tapes. This label scheme requires that three tape marks, sometimes referred to as file marks, be written for each logical file written to the tape.

Please note: See ANSI Magnetic Tape Labels and File Structure for Information Exchange (ANSI X3.27-1978).

The format of the tape is illustrated below.

Exabyte and 4mm DAT drives support a tape mark function, but it is costly in terms of its duration and the amount of tape consumed. For example, Exabyte tapes take approximately 20-30 seconds to write 2,160 kilobytes (2,211,840 bytes or 2.1 megabytes) of data. This equates to 270 helical tracks of data.

Figure 5-1 Vault Tape Format



Vault Tape Formats

Vault supports three recording formats which are identified by the manner in which the tape is initialized (labeled).

- ANSI Standard

  This format uses three physical (real) tape marks per logical data file and is identical to 1/2” 9-track, DEC TK50, and 3480/3490E recording methods.

- Vault Exabyte or 4mm DAT Using One Real Tape Mark

  This format uses one real tape mark between files. Vault places a tape mark at the end of each file. The two other tape marks for each file are logical tape marks written by Vault. This method allows Vault to use tape marks for file positioning. It also improves the performance of Exabyte or 4mm DAT tapes over ANSI standard tapes because of the elimination of two real tape marks.

- Vault Exabyte or 4mm DAT without Real Tape Marks

  This format does not place any real tape marks between files. This method provides the best performance and capacity of the Exabyte or 4mm DAT drive.

  There is, however, a restriction for Exabyte. If the physical end-of-tape is reached, the Exabyte drive does not provide a mechanism for properly closing the tape. Therefore, if this format is used, Vault uses a certain capacity value to avoid reaching the full capacity of the tape. The two possible values are

  - 2 indicating a 2.2 gigabyte capacity. The actual maximum capacity is 2.0 gigabytes (2,147,483,648 bytes).
  - 5 indicating a 5 gigabyte capacity. The actual maximum capacity is 4.5 gigabytes (4,831,838,208 bytes).

For more information, refer to “Setting the Capacity Value on Exabyte Tapes” on page 5-6.

Please note: Commands issued to the Exabyte unit are handled by the Solaris or SunOS device driver for the Exabyte unit. Vault code does not communicate directly with the Exabyte tape unit.

Labeling Exabyte or DAT Tapes

When you label a tape, determine which type of recording method is required.

- ANSI: three tape marks
- Exabyte or 4mm DAT: one tape mark
- Exabyte or 4mm DAT: no tape marks

You label a tape with the `tapenlabel` utility. The format of the `tapenlabel` command appears below.

```
tapenlabel edm_tape_name   tape_label   tape_format
```

where:

edm_tape_name is the same as tapedevice. This can be TAPE1, TAPE2, TAPE3, or TAPE4. There is no default.

tape_label_name is the same as volumeserialnumber. It can be six or less alphanumeric characters. There is no default.

tape_format is the number of real tape marks per file. It can be 0, 1, or 3. The default is 1.

Example: `tapenlabel TAPE1 ibu001 0`

After you perform the `tapenlabel` command, Vault has the information it needs to correctly copy information to a tape on the tape device you specified.

Once labeled, Vault can handle the content of each tape format and process accordingly when positioning (file skipping) or reading files.

Please note: Use the `tapelabel` utility for tape media other than Exabyte or 4mm DAT.

Restrictions and Limitations

Consider the following when using Exabyte and 4mm DAT tape drives:

- Tapes that are archived on Windows-NT systems cannot be read on Unix Systems.
- Tapes with no tape marks cannot be appended to using the Vault append feature. If you try to append to a tape written with zero (0) physical file marks, an error is returned indicating that this option is not supported.
- After you insert an Exabyte or 4mm DAT tape into the drive, wait for the green indicator light before you execute the Vault tape command.
- Do not use Exabyte or 4mm DAT tape drives when unloading data with the `UNLOAD` command to create FUTIL-format tapes.

The following restrictions apply only to Exabyte tapes:

- Exabyte tapes written with no tape marks are never filled to capacity. Vault stops the operation before full capacity is reached.
- If the physical end-of-tape (PEOT) is reached before the capacity value has been attained, Vault terminates the tape operation with a fatal input/output error of the following standard Vault format:

  ```
  Processing not done - fatal return code from &1.
  RC = 30448  EC = 0  Please notify your EDM administrator.
  ```

  The fatal return code is from some low-level tape routine. The RC value that is returned indicates a premature physical end-of-tape failure. It is assumed that this condition is caused by one of the following user errors.

  - You specified a 5 gigabyte drive when you were using a 2.2 gigabyte drive.
  - You put a short (for example, less than 2.2 gigabyte capacity) tape in the drive.

Setting the Capacity Value on Exabyte Tapes

The information in this section applies only to Exabyte tapes.

Should the physical end-of-tape be reached, the Exabyte drive does not provide a mechanism for properly closing the tape. Because of this limitation, if Exabyte without real tape marks is the format being used, Vault uses a capacity value to avoid reaching the full capacity of the tape. The two possible values are

- 2 indicating a 2.2 gigabyte capacity. The actual maximum capacity is 2.0 gigabytes (2,147,483,648 bytes).
- 5 indicating a 5 gigabyte capacity. The actual maximum capacity is 4.5 gigabytes (4,831,838,208 bytes).

If you are using a 2.2 gigabyte drive, you do not need to specify a capacity value because the default capacity value of 2 indicates a 2.2 gigabyte drive.

If you are using a 5 gigabyte drive (and you want to put more than 2 gigabytes on it), you must set a UNIX environment variable, ETAPESIZE, before you invoke the `IBKUP` or `ARCHIVE` command.

Please note: Specify the high density drive.

The ETAPESIZE environment variable identifies the capacity of the Exabyte tape drive. The allowable values of the environment variable are:

- 2 indicating a 2.2 gigabyte capacity
- 5 indicating a 5 gigabyte capacity

The ETAPESIZE environment variable has a default value of **2** under the following conditions.

- The environment variable is not set
- The environment variable does not contain a valid value.

There are no messages to inform you of the value of ETAPESIZE.

Use the `setenv` command of UNIX operating system level to set the ETAPESIZE variable.

```
setenv ETAPESIZE {2, 5}
```

Specify 2 or 5 as the value.

Chapter 6

Setting Up a Rulebase

This chapter provides an information and instructions for using the Custom Part Facility to implement a user-defined rulebase that defines the files included in a part. This is an optional activity that allows you to implement multiple definitions of parts.

The Optegra Interface for CADDS 5i allows you to store objects as individual part definitions and to store standalone files that are not included in any part definition. To change the definition of a CADDS part, use the Flexible CADDS Part Definition Facility.

- Overview of the Custom Part Facility
- STORE Command
- GET/READ Commands
- REPLACE/UPDATE Commands
- EPD.Connect Support
- LIST Command
- CREATE/EXTRACT Commands
- LOCK/UNLOCK Commands

Overview of the Custom Part Facility

With the Custom Part Facility, you can implement user-defined rulebases that allow Vault to recognize different groups of files as parts.

Part

A Vault part is a collection of files known as a unit to an application, such as CADDS 5i or MEDUSA. An operation performed on a part is an all-or-nothing transaction. For example, changing the status code of a part changes the status code of all files that are members of the part. Executing the GET command for a part transfers all files belonging to that part. If any one of the files cannot be transferred, no files are transferred.

Rulebase

A rulebase is an executable that enforces a set of rules. These rules define the files of a part for a specific application and how the files should be handled. You can implement a rulebase with an operating system command language, such as the UNIX shell commands, or compiled C source code. Rulebases are invoked by the STORE, READ, GET, UPDATE, and REPLACE commands.

With the Custom Part Facility, you can write your own rulebase. A user-defined rulebase can only support parts. Your rulebase cannot have knowledge of a MEDUSA catalog, for instance, or even an individual MEDUSA file, unless it is a member of a MEDUSA part.

Vault has always included two rulebases:

- CADDS
- Local Files

The CADDS and LOCAL rulebases are not limited to handling only parts. When you execute the STORE command, the value specified for the Environment parameter determines which rulebase Vault uses. The CADDS rulebase is the only rulebase that can handle CADDS parts. When you define your own rulebase, it handles only parts, but they are not considered to be CADDS parts.

When you perform a transfer command (GET, READ, UPDATE, or REPLACE) on a standalone file, Vault assumes it is of type LOCAL if it has no file type. If it has a file type, Vault assumes it is of type CADDS. Based upon the assumed type, Vault can call either the local or CADDS rulebase, not the rulebase that originally stored the file.

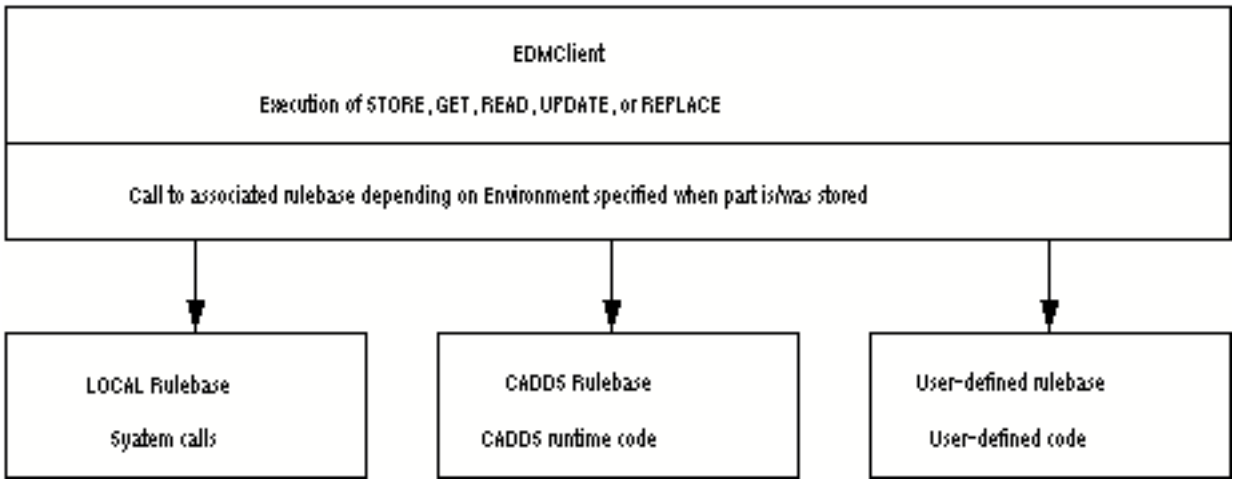
If your custom rulebase always stores parts instead of single files, Vault can have the assigned part object type to use in determining the rulebase to call on a GET, READ, UPDATE, or REPLACE command.

Please note: When performing any Vault transfer command (GET, READ, UPDATE, or REPLACE) make sure that the first Vault in the pm.config file is up. If it is down, swap the Vault entries so that the first Vault in the pm.config file is up. If this is not done you get an error message as follows:

```
Error - xxxxxxxx calling <rulebase>
```

<rulebase> = CADDS, PTCASM, MEDUSA, CATIA, or STEP

Figure 6-1 Environment Parameter Determines Which Rulebase Vault Calls



User Interface

When you implement a user-defined rulebase, the only change that end-users see is that there is a new value they can specify for the Environment parameter when they execute the STORE command. The value specified for the Environment parameter determines which rulebase Vault uses when users execute the STORE command. You need to tell them what this value is and when they should use it.

Application of a Rulebase

The environment value you assign when you store a part always remains the same. Once stored, you cannot change a part's environment value. Each subsequent execution of the UPDATE, REPLACE, GET or READ command uses the rulebase associated with the environment that was specified when the part was stored.

With a user-defined rulebase, when you execute the UPDATE or REPLACE command, you can add or delete part member files. There might be certain situations where rulebase should check some things before allowing files to be added or deleted. For example, the CADDS rulebase checks to make sure that the pd file is present before adding or deleting files.

Selecting a Rulebase with Vault

The environment value you assign when you store a part is used by the client to map to the appropriate rulebase. The information stored in the EDM.DEFAULTS file with the syntax

```
ENV(rulebase)=executable
```

allows the client to locate the rulebase executable.

The rulebase itself can choose to assign any object class(es) to the parts that it stores. Beginning with Optegra 1.1, the object class stored in Vault for a part does not have to be the same as the name of the rulebase used to store it.

When the part is retrieved from Vault on a GET or READ, or when it is replaced in Vault with an UPDATE or REPLACE command, the stored object class is used to determine which rulebase to use to process the object. The information in the EDM.DEFAULTS file stored with the syntax

```
OBJECT-CLASS(object_class)=rulebase
```

maps the stored object class back to the rulebase name. Once an object-class has been assigned to a part during the initial STORE of the part, it cannot be changed later. Your custom rulebase should store parts, not single standalone files. Vault associates an object type with all parts, but not with files.

Implementing a User-defined Rulebase

Perform the following steps to implement a user-defined rulebase.

1. Define the rules to be included in the rulebase, including
   a. The members of the part.
   b. The directories in which the members reside.
   c. The name mapping for the member file in Vault. That is, how are the local file names mapped to the Vault object names.

2. Create an executable that
   a. Implements the rules for determining the content of a part
   b. Reads the rulebase input files produced by Vault
   c. Produces the rulebase output file sent to Vault

   The content of the rulebase input and output files varies, depending on the command being executed and whether the rulebase is being invoked for initial or cleanup processing. The content of these files is described later in this chapter.
3. Add a line to the EDM.DEFAULTS file that defines the name of your rulebase and the name of the executable. The name of the rulebase is the value users can enter for the ENVIRONMENT keyword when storing parts. For example, with the line below in the EDM.DEFAULTS file, users can enter JET as the environment value for the STORE command.

   ```
   JET=/$EPD_HOME/bin/jet_rulebase
   ```

   The rules defined in /$EPD_HOME/bin/jet_rulebase would then be used for the part being stored.
4. Inform users about the new value for the ENVIRONMENT parameter and when to specify it for the STORE command.

Please note: When using the Custom Part Facility to define a rulebase for parts, the names of any file you store are translated to uppercase. Setting EDMCASE in EDM.DEFAULTS file has no effect. To retrieve the files, you must identify them with uppercase names.

Description of Programmatic Interface

Vault invokes a user-defined rulebase twice for each transaction.

- Vault invokes your rulebase after a command is issued before processing begins. For the STORE, UPDATE, and REPLACE commands, this initial call verifies that the member files of the part exist. For the GET and READ commands, the initial call prepares the user's workspace for the member files.
- After processing the command or sending an error message, Vault invokes your rulebase second time. This cleanup call performs the required cleanup steps. For example, you might specify in your rulebase that local files should be deleted when signout equals No for the STORE command.

The rulebase should exit with a value of zero only. Any error detected by the rulebase should be included in the ERROR-TEXT, argument so that the user executing the commands receives a meaningful message. Vault cannot correctly interpret errors returned by the program exit.

Invoking the Rulebase Upon Execution of Vault Commands

When you execute the `STORE, GET, READ, UPDATE`, or `REPLACE` commands, Vault determines the path name of the executable associated with the rulebase name. The rulebase name is the value that was specified for the Environment parameter when the file or part was stored. The rulebase name and its full path name are defined in the `EDM.DEFAULTS` file.

Vault uses input and output files to communicate with your rulebase. After you issue one of the above mentioned commands, Vault creates an input file using data from the command input and the relational database. The rulebase creates the output file using data from the input file, the local file system, and anywhere else where it has access.

When Vault invokes the rulebase the first time for a transaction, the command line input to the rulebase has the following format.

```
command call_flag input_file_name output_file_name
```

where:

command—Is the name of the command being processed. The value can be `store, get, read, update,` or `replace`.

call_flag —Is a flag indicating whether this is a call for initial processing or cleanup processing. The value is always initial when Vault invokes your rulebase the first time for a transaction. The value is cleanup when Vault invokes the rulebase the second time.

input_file_name —Is the name of the rulebase input file. Vault determines this name.

output_file_name —Is the name of the rulebase output file. Vault determines this name.

The format of the input and output files is the same, no matter which rulebase is called. The content of the input and output files varies slightly, depending on the command being executed.

For example, the arguments in an initial system call to your rulebase might look like this:

```
store initial STRII.0XREF STRIO.0XREF
```

Your rulebase processes the input and creates an output file that Vault uses to continue processing the command. If any errors occurred during the initial call to the user-defined rulebase, the rulebase would include them in the output file. Vault then writes these errors in the audit file.

Additions and Changes to Rulebase Communication Files for Optegra 1.1

From Optegra 2.0, the content of the communication files sent from the Optegra Vault client to the rulebase has expanded to include the values for most of the command line parameters and values for important system attributes of the parts and files.

Knowledge of the command line parameter values allows developers of rulebases to know more about the files and parts being created and updated in the vault and to affect the outcome of the command. For example, there can always be a user-attribute file name assigned for `Store`, `Update`, and `Replace`, even if the user has not provided one, so that the rulebase can add to or create attribute definitions to send to Vault. When no value is assigned to a command line parameter either by the user or through defaults, the command parameter has the value '$$NULL$$'. The command line values are provided to the rulebase as information. The rulebase cannot change command line values.

One system attribute that has changed is the value for `OBJECT-CLASS`. Prior to Optegra 1.1, `OBJECT-CLASS` was identical to the name of the rulebase. Starting with Optegra 1.1, the Store rulebase can assign any object class to parts and when the part is later retrieved, the object class is mapped back to the appropriate rulebase using values in the `EDM.DEFAULTS` file.

A new system attribute available with Optegra 1.1 onwards. By knowing the revision code assigned in Vault, a rulebase can, for example, determine appropriate local file names when retrieving several revisions of the same object. When an object system attribute has a blank value, the parameter is not sent to the rulebase. Hence, if an object is assigned to a project with revision sequence of NONE, the object revision is not sent to the rulebase.

The content of the rulebase 'communication files' for initial and cleanup calls, for client to rulebase and rulebase back to client, and for each of the five rulebase commands are described below. Sample rulebase communication files are also provided.

Content of Initial Call Input File

The content of the input file varies somewhat, depending on the command being executed. The following items are included in all input files. Additional items that appear for a particular command are described later in this chapter.

- OBJECT-CLASS —Is the name of your user-defined rulebase. This is the value you specify for the Environment parameter when storing a file or part.

  OBJECT-CLASS —Identifies the application that created the object. After you issue the STORE command, Vault stores the value for OBJECT-CLASS in the part directory.

  Vault provides two rulebases that are identified by these object classes:

  a. LOCAL—Identifies the operating system rulebase. On UNIX systems, LOCAL means that UNIX rules apply.

  b. CADDS —Identifies the CADDS rulebase. CADDS rules apply, regardless of the operating system you are using.

- OBJECT-NAME—Is what the files and parts are called in Vault. For the STORE command, OBJECT-NAME is the selection name if specified in the command input, otherwise it is the local file name. For the GET, READ, UPDATE, and REPLACE commands, OBJECT-NAME is always the selection name.
- OBJECT-TYPE—Indicates whether the object to be operated on is a part or a file that is a member of a part or a standalone file. The value can be PART, MEMBER, or FILE.

  When using the CADDS rulebase, the object type can also be CATALOG. When using the LOCAL rulebase, the object type can also be DIRECTORY.
- LOCAL-FILENAME—Is the full name of the file or part, including the node and path where the file resides.

  If the object type is FILE, the command must be adding the file to a part.

Content of Initial Call Output File

For each object, the output file can contain any of the following entries, depending on whether the transaction is for a part, part-member, or standalone file. The possible entries in the output file are the same for each command (STORE, GET, READ, UPDATE, and REPLACE).

- OBJECT-NAME — Is the name the file can have when stored in Vault.
- OBJECT-TYPE — Indicates whether the object is a part, a member of a part, or a standalone file. The value can be PART, MEMBER, or FILE.

- OBJECT-STATUS — Is a flag indicating whether or not the file is available.

  The value can be AVAILABLE, meaning no problems exist with this object, or UNAVAILABLE, meaning a problem does exist. If a problem exists, the error text provides an explanation. For parts, if any member is not available, the part is not available.

- ERROR-TEXT—Is the text of the error (if there was one) explaining why the file is unavailable. Vault uses ERROR-TEXT to construct a message in this form:

  ```
  CDMxxx950E Error from rulebase. Error text is “The file was not found.”
  ```

- LOCAL-FILENAME—Is the full name of the file including the node and path where the file resides.

Having defined your own rulebase, you are no longer handling the CADDS parts. The content of the output file must list the part first and indicate whether or not the part is available. After the part is identified, each member must be listed in the output file.

Examples of Input and Output Files

The format of the input and output files is the same, regardless of which rulebase is invoked. The CADDS rulebase is used for example purposes only. You cannot modify the CADDS rulebase.

The CADDS rulebase maps CADDS file names to Vault file names. Your rulebase must include rules that perform this function.

Examples of input and output files for the STORE command using the CADDS rulebase appear below. A selection name and a local file name has been specified.

```
# STORE of a CADDS part example
# Initial call, rulebase input file
OBJECT-CLASS=CADDS 5
OBJECT-NAME=CARLSTEST.PARTS.500
OBJECT-TYPE=PART
LOCAL-FILENAME=use4
# Initial call, rulebase output file
# Notes: the OBJECT-STATUS is AVAILABLE or UNAVAILABLE.
# If UNAVAILABLE then an attribute called
# ERROR-TEXT will have the error message.
# If any member file is UNAVAILABLE then the PART
# must also be UNAVAILABLE.
OBJECT-NAME=CARLSTEST.PARTS.500
OBJECT-TYPE=PART
OBJECT-STATUS=AVAILABLE
```

```
LOCAL-FILENAME=extra:/users/ccombs/parts/use4
OBJECT-NAME=CARLSTEST.PARTS.500.&PICT.03
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/use4/_pict/03
CGOS-FILETYPE=x‘00’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-USERATTRIBUTES=x‘00000000’
CGOS-UPDATE-DATETIME=x‘c2c85141’
CGOS-CHECKSUM=x‘0000’
OBJECT-NAME=CARLSTEST.PARTS.500.B
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABL
LOCAL-FILENAME=extra:/users/ccombs/parts/use4/b
CGOS-FILETYPE=x‘21’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-USERATTRIBUTES=x‘00000000’
CGOS-UPDATE-DATETIME=x‘c2c85140’
CGOS-CHECKSUM=x’0000’
OBJECT-NAME=CARLSTEST.PARTS.500.&PD
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/use4/_pd
CGOS-FILETYPE=x‘20’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-USERATTRIBUTES=x‘00000000’
CGOS-UPDATE-DATETIME=x‘c2c85140’
CGOS-CHECKSUM=x‘0000’
```

Invoking the Rulebase Upon Completion of Vault Command Processing:

When the Vault command is processed, it invokes your rulebase for the second time. This call indicates that any cleanup required by your rulebase should be done now. The cleanup invocation includes the following arguments along with the name of the command being processed. The command can be `STORE, GET, READ, UPDATE,` or `REPLACE`.

- A flag indicating a cleanup call. The value is `cleanup`.
- The name of the input file. Vault determines this name. This input file contains the name of each file involved in the transfer and the status of the file at the end of processing.
- The name of the output file. Vault determines this name. You can ignore the presence of this argument.

For example, the arguments in a cleanup call to your rulebase might look like this.

```
store cleanup STRCI.0XREF STRCO.0XREF
```

Content of Cleanup Call Input File

The input file in the cleanup call contains the same information, regardless of which rulebase is being called and which command is being processed. It includes the data listed below for each object involved in the transaction.

- OBJECT-NAME—Is the name the file or part has when stored in Vault.
- OBJECT-TYPE—Indicates whether the object is a standalone file, a part, or a member of a part. The value can be FILE, PART, or MEMBER.
- LOCAL-FILENAME—Is the full local name of the file or part, including the node and path where the file resides.
- OBJECT-STATUS—Is a flag indicating if the transaction was completed for the file or part. The value can be transferred or not transferred. For parts, if any member cannot be transferred, the part is not transferred.

Example of Cleanup Call Input File: The rulebase input file included in the cleanup call following a successful STORE command using the CADDS rulebase appears below. The cleanup call does not generate an output file.

The CADDS rulebase is used for example purposes only. You cannot modify the CADDS rulebase.

```
# Cleanup call, rulebase input file
# Notes: the OBJECT-STATUS is either TRANSFERRED or
# NOT-TRANSFERRED. The intent is if a file is
# TRANSFERRED then the file should be deleted.
# If a member file is NOT-TRANSFERRED then the part is also
# NOT-TRANSFERRED.
OBJECT-CLASS=CADDS 5
OBJECT-NAME=CARLSTEST.PARTS.500
OBJECT-TYPE=PART
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/use4
OBJECT-NAME=CARLSTEST.PARTS.500.&PD
OBJECT-TYPE=MEMBER
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/use4/_pd
OBJECT-NAME=CARLSTEST.PARTS.500.B
OBJECT-TYPE=MEMBER
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/use4/b
OBJECT-NAME=CARLSTEST.PARTS.500.&PICT.03
OBJECT-TYPE=MEMBER
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/use4/_pict/03
```

STORE Command

When users execute the STORE command, a user-defined rulebase must determine:

- The part type to assign to parts transferred.
- The naming convention of files involved in the transfer.
- Where the files are located locally.
- Attribute data needed to store the files.
- Whether or not the files are available to transfer.

After successful completion of the command, your rulebase can do any cleanup that you specify. For example, you can define whether or not the local copies should be deleted.

When a user executes the STORE command and specifies a user-defined rulebase for the Environment parameter, the selection scope must be:

- File
- Part
- List

If the selection scope is File, the file must be being added to a part.

If the selection scope is List, Vault treats each entry in the list as a single transaction. That is, Vault invokes your rulebase twice (an initial call and a cleanup call) for each list item.

Initial Call Input File Content

For the STORE command, the input file contains the following information.

- OBJECT-CLASS is the name of the user-defined rulebase. This is the value users specify for the Environment parameter.
- OBJECT-NAME is the Vault selection name if present, otherwise the Vault local file name.
- OBJECT-TYPE indicates whether the object is a part or a standalone file. The value can be PART or FILE.
- LOCAL-FILENAME is the local file name, including the node name.

Initial Call Input File

The STORE client sends the rulebase the following parameters in the order shown.

Table 6-1 Parameters STORE Client Initially Sends to Rulebase

PARAMETER	DEFINITION
COMMAND-ENV	Name of rulebase to invoke to process STORE. Executable is found by searching EDM.DEFAULTS for 'ENV(rulebase)=fully-qualified_executable_name' Command line value, selection list value (if using List selscope), or default 'ENV=rulebase' from EDM.DEFAULTS. There is no longer a system default of 'LOCAL'. For consistency with releases prior to Optegra 1.1, specify ENV=LOCAL in your EDM.DEFAULTS file.
COMMAND-SELSCOPE	Command line value, PART or FILE (if using List selscope), or from EDM.DEFAULTS 'SELSCOPE=value. The EDM.DEFAULTS file entry 'SELSCOPE(rulebase)=F,P,..' provides the list of selscopes that are permitted. There is no longer a system default of 'FILE'. For consistency with releases prior to Optegra 1.1, specify SELSCOPE=F in your EDM.DEFAULTS file.
COMMAND-LFNAME	Command line value or selection list value (if using List selscope).
COMMAND-SELNAME	Command line value, selection list value (if using List selscope), or '$$NULL$$'.
COMMAND-REVISION	Command line value, selection list value (if using List selscope), or '$$NULL$$'.
COMMAND-CLASS	Command line value or 'PUB'.
COMMAND-PROJID	Command line value or '$$NULL$$'.
COMMAND-STATCD	Command line value or '$$NULL$$'.
COMMAND-FILETYPE	Command line value or '$$NULL$$'.
COMMAND-PFNAME	Command line value or '$$NULL$$'.
COMMAND-SIGNOUT	Command line value or 'N'.
COMMAND-ATTRFILE	Command line value or assigned by Optegra.
COMMAND-SYSTYPE	Command line value or '$$NULL$$'.
COMMAND-USERTYPE	Command line value or '$$NULL$$'.
COMMAND-DESC	Command line value or '$$NULL$$'.
COMMAND-PARTNUM	Command line value or '$$NULL$$'.
COMMAND-GTCODE	Command line value or '$$NULL$$'.
CLIENT-RELEASE	Optegra release number.
SIGNON-USERID	Optegra ID of user issuing the command.
OBJECT-CLASS	Same value as 'COMMAND-ENV' on the initial call.
OBJECT-NAME	Same value as 'COMMAND-SELNAME' if not '$$NULL$$'. Otherwise, same value as 'COMMAND-LFNAME'.
OBJECT-TYPE	'PART' or 'FILE'.
LOCAL-FILENAME	Same value as 'COMMAND-LFNAME' on the initial call.

Initial Call Output File Content

The format and content of the initial call output file is the same for the STORE, GET, READ, UPDATE, and REPLACE commands. It has been described earlier in this chapter.

Initial Call Output File

The rulebase can return multiple sets of the following parameters in response to one set of parameters on the initial rulebase call. Values for OBJECT-NAME, OBJECT-TYPE, and LOCAL-FILENAME are required for each set. The presence of OBJECT-CLASS or OBJECT-NAME signals the beginning of a new entry in a repeating set. All other parameters are optional, can be presented in any order, and should be omitted if no value is available.

Table 6-2 Parameters Rulebase Initially Returns to STORE Client

PARAMETER	DEFINITION
OBJECT-CLASS	Optional, used for parts only. Object class to associate with parts in the vault. Rulebase can assign any value to store in database when OBJECT-TYPE is 'PART'. If the rulebase does not return a value for OBJECT-CLASS, the 'COMMAND-ENV' is used.
OBJECT-NAME	Required: Name of part or file to store in the vault. The rulebase must return an OBJECT-NAME for each part and file to store in the database. Usually, the 'OBJECT-NAME' sent forms the basis for the vault name.
OBJECT-TYPE	Required: PART, MEMBER (of part), or (standalone) FILE
LOCAL-FILENAME	Required: Full path name of the file including the node and path where the file resides.
OBJECT-REVISION	Optional: Rulebase can assign a revision code by returning it during the initial rulebase call.
OBJECT-PARENT	Optional, name of parent part when command used in the mode 'add member file to existing part'. Parent name is derived from PFNAME command parameter.
CGOS-FILETYPE	Optional for member files. Used only by CADDS rulebase.
CGOS-PROTECTIONGROUP	Optional for member files. Used only by CADDS rulebase.
CGOS-USERATTRIBUTES	Optional for member files. Used only by CADDS rulebase.
CGOS-UPDATE-DATETIME	Optional for member files. Used only by CADDS rulebase.
CGOS-CHECKSUM	Optional for member files. Used only by CADDS rulebase.
OBJECT-STATUS	Required: AVAILABLE or UNAVAILABLE. If 'unavailable' the rulebase should also return a MESSAGE-TEXT field.
MESSAGE-TEXT	Optional: Severity code and text of error, if any, explaining why the file is unavailable or any informational message to display with the part or file.

Cleanup Call Input File

A separate file is used for each part or standalone file in the transaction. All command parameters, COMMAND-ENV through SIGNON-USERID, are repeated. The following table shows other parameters that the client sends to the rulebase.

Table 6-3 Parameters STORE Client Sends to Rulebase in Cleanup

PARAMETER	DEFINITION
OBJECT-CLASS	As returned from rulebase on initial call or as assigned by client on initial call.
OBJECT-NAME	As returned from rulebase on initial call.
OBJECT-TYPE	PART, MEMBER, or FILE.
LOCAL-FILENAME	As returned from rulebase on initial call.
OBJECT-REVISION	As returned from rulebase on initial call or from the command line input, selection list input, or default assigned by the vault. No value sent if value is 'blank'.
OBJECT-STATUS	TRANSFERRED or NOT-TRANSFERRED

Cleanup Call Output File

A final output file from the rulebase to the client is optional. If used, it can contain only the following parameter.

Table 6-4 Parameter Rulebase Sends to STORE Client in Cleanup

PARAMETER	DEFINITION
FINAL-MESSAGE	Optional message to be used as the final message for the selection.

STORE Command Examples

Example 1

This is a simple case of a file stored by the LOCAL rulebase. The file is signed out in the database and not erased locally.

```
cistore env=local selscope=f lfname=alocalfile signout=y
```

STORE Command Initial Call Client to Rulebase:

```
COMMAND-ENV=LOCAL
COMMAND-SELSCOPE=FILE
COMMAND-LFNAME=alocalfile
COMMAND-SELNAME=$$NULL$$
COMMAND-REVISION=$$NULL$$
COMMAND-CLASS=PUB
COMMAND-PROJID=$$NULL$$
COMMAND-STATCD=$$NULL$$
```

```
COMMAND-FILETYPE=$$NULL$$
COMMAND-PFNAME=$$NULL$$
COMMAND-SIGNOUT=Y
COMMAND-ATTRFILE=ATTRFILE20523
COMMAND-SYSTYPE=$$NULL$$
COMMAND-USERTYPE=$$NULL$$
COMMAND-DESC=$$NULL$$
COMMAND-PARTNUM=$$NULL$$
COMMAND-GTCODE=$$NULL$$
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=LOCAL
OBJECT-NAME=alocalfile
OBJECT-TYPE=FILE
LOCAL-FILENAME=alocalfile
```

STORE Command Initial Call Rulebase to Client:

```
OBJECT-CLASS=LOCAL
OBJECT-NAME=alocalfile
OBJECT-TYPE=FILE
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/alocalfile
```

STORE Command Cleanup Call Client to Rulebase:

```
COMMAND-ENV=LOCAL
COMMAND-SELSCOPE=FILE
COMMAND-LFNAME=alocalfile
COMMAND-SELNAME=$$NULL$$
COMMAND-REVISION=$$NULL$$
COMMAND-CLASS=PUB
COMMAND-PROJID=$$NULL$$
COMMAND-STATCD=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-PFNAME=$$NULL$$
COMMAND-SIGNOUT=Y
COMMAND-ATTRFILE=ATTRFILE20523
COMMAND-SYSTYPE=$$NULL$$
COMMAND-USERTYPE=$$NULL$$
COMMAND-DESC=$$NULL$$
COMMAND-PARTNUM=$$NULL$$
COMMAND-GTCODE=$$NULL$$
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=LOCAL
OBJECT-NAME=alocalfile
OBJECT-TYPE=FILE
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/alocalfile
OBJECT-REVISION=1
OBJECT-STATUS=TRANSFERRED
```

Example 2

This is a directory stored by the LOCAL rulebase. Numerous optional command parameters are provided.

Please note: A separate cleanup call is made to the rulebase for each file stored.

```
cistore env=local selscope=d lfname=localdir selname=Sample.Dir.
revision=3 -statcd=rl systype=asystype usertype=utypedef
partnum=012345 -gtcode=abcdefg filedesc=This_is_a_sample_dir
```

Store Command Initial Call Client to Rulebase:

```
COMMAND-ENV=LOCAL
COMMAND-SELSCOPE=DIRECTORY
COMMAND-LFNAME=localdir
COMMAND-SELNAME=Sample.Dir.
COMMAND-REVISION=3
COMMAND-CLASS=PUB
COMMAND-PROJID=$$NULL$$
COMMAND-STATCD=RL
COMMAND-FILETYPE=$$NULL$$
COMMAND-PFNAME=$$NULL$$
COMMAND-SIGNOUT=N
COMMAND-ATTRFILE=ATTRFILE21077
COMMAND-SYSTYPE=asystype
COMMAND-USERTYPE=utypedef
COMMAND-DESC=This_is_a_sample_dir
COMMAND-PARTNUM=012345
COMMAND-GTCODE=abcdefg
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=LOCAL
OBJECT-NAME=Sample.Dir.
OBJECT-TYPE=DIRECTORY
LOCAL-FILENAME=localdir
```

Store Command Initial Call Rulebase to Client:

```
OBJECT-CLASS=LOCAL
OBJECT-NAME=Sample.Dir.Sample.local.file1
OBJECT-TYPE=FILE
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/localdir/
Sample.local.file1
OBJECT-CLASS=LOCAL
OBJECT-NAME=Sample.Dir.Sample.local.file2
OBJECT-TYPE=FILE
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/localdir/
Sample.local.file2
```

Store Command Cleanup Call Client to Rulebase:

```
COMMAND-ENV=LOCAL
COMMAND-SELSCOPE=DIRECTORY
COMMAND-LFNAME=localdir
COMMAND-SELNAME=Sample.Dir.
COMMAND-REVISION=3
COMMAND-CLASS=PUB
COMMAND-PROJID=$$NULL$$
COMMAND-STATCD=RL
COMMAND-FILETYPE=$$NULL$$
COMMAND-PFNAME=$$NULL$$
COMMAND-SIGNOUT=N
COMMAND-ATTRFILE=ATTRFILE21077
COMMAND-SYSTYPE=asystype
COMMAND-USERTYPE=utypedef
COMMAND-DESC=This_is_a_sample_dir
COMMAND-PARTNUM=012345
COMMAND-GTCODE=abcdefg
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=LOCAL
OBJECT-NAME=Sample.Dir.Sample.local.file1
OBJECT-TYPE=FILE
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/localdir/
Sample.local.file1
OBJECT-REVISION=3
OBJECT-STATUS=TRANSFERRED
```

Example 3

This example shows a CADDS Part stored by the CADDS rulebase. The part is assigned to a project with no revision sequence, so the revision code is not given to the rulebase on the cleanup call.

```
cistore env=CADDS 5 selscope=p lfname=samplepart class=pro
projid=rccnorvs
```

STORE Command Initial Call Client to Rulebase:

```
COMMAND-ENV=CADDS 5
COMMAND-SELSCOPE=PART
COMMAND-LFNAME=samplepart
COMMAND-SELNAME=$$NULL$$
COMMAND-REVISION=$$NULL$$
COMMAND-CLASS=PRO
COMMAND-PROJID=RCCNORVS
COMMAND-STATCD=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-PFNAME=$$NULL$$
COMMAND-SIGNOUT=N
COMMAND-ATTRFILE=ATTRFILE21310
COMMAND-SYSTYPE=$$NULL$$
COMMAND-USERTYPE=$$NULL$$
```

```
COMMAND-DESC=$$NULL$$
COMMAND-PARTNUM=$$NULL$$
COMMAND-GTCODE=$$NULL$$
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=CADDS 5
OBJECT-NAME=samplepart
OBJECT-TYPE=PART
LOCAL-FILENAME=samplepart
```

STORE Command Initial Call Rulebase to Client:

```
OBJECT-CLASS=CADDS 5
OBJECT-NAME=SAMPLEPART
OBJECT-TYPE=PART
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/samplepart
OBJECT-STATUS=AVAILABLE
OBJECT-NAME=SAMPLEPART.DRAW1
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/samplepart/draw1
OBJECT-STATUS=AVAILABLE
CGOS-FILETYPE=x'21'
CGOS-PROTECTIONGROUP=x'0000'
CGOS-USERATTRIBUTES=x'19970000'
CGOS-UPDATE-DATETIME=x'C9651614'
CGOS-CHECKSUM=x'0000'
OBJECT-NAME=SAMPLEPART.&PD
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/samplepart/_pd
OBJECT-STATUS=AVAILABLE
CGOS-FILETYPE=x'20'
CGOS-PROTECTIONGROUP=x'0000'
CGOS-USERATTRIBUTES=x'19970000'
CGOS-UPDATE-DATETIME=x'C9651614'
CGOS-CHECKSUM=x'0000'
```

STORE Command Cleanup Call Client to Rulebase:

```
COMMAND-ENV=CADDS 5
COMMAND-SELSCOPE=PART
COMMAND-LFNAME=samplepart
COMMAND-SELNAME=$$NULL$$
COMMAND-REVISION=$$NULL$$
COMMAND-CLASS=PRO
COMMAND-PROJID=RCCNORVS
COMMAND-STATCD=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-PFNAME=$$NULL$$
COMMAND-SIGNOUT=N
COMMAND-ATTRFILE=ATTRFILE21310
COMMAND-SYSTYPE=$$NULL$$
COMMAND-USERTYPE=$$NULL$$
COMMAND-DESC=$$NULL$$
COMMAND-PARTNUM=$$NULL$$
```

```
COMMAND-GTCODE=$$NULL$$
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=CADDS 5
OBJECT-NAME=SAMPLEPART
OBJECT-TYPE=PART
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/samplepart
OBJECT-STATUS=TRANSFERRED
OBJECT-NAME=SAMPLEPART.&PD
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/samplepart/_pd
OBJECT-STATUS=TRANSFERRED
OBJECT-NAME=SAMPLEPART.DRAW1
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/samplepart/draw1
OBJECT-STATUS=TRANSFERRED
```

Example 4

This example uses the STORE command to add a file to the part just stored.

```
cistore env=CADDS 5 selscope=f lfname=analysis pfname=samplepart -
selname=samplepart.analysis
```

STORE Command Initial Call Client to Rulebase:

```
COMMAND-ENV=CADDS
COMMAND-SELSCOPE=FILE
COMMAND-LFNAME=analysis
COMMAND-SELNAME=samplepart.analysis
COMMAND-REVISION=$$NULL$$
COMMAND-CLASS=PUB
COMMAND-PROJID=$$NULL$$
COMMAND-STATCD=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-PFNAME=samplepart
COMMAND-SIGNOUT=N
COMMAND-ATTRFILE=ATTRFILE21526
COMMAND-SYSTYPE=$$NULL$$
COMMAND-USERTYPE=$$NULL$$
COMMAND-DESC=$$NULL$$
COMMAND-PARTNUM=$$NULL$$
COMMAND-GTCODE=$$NULL$$
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=CADDS 5
OBJECT-NAME=samplepart.analysis
OBJECT-TYPE=FILE
LOCAL-FILENAME=analysis
```

STORE Command Initial Call Rulebase to Client:

```
OBJECT-CLASS=CADDS 5
OBJECT-NAME=SAMPLEPART.ANALYSIS
```

```
OBJECT-TYPE=FILE
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/analysis
OBJECT-PARENT=SAMPLEPART
CGOS-FILETYPE=x'03'
CGOS-PROTECTIONGROUP=x'0000'
CGOS-USERATTRIBUTES=x'00000000'
CGOS-UPDATE-DATETIME=x'C9651F62'
CGOS-CHECKSUM=x'0000'
OBJECT-STATUS=AVAILABLE
```

Store Command Cleanup Call Client to Rulebase:

```
COMMAND-ENV=CADDS 5
COMMAND-SELSCOPE=FILE
COMMAND-LFNAME=analysis
COMMAND-SELNAME=samplepart.analysis
COMMAND-REVISION=$$NULL$$
COMMAND-CLASS=PUB
COMMAND-PROJID=$$NULL$$
COMMAND-STATCD=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-PFNAME=samplepart
COMMAND-SIGNOUT=N
COMMAND-ATTRFILE=ATTRFILE21526
COMMAND-SYSTYPE=$$NULL$$
COMMAND-USERTYPE=$$NULL$$
COMMAND-DESC=$$NULL$$
COMMAND-PARTNUM=$$NULL$$
COMMAND-GTCODE=$$NULL$$
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=CADDS 5
OBJECT-NAME=SAMPLEPART.ANALYSIS
OBJECT-TYPE=FILE
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/analysis
OBJECT-PARENT=SAMPLEPART
OBJECT-STATUS=TRANSFERRED
```

GET/READ Commands

When users execute the GET or READ commands, a user-defined rulebase must determine:

- The naming convention of the files involved in the transfer.
- Where the files can be located.

After successful completion of the command, your rulebase can do any cleanup that you specify.

When a user executes the GET or READ command and specifies a user-defined rulebase for the Environment parameter, the Vault selection scope must be:

- File
- File set
- Part
- List

If the selection scope for the command is List or File set, Vault invokes the rulebase twice (initial call and cleanup call) for each object in the list.

Initial Call Input File Content

For the GET or READ command, the input file contains the following information.

- OBJECT-CLASS—Is the name of your user-defined rulebase. This is the value specified for the Environment parameter when the object was stored.
- OBJECT-NAME—Is the Vault selection name.
- OBJECT-TYPE—Indicates whether the object is a standalone file, a part, or a member of a part. The value can be FILE, PART, or MEMBER.
- LOCAL-FILENAME—Is the local file name. Vault constructs this name from the selection name, local file name, and local directory input fields.
- OVER-WRITE—Indicates whether or not to overwrite an existing local file with the Vault file. The value can be YES or NO.
- PLACEHOLDER—Is a flag that indicates if the file is an actual data file or an empty placeholder. A value of YES indicates a placeholder; NO indicates an actual data file.

Initial Call Input File

The GET and READ clients send the rulebase the following command parameters in the order shown.

Please note: Certain parameters are used only for READ (these are marked). A separate rulebase call is made for each part or standalone file in the selection.

Table 6-5 Parameters GET and READ Clients Initially Send to Rulebase

Parameter	Definition
COMMAND-SELSCOPE	FILE/PA RT/BINDER/FILESET/LIST (or first letter).
COMMAND-SELNAME	Name of selection in vault, or selection list name.
COMMAND-LDIRNAME	Command line value or $$NULL$$.
COMMAND-LFNAME	Command line value or $$NULL$$.
COMMAND-REVISION	Command line value or $$NULL$$(READ command only).
COMMAND-FILETYPE	Command line value or $$NULL$$.
COMMAND-CATLEVEL	Command line value or $$NULL$$.
COMMAND-DATECRIT	Command line value or $$NULL$$.
COMMAND-DATE	Command line value or $$NULL$$.
COMMAND-OVERLAY	Command line value or 'N' (formerly OVER-WRITE).
COMMAND-CONT	Command line value or 'N' (READ command only).
COMMAND-ATTRFILE	Command line value or $$NULL$$.
COMMAND-VAULTID	Command line value or current system id.
CLIENT-RELEASE	Optegra release number.
SIGNON-USERID	Optegra ID of user issuing the command.
OBJECT-CLASS	Part object class as stored in vault or, if standalone file with a vault filetype, 'CADDS 5i', if standalone file with vault filetype blank, 'LOCAL'. Not repeated for member files.
OBJECT-NAME	Name of part or file as stored in the vault.
OBJECT-TYPE	FILE or PART or MEMBER.
LOCAL-FILENAME	based on command line dirname, lfname, and file_name.
PLACEHOLDER	YES (if only a reserved name in vault) NO (if the file is stored in vault).
OBJECT-REVISION	Revision of object in the vault, if not blank.

Initial Call Output File

The initial call output file that the rulebase returns to the client should contain one part, member file, or standalone file entry for each such entry in the file above. The beginning of data for a member file in a part is identified by the presence of the `OBJECT-NAME` parameter.

Table 6-6 Parameters Rulebase Initially Returns GET and READ Clients

Parameter	Definition
OBJECT-CLASS	Optional: identifies the part. Used only for parts.
OBJECT-NAME	Required: identifies the part or file. Signals the beginning of another object's data.
OBJECT-TYPE	Required: FILE/MEMBER/PART identifies the part or file.
LOCAL-FILENAME	Required: Provides the fully-qualified name of the object as it can be stored locally. [node:/rootdir/../partdir/filename]
OBJECT-REVISION	Optional: identifies the part or file when more than one revision of the same part or file is obtained.
OBJECT-STATUS	Required: AVAILABLE or UNAVAILABLE. If 'unavailable' the rulebase should also return a MESSAGE-TEXT field.
MESSAGE-TEXT	Optional: Severity code and text of error if any explaining why the file is unavailable or any informational message to display with the part or file.

Cleanup Call Input File

In the cleanup call input file that the client sends to the rulebase, the command parameters from `COMMAND-SELSCOPE` to `SIGNON-USERID` are repeated. These parameters are followed by the repeating parameters in the next table:

Table 6-7 Repeating Parameters GET and READ Clients Sent to Rulebase in Cleanup

Parameter	Definition
OBJECT-CLASS	Part object class as stored in vault or, if standalone file with a vault filetype, 'CADDS 5i', if standalone file with vault filetype blank, 'LOCAL'.
OBJECT-NAME	Name of part or file as stored in the vault.
OBJECT-TYPE	FILE or PART or MEMBER.
LOCAL-FILENAME	Local filename as returned by rulebase on initial call.
PLACEHOLDER	YES (if only a reserved name in vault) NO (if the file is stored in vault) Placeholders are not transferred.
OBJECT-REVISION	Revision of object in the vault, if not blank.
OBJECT-PARENT	If file is a part member but part not transferred, then Object-parent identifies the part name. Otherwise, the parameter is omitted.
OBJECT-SIGNOUT	'Y' If object signed out in vault. 'N' If object obtained only for Read. If you GET a Binder or List, some of the members can be obtained for Read only and others Signed out.
CGOS-FILETYPE	Used only by CADDS rulebase.

Table 6-7 Repeating Parameters GET and READ Clients Sent to Rulebase in Cleanup

Parameter	Definition
CGOS-PROTECTIONGROUP	Used only by CADDS rulebase.
CGOS-USERATTRIBUTES	Used only by CADDS rulebase.
CGOS-UPDATE-DATETIME	Used only by CADDS rulebase.
CGOS-CHECKSUM	Used only by CADDS rulebase.
OBJECT-STATUS	TRANSFERRED or NOT-TRANSFERRED

Cleanup Call Output File

A final output file from the rulebase to the client is optional. If used, it can contain only the following parameter.

Table 6-8 Parameter Rulebase Sends to GET and READ Clients in Cleanup

PARAMETER	DEFINITION
FINAL-MESSAGE	Optional message to be used as the final message for the selection.

Examples of Input and Output files for GET or READ

The following are examples of input and output files for the `GET` or `READ` commands.

Initial Call Input File

In the example below, the local directory and local file name input fields were left blank. Consequently, the local file name defaults to the object name.

```
# GET of a CADDS part example
# Initial call, rulebase input file
# Notes: OVER-WRITE is either YES or NO.
# Depends on user's input. The intent is to allow the user to
# overwrite existing files.
OBJECT-CLASS=CADDS 5
OBJECT-NAME=CARLSTEST.PARTS.500
OBJECT-TYPE=PART
LOCAL-FILENAME=CARLSTEST.PARTS.500
OVER-WRITE=NO
OBJECT-NAME=CARLSTEST.PARTS.500.&PICT.03
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=CARLSTEST.PARTS.500.&PICT.03
OBJECT-NAME=CARLSTEST.PARTS.500.B
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=CARLSTEST.PARTS.500.B
OBJECT-NAME=CARLSTEST.PARTS.500.&PD
```

```
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=CARLSTEST.PARTS.500.&PD
```

Initial Call Output File

In this example, the local file name fields are the actual file system file names.

```
# GET of a CADDS part example
# Initial call, rulebase output file
OBJECT-NAME=CARLSTEST.PARTS.500
OBJECT-TYPE=PART
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500
OBJECT-NAME=CARLSTEST.PARTS.500.&PD
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest /parts/500/_pd
OBJECT-NAME=CARLSTEST.PARTS.500.B
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500/b
OBJECT-NAME=CARLSTEST.PARTS.500.&PICT.03
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest
  /parts/500/_pict/03
```

Cleanup Call Input File:

```
# GET of a CADDS part example
# Cleanup call, rulebase input file
# Notes: PLACEHOLDER is either YES or NO.
# The intent is to indicate that the name was reserved.
OBJECT-CLASS=CADDS 5
OBJECT-NAME=CARLSTEST.PARTS.500
OBJECT-TYPE=PART
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500
PLACEHOLDER=NO
OBJECT-NAME=CARLSTEST.PARTS.500.&PICT.03
OBJECT-TYPE=MEMBER
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest
/parts/500/_pict/03
PLACEHOLDER=NO
CGOS-FILETYPE=x‘00’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-CHECKSUM=x‘0000’
CGOS-UPDATE-DATETIME=x‘c2c85141’CGOS-USERATTRIBUTES=x‘00000000’
OBJECT-NAME=CARLSTEST.PARTS.500.B
OBJECT-TYPE=MEMBER
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500/b
PLACEHOLDER=NO
CGOS-FILETYPE=x‘21’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-CHECKSUM=x‘0000’
CGOS-UPDATE-DATETIME=x‘c2c85140’
CGOS-USERATTRIBUTES=x‘00000000’
OBJECT-NAME=CARLSTEST.PARTS.500.&PD
OBJECT-TYPE=MEMBER
OBJECT-STATUS=TRANSFERED
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500/_pd
PLACEHOLDER=NO
CGOS-FILETYPE=x‘20’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-CHECKSUM=x’0000’
CGOS-UPDATE-DATETIME=x‘c2c85140’
CGOS-USERATTRIBUTES=x‘00000000’
```

The cleanup call does not generate an output file.

GET/READ Command Examples

Example 1

This example retrieves the local file stored in a previous example and assigns a directory and new local name.

```
ciread selscope=f selname=alocalfile ldirname=newdir
lfname=newlocalname - attrfile=getattribs
```

READ Command Initial Call Client to Rulebase:

```
COMMAND-SELSCOPE=F
COMMAND-SELNAME=alocalfile
COMMAND-LDIRNAME=newdir
COMMAND-LFNAME=newlocalname
COMMAND-REVISION=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-CATLEVEL=$$NULL$$
COMMAND-DATECRIT=$$NULL$$
COMMAND-DATE=$$NULL$$
COMMAND-OVERLAY=N
COMMAND-CONT=N
COMMAND-ATTRFILE=getattribs
COMMAND-VAULTID=BLACKDOG
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=LOCAL
OBJECT-NAME=alocalfile
OBJECT-TYPE=FILE
LOCAL-FILENAME=newdir/newlocalname
PLACEHOLDER=NO
OBJECT-REVISION=1
```

READ Command Initial Call Rulebase to Client:

```
OBJECT-CLASS=LOCAL
OBJECT-NAME=alocalfile
OBJECT-TYPE=FILE
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/newdir/newlocalname
OBJECT-REVISION=1
OBJECT-STATUS=AVAILABLE
```

READ Command Cleanup Call Client to Rulebase:

```
COMMAND-SELSCOPE=F
COMMAND-SELNAME=alocalfile
COMMAND-LDIRNAME=newdir
COMMAND-LFNAME=newlocalname
COMMAND-REVISION=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-CATLEVEL=$$NULL$$
COMMAND-DATECRIT=$$NULL$$
COMMAND-DATE=$$NULL$$
COMMAND-OVERLAY=N
COMMAND-CONT=N
COMMAND-ATTRFILE=getattribs
COMMAND-VAULTID=BLACKDOG
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=LOCAL
OBJECT-NAME=alocalfile
OBJECT-TYPE=FILE
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/newdir/
      newlocalname
PLACEHOLDER=NO
OBJECT-REVISION=1
OBJECT-SIGNOUT=N
OBJECT-STATUS=TRANSFERRED
```

Example 2

This example generates a Binder that contains the CADDS part. 'SAMPLE.BINDER.PART' as an aggregated member.

```
ciget selscope=b selname=sample.binder overlay=y
```

Get Command Initial Call Client to Cadds Rulebase:

```
COMMAND-SELSCOPE=B
COMMAND-SELNAME=sample.binder
COMMAND-LDIRNAME=$$NULL$$
COMMAND-LFNAME=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-CATLEVEL=$$NULL$$
COMMAND-DATECRIT=$$NULL$$
COMMAND-DATE=$$NULL$$
COMMAND-OVERLAY=Y
```

```
COMMAND-ATTRFILE=$$NULL$$
COMMAND-VAULTID=BLACKDOG
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=CADDS
OBJECT-NAME=SAMPLE.BINDER.PART
OBJECT-TYPE=PART
LOCAL-FILENAME=SAMPLE.BINDER.PART
PLACEHOLDER=NO
OBJECT-REVISION=2
OBJECT-NAME=SAMPLE.BINDER.PART.&PD
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=SAMPLE.BINDER.PART.&PD
PLACEHOLDER=NO
OBJECT-REVISION=2
OBJECT-NAME=SAMPLE.BINDER.PART.A
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=SAMPLE.BINDER.PART.A
PLACEHOLDER=NO
OBJECT-REVISION=2
```

GET Command Initial Call Rulebase to Client:

```
OBJECT-CLASS=CADDS 5
OBJECT-NAME=SAMPLE.BINDER.PART
OBJECT-TYPE=PART
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/part
OBJECT-STATUS=AVAILABLE
OBJECT-NAME=SAMPLE.BINDER.PART.A
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/
      part/a
OBJECT-STATUS=AVAILABLE
OBJECT-NAME=SAMPLE.BINDER.PART.&PD
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/
      part/_pd
OBJECT-STATUS=AVAILABLE
```

GET Command Cleanup Call Client to Rulebase: :

```
COMMAND-SELSCOPE=B
COMMAND-SELNAME=sample.binder
COMMAND-LDIRNAME=$$NULL$$
COMMAND-LFNAME=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-CATLEVEL=$$NULL$$
COMMAND-DATECRIT=$$NULL$$
COMMAND-DATE=$$NULL$$
COMMAND-OVERLAY=Y
COMMAND-ATTRFILE=$$NULL$$
COMMAND-VAULTID=BLACKDOG
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=CADDS 5
```

```
OBJECT-NAME=SAMPLE.BINDER.PART
OBJECT-TYPE=PART
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/part
PLACEHOLDER=NO
OBJECT-REVISION=2
OBJECT-SIGNOUT=N
OBJECT-STATUS=TRANSFERRED
OBJECT-NAME=SAMPLE.BINDER.PART.&PD
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/
      part/_pd
PLACEHOLDER=NO
OBJECT-REVISION=2
OBJECT-SIGNOUT=N
CGOS-FILETYPE=x'20'
CGOS-PROTECTIONGROUP=x'0000'
CGOS-USERATTRIBUTES=x'19970000'
CGOS-UPDATE-DATETIME=x'C96574F6'
CGOS-CHECKSUM=x'0000'
OBJECT-STATUS=TRANSFERRED
OBJECT-NAME=SAMPLE.BINDER.PART.A
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/
      sample/binder/part/a
PLACEHOLDER=NO
OBJECT-REVISION=2
OBJECT-SIGNOUT=N
CGOS-FILETYPE=x'21'
CGOS-PROTECTIONGROUP=x'0000'
CGOS-USERATTRIBUTES=x'19970000'
CGOS-UPDATE-DATETIME=x'C96574F6'
CGOS-CHECKSUM=x'0000'
OBJECT-STATUS=TRANSFERRED
```

REPLACE/UPDATE Commands

When users execute the REPLACE or UPDATE commands, a user-defined rulebase must determine

- The files involved in the transfer; that is, if any members have been added to or deleted from the part.
- The naming convention of the files.
- Where the files can be located.

With a user-defined rulebase, when you execute the UPDATE or REPLACE command, you can add or delete part member files. There might be certain situations where your rulebase should check something before allowing files to be added or deleted. For example, the CADDS rulebase checks to make sure that the pd file is present before adding or deleting files.

After successful completion of the command, your rulebase can do any cleanup that you specify.

When a user executes the UPDATE or REPLACE command and specifies a user-defined rulebase for the Environment parameter, the Vault selection scope must be:

- File
- File set
- Part
- List

When the selection scope for the command is List or File set, Vault invokes the rulebase twice (initial call and cleanup call) for each object in the list.

Initial Call Input File Content

For the REPLACE or UPDATE commands, the input file contains the following information.

- OBJECT-CLASS—Is the name of your rulebase. This is the value specified for the Environment parameter when the object was stored.
- OBJECT-NAME—Is the Vault selection name.
- OBJECT-TYPE—Indicates whether the object is a standalone file, a part, or a member of a part. The value can be FILE, PART, or MEMBER.

- LOCAL-FILENAME—Is the full name of the file including the node and path where the file resides. this is the name that was generated by the associated GET command.
- PLACEHOLDER—Is a flag that indicates if the file is an actual data file or an empty placeholder. A value of YES indicates a placeholder; NO indicates an actual data file.

Initial Call Output File Content

The format and content of the initial call output file is the same for the STORE, GET, READ, UPDATE, and REPLACE commands. It has been described earlier in this chapter.

Initial Call Input File

The UPDATE and REPLACE clients initially send the rulebase these command parameters in the order shown.

Table 6-9 Parameters the UPDATE and REPLACE Clients Initially Send the Rulebase

Parameter	Definition
COMMAND-SELSCOPE	FILE/PART/BINDER/FILESET/LIST (or first letter).
COMMAND-SELNAME	Name of selection in vault, or selection list name.
COMMAND-ATTRFILE	Command line value or assigned by Optegra.
COMMAND-VAULTID	Command line value or current system id.
CLIENT-RELEASE	Optegra release number.
SIGNON-USERID	Optegra ID of user issuing the command.

Table 6-10 Repeating Parameters the UPDATE and REPLACE Clients Send the Rulebase

Parameter	Definition
OBJECT-CLASS	Part object class as stored in vault or, if standalone file with a vault filetype, 'CADDS 5i', if standalone file with vault filetype blank, 'LOCAL'.
OBJECT-NAME	Name of part or file as stored in the vault.
OBJECT-TYPE	FILE or PART or MEMBER.
LOCAL-FILENAME	Fully-qualified name of object as assigned when the object was signed out from the vault.
PLACEHOLDER	YES (if only a reserved name in vault) NO (if the file is stored in vault).
OBJECT-REVISION	Revision of object in the vault, if not blank.
CGOS-FILETYPE	For files, filetype as stored in the vault. Used only for CADDS rulebase. Not sent if blank.

Initial Call Output File

The initial call output file that the rulebase returns to the client contains information for one part or standalone file. It can include additional member files for a part. The beginning of data for a part is signalled by the OBJECT-CLASS parameter. The beginning of data for a member file in a part is identified by the OBJECT-NAME parameter. The order of other parameters is optional.

Table 6-11 Parameters the Rulebase Initially Sends the UPDATE and REPLACE Clients

Parameter	Definition
OBJECT-CLASS	Optional: identifies the part. Rulebase cannot change value in the vault during UPDATE/REPLACE.
OBJECT-NAME	Required: identifies existing or new parts or files.
OBJECT-TYPE	Required: FILE/MEMBER/PART identifies the part or file.
LOCAL-FILENAME	Required: For new member files, the fully-qualified name of the object as it is stored locally. [node:/rootdir/../partdir/filename]
OBJECT-REVISION	Optional: identifies the part or file. Rulebase cannot change value in the vault during UPDATE/REPLACE.
CGOS-FILETYPE	Optional for member files. Used only by CADDS rulebase.
CGOS-PROTECTIONGROUP	Optional for member files. Used only by CADDS rulebase.
CGOS-USERATTRIBUTES	Optional for member files. Used only by CADDS rulebase.
CGOS-UPDATE-DATETIME	Optional for member files. Used only by CADDS rulebase.
CGOS-CHECKSUM	Optional for member files. Used only by CADDS rulebase.
OBJECT-STATUS	Required: AVAILABLE or UNAVAILABLE. If 'unavailable' the rulebase should also return a MESSAGE-TEXT field.
MESSAGE-TEXT	Optional: Severity code and text of error if any explaining why the file is unavailable or any informational message to display with the part or file.

Cleanup Call Input File

In the cleanup call input file that the client sends to the rulebase, command parameters from `COMMAND-SELSCOPE` to `SIGNON-USERID` are repeated. Those parameters are followed by the repeating parameters in the next table:

Table 6-12 Repeating Parameters UPDATE and REPLACE Clients Send to the Rulebase in Cleanup

Parameter	Definition
OBJECT-CLASS	Part object class as stored in vault or, if standalone file with a vault filetype, 'CADDS 5i', if standalone file with vault filetype blank, 'LOCAL'.
OBJECT-NAME	Name of part or file as stored in the vault.
OBJECT-TYPE	FILE or PART or MEMBER.
LOCAL-FILENAME	Fully-qualified name of object as assigned when the object was signed out from the vault.

Table 6-12 Repeating Parameters UPDATE and REPLACE Clients Send to the Rulebase in Cleanup

Parameter	Definition
PLACEHOLDER	YES (if only a reserved name in vault) NO (if the file is stored in vault).
OBJECT-REVISION	Revision of object in the vault, if not blank.
CGOS-FILETYPE	For files, filetype as stored in the vault. Used only for CADDS rulebase. Not sent if blank.
OBJECT-STATUS	TRANSFERRED or NOT-TRANSFERRED.

Cleanup Call Output File

A final output file from the rulebase to the client is optional. If used, it can contain only the following parameter.

Table 6-13 Parameter Rulebase Sends to UPDATE or REPLACE Client in Cleanup

Parameter	Definition
FINAL-MESSAGE	Optional message to be used as the final message for the selection.

Examples of Input and Output Files for REPLACE or UPDATE

The following are examples of input and output files for the REPLACE or UPDATE command. The CADDS rulebase is used for example purposes only. You cannot modify the CADDS rulebase.

Initial Call Input File

In the example below, the local file name fields are fully qualified. The local file name information is retrieved from the cross-reference information stored in Vault.

```
# REPLACE of a CADDS part example
# Initial call, rulebase input file
# Notes: Update is exactly like replace.
# Only difference is the cleanup is not called for update.
OBJECT-CLASS=CADDS 5
OBJECT-NAME=CARLSTEST.PARTS.500
OBJECT-TYPE=PART
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500
PLACEHOLDER=NO
OBJECT-NAME=CARLSTEST.PARTS.500.&PICT.03
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/
      500/_pict/03
PLACEHOLDER=NO
OBJECT-NAME=CARLSTEST.PARTS.500.B
OBJECT-TYPE=MEMBER
```

```
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500/b
PLACEHOLDER=NO
OBJECT-NAME=CARLSTEST.PARTS.500.&PD
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500/_pd
PLACEHOLDER=NO
```

Initial Call Output File

```
# REPLACE of a CADDS part example
# Initial call, rulebase output file
OBJECT-NAME=CARLSTEST.PARTS.500
OBJECT-TYPE=PART
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500
OBJECT-NAME=CARLSTEST.PARTS.500.&PICT.03
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500
/_pict/03
CGOS-FILETYPE=x‘00’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-USERATTRIBUTES=x‘00000000’
CGOS-UPDATE-DATETIME=x‘c2c85141’
CGOS-CHECKSUM=x‘0000
OBJECT-NAME=CARLSTEST.PARTS.500.&PD
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500/_pd
CGOS-FILETYPE=x‘20’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-USERATTRIBUTES=x‘00000000’
CGOS-UPDATE-DATETIME=x‘c2c85140’
CGOS-CHECKSUM=x‘0000’
OBJECT-NAME=CARLSTEST.PARTS.500.B
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/ccombs/parts/arlstest/parts/500/b
CGOS-FILETYPE=x‘21’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-USERATTRIBUTES=x‘00000000’
CGOS-UPDATE-DATETIME=x‘c2c85140’
CGOS-CHECKSUM=x‘0000’
```

Cleanup Call Input File

```
# REPLACE of a CADDS part example
# Cleanup call, rulebase output file
OBJECT-CLASS=CADDS 5
OBJECT-NAME=CARLSTEST.PARTS.500
OBJECT-TYPE=PART
OBJECT-STATUS=TRANSFERRED
```

```
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500
OBJECT-NAME=CARLSTEST.PARTS.500.B
OBJECT-TYPE=MEMBER
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500/b
OBJECT-NAME=CARLSTEST.PARTS.500.&PD
OBJECT-TYPE=MEMBER
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/500/_pd
OBJECT-NAME=CARLSTEST.PARTS.500.&PICT.03
OBJECT-TYPE=MEMBER
OBJECT-STATUS=TRANSFERRED
LOCAL-FILENAME=extra:/users/ccombs/parts/carlstest/parts/
500/_pict/03
```

Update and Replace Examples

Example 1

This example replaces the part obtained earlier using the selection scope of binder.

```
cireplace selscope=p selname=SAMPLE.BINDER.PART
```

REPLACE Command Initial Call Client to CADDS Rulebase:

```
COMMAND-SELSCOPE=P
COMMAND-SELNAME=SAMPLE.BINDER.PART
COMMAND-ATTRFILE=ATTRFILE23033
COMMAND-VAULTID=BLACKDOG
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=CADDS 5
OBJECT-NAME=SAMPLE.BINDER.PART
OBJECT-TYPE=PART
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/part
PLACEHOLDER=NO
OBJECT-REVISION=2
OBJECT-NAME=SAMPLE.BINDER.PART.A
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/
part/a
PLACEHOLDER=NO
OBJECT-REVISION=2
CGOS-FILETYPE=x'21'
OBJECT-NAME=SAMPLE.BINDER.PART.&PD
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/
part/_pd
PLACEHOLDER=NO
OBJECT-REVISION=2
CGOS-FILETYPE=x'20'
```

REPLACE Command Initial Call Rulebase to Client:

```
OBJECT-CLASS=CADDS 5
OBJECT-NAME=SAMPLE.BINDER.PART
OBJECT-TYPE=PART
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/part
OBJECT-STATUS=AVAILABLE
OBJECT-NAME=SAMPLE.BINDER.PART.A
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/
sample/binder/part/a
OBJECT-STATUS=AVAILABLE
CGOS-FILETYPE=x'21'
CGOS-PROTECTIONGROUP=x'0000'
CGOS-USERATTRIBUTES=x'19970000'
CGOS-UPDATE-DATETIME=x'C96574F6'
CGOS-CHECKSUM=x'0000'
OBJECT-NAME=SAMPLE.BINDER.PART.&PD
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/
      sample/binder/part/_pd
OBJECT-STATUS=AVAILABLE
CGOS-FILETYPE=x'20'
CGOS-PROTECTIONGROUP=x'0000'
CGOS-USERATTRIBUTES=x'19970000'
CGOS-UPDATE-DATETIME=x'C96574F6'
CGOS-CHECKSUM=x'0000'
```

Replace Command Cleanup Call Client to Rulebase:

```
COMMAND-SELSCOPE=P
COMMAND-SELNAME=SAMPLE.BINDER.PART
COMMAND-ATTRFILE=ATTRFILE23033
COMMAND-VAULTID=BLACKDOG
CLIENT-RELEASE=EDM7.1.0
SIGNON-USERID=RCC
OBJECT-CLASS=CADDS 5
OBJECT-NAME=SAMPLE.BINDER.PART
OBJECT-TYPE=PART
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/sample/binder/part
PLACEHOLDER=NO
OBJECT-REVISION=2
OBJECT-STATUS=TRANSFERRED
OBJECT-NAME=SAMPLE.BINDER.PART.&PD
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/
sample/binder/part/_pd
PLACEHOLDER=NO
OBJECT-REVISION=2
CGOS-FILETYPE=x'20'
OBJECT-STATUS=TRANSFERRED
OBJECT-NAME=SAMPLE.BINDER.PART.A
OBJECT-TYPE=MEMBER
LOCAL-FILENAME=grindelwald:/home/rcrumrin/test/
sample/binder/part/a
```

```
PLACEHOLDER=NO
OBJECT-REVISION=2
CGOS-FILETYPE=x'21'
OBJECT-STATUS=TRANSFERRED
```

Examples of Input/Output Files with Error Conditions

The following examples of rulebase input and output files include error conditions.

One or More Members of Part Not Available

```
# Rulebase input file, initial call, for Selscope = P.
OBJECT-CLASS=CADDS 5
OBJECT-NAME=CARLSTEST.PARTS.005
OBJECT-TYPE=PART
LOCAL-FILENAME=use1
# Rulebase output file, initial call, for Selscope = P.
OBJECT-NAME=CARLSTEST.PARTS.005
OBJECT-TYPE=PART
OBJECT-STATUS=UNAVAILABLE
ERROR-TEXT=One or more members of the part are unavailable
OBJECT-NAME=CARLSTEST.PARTS.005.&PD
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/vaishali/parts/use1/_pd
CGOS-FILETYPE=x‘20’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-USERATTRIBUTES=x‘00000000’
CGOS-UPDATE-DATETIME=x‘c2bc5f5b’
CGOS-CHECKSUM=x‘0000’
OBJECT-NAME=CARLSTEST.PARTS.005.B
OBJECT-TYPE=MEMBER
OBJECT-STATUS=AVAILABLE
LOCAL-FILENAME=extra:/users/vaishali/parts/use1/b
CGOS-FILETYPE=x‘21’
CGOS-PROTECTIONGROUP=x‘0000’
CGOS-USERATTRIBUTES=x‘00000000’
CGOS-UPDATE-DATETIME=x‘c2bc5f5e’
CGOS-CHECKSUM=x‘0000’
OBJECT-NAME=CARLSTEST.PARTS.005.&PICT.03
OBJECT-TYPE=MEMBER
OBJECT-STATUS=UNAVAILABLE
ERROR-TEXT=Acess denied
```

File or Part Does Not Exist

```
# Rulebase input file, initial call, for Selscope = P.
OBJECT-CLASS=CADDS 5
OBJECT-NAME=CARLSTEST.PARTS.006
OBJECT-TYPE=PART
LOCAL-FILENAME=doesnotexist
# Rulebase output file.
OBJECT-NAME=CARLSTEST.PARTS.006
OBJECT-TYPE=PART
OBJECT-STATUS=UNAVAILABLE
ERROR-TEXT=The part does not exist
# Rulebase input file, initial call,
# for Selscope = F
OBJECT-CLASS=CADDS 5
OBJECT-NAME=&BCD.COLORIGES
OBJECT-TYPE=FILE
LOCAL-FILENAME=&BCD.COLORIGES
#Rulebase output file.
OBJECT-NAME=&BCD.COLORIGES
OBJECT-TYPE=FILE
OBJECT-STATUS=UNAVAILABLE
ERROR-TEXT=The file does not exist
```

Empty Directory

```
# Rulebase input file initial call. Selscope = D
# Note that user-defined rulebases cannot accommodate directories.
OBJECT-CLASS=LOCAL
OBJECT-NAME=CARLSTEST.FILES.
OBJECT-TYPE=DIRECTORY
LOCAL-FILENAME=datadd
# Rulebase output file. Error text should be “directory empty”.
OBJECT-NAME=CARLSTEST.FILES.
OBJECT-TYPE=FILE
OBJECT-STATUS=UNAVAILABLE
LOCAL-FILENAME=extra:/users/vaishali/submit/testclient/datadd
ERROR-TEXT=The directory is empty
```

EPD.Connect Support

EPD.Connect requires that all application types be supported by an application-specific rulebase that supports the internal `LIST` command. For application types that can be converted into product structure format, the application-specific rulebase must support the internal `EXTRACT, CREATE, LOCK,` and `UNLOCK` commands.

For more information about application types, refer to the *EPD.Connect User Guide*.

Rulebases and PC Client Applications

For your rulebase to support EPD.Connect commands on the PC, the rulebase must be an executable.

When EPD.Connect is run on Windows, the EDM_<application type> variable setting specifies what rulebase executable to use to execute the required EPD.Connect command. Specify this in your `cvepd.ini` file. Refer to the "Configuring EPD.Connect" chapter of *Installing EPD.Connect, EPD Roles, and EPD Visualizer* for more information about the `cvepd.ini` file.

Applying a Rulebase

The rulebase is applied when the EPD.Connect user executes any of the following from EPD.Connect:

- File > Open (`EXTRACT`, `LOCK`, `UNLOCK` internal commands are executed)
- File > Save (`CREATE`, `LOCK`, `UNLOCK` internal commands are executed)
- File > Save As.. (`CREATE`, `LOCK`, `UNLOCK` internal commands are executed)

EPD.Connect also applies the application rulebase when the user requests a list (internal `LIST` command is executed) of files to be displayed in the EPD.Connect Data Browser or any query to list application files.

Please note: The rulebase is also applied when an EPD.Connect user executes the following Optegra Vault commands from EPD.Connect:

`STORE, GET, UPDATE, REPLACE, READ`

Selecting a Rulebase in EPD.Connect

The rulebase executable implements the EPD.Connect internal commands.The application type of the file selected in EPD.Connect determines the rulebase executable to be used.

UNIX: The information stored in the `EDM.DEFAULTS` file with the syntax

```
ENV(application type)=executable
```

allows EPD.Connect to locate the rulebase executable.

Windows: The `wedmlib.ini` file, located in the Windows Directory contains the information, with the syntax:

```
application type=executable
```

Implementing a User-defined Rulebase

Perform the following steps to implement a user-defined rulebase.

1. Create an executable that
   a. Implements the commands mentioned in the overview.
   b. Reads the rulebase input file produced by EPD.Connect
   c. Produces a rulebase output file that is read by EPD.Connect.
   d. Produces a rulebase command-specific output file to be read by EPD.Connect. The output file is specified as an input parameter in the rulebase input file.

   The content of the rulebase input and output files varies, depending on the command being executed. The content is described later in this chapter.
2. On UNIX, add a line to the `EDM.DEFAULTS` file that defines the name of your rulebase and the name of the executable. The name of the rulebase is the application type. For example, with the line below in the `EDM.DEFAULTS` file, a user can select a file of application type `MYAPPL` from the File > Open query list.

   ```
   ENV(MYAPPL)=$EPD_HOME/bin/myappl_rulebase
   ```

   The implementation, in `$EPD_HOME/bin/myappl_rulebase`, of the internal EPD.Connect commands required by the File > Open operation would then be executed.
3. On Windows, add the following line to the `wedmlib.ini` file:

   ```
   MYAPPL=$EPD_HOME\bin\myappl_rulebase
   ```

Invoking the Rulebase Upon Execution of an EPD.Connect Command

When you execute EPD.Connect commands, EPD.Connect determines the path name of the executable associated with the specified application type. The rulebase name and its full path name are defined in the `EDM.DEFAULTS` file.

EPD.Connect uses input and output files to communicate with your rulebase. After you invoke one of the above-mentioned internal commands through an EPD.Connect operation, EPD.Connect creates an input file using user-supplied data. The rulebase creates the output files using data from the input file, the local file system, and anywhere else it can access.

When EPD.Connect invokes the rulebase, the command line input to the rulebase has the following format.

```
command call_flag input_file_name output_file_name
```

- command — Is the name of the command being processed. The value can be LIST, EXTRACT, CREATE, LOCK, UNLOCK.
- call_flag — Is a flag indicating whether this is a call for initial processing or cleanup processing. The value is always initial.
- input_file_name — Is the name of the rulebase input file. EPD.Connect determines this name.
- output_file_name — Is the name of the rulebase output file. EPD.Connect determines this name.

The format of the input and output files is the same, regardless of which rulebase is called. The content of the input and output files varies slightly, depending on the command being executed.

For example, the arguments in a call to your rulebase might look like this.

```
extract initial <random_file_name1> <random_file_name2>
```

Your rulebase processes the input and creates an output file that EPD.Connect uses to continue processing the operation. If any errors occurred, the rulebase would include them in the output file. EPD.Connect then writes these errors in the audit log.

LIST Command

When EPD.Connect calls the `LIST` command, a user-defined rulebase must determine

- If the directory specified to list does exist.
- What the contents of the directory are, based on the application type rules.
- The naming convention of files involved in the transfer.

After successful completion of the command, your rulebase can do any cleanup that you specify.

Input File Content

For the `LIST` command, EPD.Connect sends the rulebase the following parameters in the order shown below.

Table 6-14 Parameters EPD.Connect for LIST Command Sends to Rulebase

Parameter	Definition
OBJECT-CLASS	Specifies the name of the application type. This is the value the user specifies.
SEARCH-MODE	Indicates where to do the listing. It is always set to LOCAL.
OBJECT-NAME	The name of the file to list. It can be '*', wild char, or pattern.
OBJECT-TYPE	Indicates whether the object is a part or a standalone file.The value can be PART or FILE.
LOCAL-FILENAME	The directory for which the content of the listing is requested.
FOLLOW-LINKS	Indicates to the rulebase if it is to follow soft links when listing the contents of the directory. The value can be Y or N. EPD.Connect always sets the value to N.

Output File Content

For the `LIST` command, the rulebase creates an output file that is empty if no files match the search (rule) criteria or with the parameters in the order shown.

If an error occurs during the listing, the file can contain only the MESSAGE-TEXT parameter. If no error occurred, the parameter MESSAGE-TEXT can not be in the file.

For each object, the output file can contain a set of OBJECT-NAME and OBJECT-STATUS parameters for each object (file) to be listed.

Table 6-15 Parameters Rulebase for LIST Command Creates for EPD.Connect

Parameter	Definition
OBJECT-CLASS	Name of the object (file) that matches the search criteria (rule) for the application type (OBJECT-CLASS) specified in the input file.
OBJECT-STATUS	This is always set to AVAILABLE.
MESSAGE-TEXT	Contains an error message, if an error occurred.

LIST Command Examples

Example 1

Here is a simple example of EPD.Connect requesting the CADDS rulebase to list all the files application type `PS` in the `/home/user/parts` directory that contains two files of type `PS`. The `LIST` command executes with no errors.

```
caddsif LIST initial /usr/tmp/nav/navJCAa003DV
/usr/tmp/nav/navJCAa003DW
```

LIST Command EPD.Connect to Rulebase:

```
OBJECT-CLASS=PS
SEARCH-MODE=LOCAL
OBJECT-NAME=*
OBJECT-TYPE=PART
LOCAL-FILENAME=/home/user/parts
FOLLOW-LINKS=N
```

LIST Command Rulebase to EPD.Connect:

```
OBJECT-NAME=CAT2PS
OBJECT-STATUS=AVAILABLE
OBJECT-NAME=PSTEST1
OBJECT-STATUS=AVAILABLE
OBJECT-NAME=STOREPS1
OBJECT-STATUS=AVAILABLE
```

Example 2

Here is a simple example of EPD.Connect asking the CADDS rulebase to list all the files application type PS in the /home/user/parts directory that contains two files of type PS. An internal interfaces rulebase error occurred.

```
caddsif LIST initial /usr/tmp/nav/navJCAa003DX
/usr/tmp/nav/navJCAa003DY
```

LIST Command EPD.Connect to Rulebase:

```
OBJECT-CLASS=PS
SEARCH-MODE=LOCAL
OBJECT-NAME=*
OBJECT-TYPE=PART
LOCAL-FILENAME=/home/user/parts
FOLLOW-LINKS=N
```

LIST Command Rulebase to EPD.Connect:

```
MESSAGE-TEXT=Error occurred in interface execution.
```

CREATE/EXTRACT Commands

When EPD.Connect calls the CREATE or EXTRACT commands, a user-defined rulebase must

- Implement the CREATE or EXTRACT functionality.
- For CREATE, create a file that is in application-specific file format from the product structure selected. For EXTRACT, create a file that is in product structure file format
- Report any errors.

Input File Content

For the CREATE or EXTRACT commands, EPD.Connect sends the rulebase the following parameters in the order shown.

Table 6-16 Parameters EPD.Connect for CREATE and EXTRACT Commands Sends to the Rulebase

Parameter	Definition
COMMAND-SELSCOPE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-SELNAME	For CREATE, specifies the name of the application-specific file to be created. For EXTRACT, specifies the application object (file) name to extract the product structure attributes.
COMMAND-LDIRNAME	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-LFNAME	For CREATE, specifies the directory where the object is to be created. For EXTRACT, specifies the directory where the object (file) specified in COMMAND-SELNAME resides.
COMMAND-REVISION	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-FILETYPE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-CATLEVEL	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-DATECRIT	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-DATE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-OVERLAY	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-CONT	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.

Table 6-16 Parameters EPD.Connect for CREATE and EXTRACT Commands Sends to the Rulebase

Parameter	Definition
COMMAND-ATTRFILE	For CREATE, specifies the product structure file from which the application assembly is created. For EXTRACT, specifies the file where product structure attributes are to be written in the product structure file format.
COMMAND-VAULTID	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
CLIENT-RELEASE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
SIGNON-USERID	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
OBJECT-CLASS	For CREATE, specifies the name of the object class to be created. For EXTRACT, specifies the name of the application type. This is the value the user specifies.
OBJECT-NAME	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
OBJECT-TYPE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
LOCAL-FILENAME	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
PLACEHOLDER	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
OBJECT-REVISION	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.

Output File Content

For the CREATE and EXTRACT commands, the rulebase creates an output file that contains a message of success or failure.

Table 6-17 Parameter Rulebase CREATE and EXTRACT Commands Creates for EPD.Connectt

Parameter	Definition
MESSAGE-TEST	Contains the severity code and a text message explaining success or failure.

Command Specific Output File Content

In addition,

- The CREATE command creates an application-specific file from the product structure file passed in the input file from EPD.Connect.
- The EXTRACT command creates a product structure file in product structure file format.

For more information about product structure format, refer to Appendix A in *Installing EPD.Connect, EPD Roles, and EPD Visualizer.*

Examples of Input and Output Files for EXTRACT and CREATE

Example 1

This is a simple example of EPD.Connect requesting the CADDS rulebase to extract the product structure from the CAMU assembly `/home/user/parts/MODEL-ASSY1` and create the product structure file `/usr/tmp/nav/navJCAa003DS` with the extracted product structure. The `EXTRACT` command executes with no errors.

```
caddsif EXTRACT initial /usr/tmp/nav/navJCAa003DT
/usr/tmp/nav/navJCAa003DU
```

EXTRACT Command EPD.Connect to Rulebase:

```
COMMAND-SELSCOPE=$$NULL$$
COMMAND-SELNAME=MODEL-ASSY1
COMMAND-LDIRNAME=$$NULL$$
COMMAND-LFNAME=/home/user/parts
COMMAND-REVISION=NULL
COMMAND-FILETYPE=$$NULL$$
COMMAND-CATLEVEL=$$NULL$$
COMMAND-DATECRIT=$$NULL$$
COMMAND-DATE=$$NULL$$
COMMAND-OVERLAY=$$NULL$$
COMMAND-CONT=$$NULL$$
COMMAND-ATTRFILE=/usr/tmp/nav/navJCAa003DS
COMMAND-VAULTID=$$NULL$$
CLIENT-RELEASE=$$NULL$$
SIGNON-USERID=$$NULL$$
OBJECT-CLASS=CAMU
OBJECT-NAME=$$NULL$$
OBJECT-TYPE=$$NULL$$
LOCAL-FILENAME=$$NULL$$
PLACEHOLDER=$$NULL$$
OBJECT-REVISION=$$NULL$$
```

EXTRACT Command Rulebase to EPD.Connect:

```
MESSAGE-TEXT=E Extracted successfully
```

Example 2

This is a simple example of EPD.Connect asking the CADDS rulebase to create the CAMU assembly `/home/user/parts/PSTOCAMU1` from the product structure file /`usr/tmp/nav/navGAAa001D0.` The `CREATE` command executes with no errors.

```
caddsif CREATE initial /usr/tmp/nav/navGAAa001D1
/usr/tmp/nav/navGAAa001D2
```

CREATE Command Epd.connect To Rulebase:

```
COMMAND-SELSCOPE=$$NULL$$
COMMAND-SELNAME=PSTOCAMU1
COMMAND-LDIRNAME=$$NULL$$
COMMAND-LFNAME=/home/user/parts
COMMAND-REVISION=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-CATLEVEL=$$NULL$$
COMMAND-DATECRIT=$$NULL$$
COMMAND-DATE=$$NULL$$
COMMAND-OVERLAY=$$NULL$$
COMMAND-CONT=$$NULL$$
COMMAND-ATTRFILE=/usr/tmp/nav/navGAAa001D0
COMMAND-VAULTID=$$NULL$$
CLIENT-RELEASE=$$NULL$$
SIGNON-USERID=$$NULL$$
OBJECT-CLASS=CAMU
OBJECT-NAME=$$NULL$$
OBJECT-TYPE=$$NULL$$
LOCAL-FILENAME=$$NULL$$
PLACEHOLDER=$$NULL$$
OBJECT-REVISION=$$NULL$$
```

CREATE Command Rulebase to EPD.Connect:

```
MESSAGE-TEXT=E Created successfully
```

LOCK/UNLOCK Commands

When EPD.Connect calls the LOCK or UNLOCK commands, a user-defined rulebase must

- Implement the LOCK or UNLOCK functionality.
- For LOCK, create a lock file that is specific to the application type. For UNLOCK, remove the lock file that is specific to the application type.
- Report any errors.

After successful completion of the command, your rulebase can do any cleanup that you specify.

Input File Content

For the LOCK and UNLOCK commands, EPD.Connect sends the rulebase the following parameters in the order shown.

Table 6-18 Parameters EPD.Connect for LOCK and UNLOCK Commands Sends to the Rulebase

Parameter	Definition
COMMAND-SELSCOPE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-SELNAME	Specifies the application object (file) name to be locked.
COMMAND-LDIRNAME	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-LFNAME	Specifies the directory where the object (file) specified in COMMAND-SELNAME resides.
COMMAND-REVISION	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-FILETYPE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-CATLEVEL	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-DATECRIT	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-DATE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-OVERLAY	This value can be “old” or “new”
COMMAND-CONT	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
COMMAND-ATTRFILE	The name of the message file where the LOCK/UNLOCK scripts write the error/warning messages.

Table 6-18 Parameters EPD.Connect for LOCK and UNLOCK Commands Sends to the Rulebase

Parameter	Definition
COMMAND-VAULTID	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
CLIENT-RELEASE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
SIGNON-USERID	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
OBJECT-CLASS	Specifies the name of the application type.
OBJECT-NAME	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
OBJECT-TYPE	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
LOCAL-FILENAME	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
PLACEHOLDER	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.
OBJECT-REVISION	This is always set to $$NULL$$ by EPD.Connect. It should be ignored by the rulebase.

Output File Content

For the `LOCK` and `UNLOCK` commands, the rulebase creates an output file that contains a message of success or failure.

Table 6-19 Parameters Rulebase for LOCK and UNLOCK Commands Creates for EPD.Connect

Parameter	Definition
MESSAGE-TEST	Contains the severity code and a text message explaining success or failure.

Command Specific Output File Content

In addition, the `LOCK` and `UNLOCK` commands create an error log file that contains application-specific error messages.

Examples of Input and Output Files for LOCK and UNLOCK

Example 1

This is a simple example of EPD.Connect asking the CADDS rulebase to lock the CAMU assembly /home/user/parts/MODEL-ASSY1 that already exists. The `LOCK` command executes with no errors.

```
caddsif EXTRACT initial /usr/tmp/nav/navJCAa003ET
/usr/tmp/nav/navJCAa003EU
```

LOCK Command EPD.Connect to Rulebase:

```
COMMAND-SELSCOPE=$$NULL$$
COMMAND-SELNAME=MODEL-ASSY1
COMMAND-LDIRNAME=$$NULL$$
COMMAND-LFNAME=/home/user/parts
COMMAND-REVISION=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-CATLEVEL=$$NULL$$
COMMAND-DATECRIT=$$NULL$$
COMMAND-DATE=$$NULL$$
COMMAND-OVERLAY=old
COMMAND-CONT=$$NULL$$
COMMAND-ATTRFILE=/usr/tmp/nav/navGCAa003ES
COMMAND-VAULTID=$$NULL$$
CLIENT-RELEASE=$$NULL$$
SIGNON-USERID=$$NULL$$
OBJECT-CLASS=CAMU
OBJECT-NAME=$$NULL$$
OBJECT-TYPE=$$NULL$$
LOCAL-FILENAME=$$NULL$$
PLACEHOLDER=$$NULL$$
OBJECT-REVISION=$$NULL$$
```

LOCK Command Rulebase to EPD.Connect:

```
MESSAGE-TEXT=E Locked successfully
```

Example 2

Here is a simple example of EPD.Connect asking the CADDS rulebase to unlock the CAMU assembly /home/user/parts/MODEL-ASSY1. The `UNLOCK` command executes with no errors.

```
caddsif UNLOCK initial /usr/tmp/nav/navJCAa003FT
/usr/tmp/nav/navJCAa003FU
```

UNLOCK Command EPD.Connect to Rulebase:

```
COMMAND-SELSCOPE=$$NULL$$
```

```
COMMAND-SELNAME=MODEL-ASSY1
COMMAND-LDIRNAME=$$NULL$$
COMMAND-LFNAME=/home/user/parts
COMMAND-REVISION=$$NULL$$
COMMAND-FILETYPE=$$NULL$$
COMMAND-CATLEVEL=$$NULL$$
COMMAND-DATECRIT=$$NULL$$
COMMAND-DATE=$$NULL$$
COMMAND-OVERLAY=old
COMMAND-CONT=$$NULL$$
COMMAND-ATTRFILE=/usr/tmp/nav/navGCAa003FS
COMMAND-VAULTID=$$NULL$$
CLIENT-RELEASE=$$NULL$$
SIGNON-USERID=$$NULL$$
OBJECT-CLASS=CAMU
OBJECT-NAME=$$NULL$$
OBJECT-TYPE=$$NULL$$
LOCAL-FILENAME=$$NULL$$
PLACEHOLDER=$$NULL$$
OBJECT-REVISION=$$NULL$$
```

LOCK Command Rulebase to EPD.Connect:

```
MESSAGE-TEXT=E Unlocked successfully
```

Chapter 7

System Administration Tasks For UNIX/NT

This chapter provides information that is specific to UNIX/NT operating systems with regard to Vault tape commands, backups, data recovery, and the e-mail trigger.

- Executing Vault Tape Commands
- Using Tapes
- Using Vault Tape Labels
- Backing Up Oracle and Vault Databases
- Backing Up the ORACLE Database
- Backing Up the Vault Database
- Keywords for the ciubkup Command
- Using ciubkup Command Lines
- Creating Your Own Backup Utility
- Creating Universal Backup Tapes
- Migrating Vault Objects or User Passwords
- Recovering from a Media or Power Failure
- Dropping and Recreating Indexes
- Activating the E-mail Trigger
- Changing Storage Pool Selection Logic

Executing Vault Tape Commands

To execute the Vault tape commands listed below, you must be locally logged on to Vault.

- ARCHIVE
- IBKUP
- LOAD
- RECSF
- RECSP
- RESTORE
- SCANTAPE
- UBKUP
- UNLOAD

With the exception of the UBKUP command, you can log on to any UNIX account on Vault to execute Vault tape commands. If a tape is not mounted, you receive an error message when you attempt to execute a tape command.

Please note: For executing Vault Tape Commands on Windows NT, do not set the tape drive as a pert of the login executable commands. When the tape device is installed on your system, use to vault commands for backup and retrieval.

Tape Device Names

On AIX, HP-UX, OSF/1, and Solaris systems, tape device names have the following format.

```
/dev/rmt/device_number
```

For example, /dev/rmt/1

On SunOS systems, tape device names have a slightly different format, as shown below.

```
/dev/rstnumber
```

For example, /dev/rst18

When following the instructions in this chapter, when you see the term tape_device_name, substitute a tape device name that is valid on your system.

Vault Logical Tape Units

You must determine what tape devices you have. You can then specify them in the `EDM.DEFAULTS` file and/or in your `.cshrc` or `.login` file.

To specify a Vault logical tape unit to be one of your tape devices, add an entry with the format below to the `EDM.DEFAULTS` file.

```
TAPEn=tape_device_name
```

For example, on a Solaris system you can enter

```
TAPE1=/dev/rmt/1
```

To set tape mappings in the `.cshrc` or `.login` file, add the entry below.

```
% setenv EDM_TAPEn tape_device_name
```

where n is 1, 2, 3, or 4 and tape_device_name is the tape drive to which you want to map the Vault logical tape unit. For example, to set the Vault logical tape unit TAPE2 to /dev/rmt/8 on a Solaris or HP-UX system, enter

```
% setenv EDM_TAPE2 /dev/rmt/8
```

Remote Backups

To perform a remote backup, you can set an environment variable, EDM_TAPE*n* as remote, where TAPEn is the tape unit to be used for backing up or restoring.

Please note: To perform remote backups both the local and the remote systems must be UNIX systems.

For example, to use tape unit TAPE1, enter the following:

```
% setenv EDM_TAPE1 node_name:tape_device_name
```

To set a default tape mapping to a remote tape device, enter

```
% setenv EDM_TAPEn node_name:tape_device_name
```

where node_name is the name of the remote system. The remote system must be set up to perform Vault tape commands. For example, on an AIX, HP-UX, OSF/1, SGI or Solaris system, enter

```
% setenv EDM_TAPE3 chaucer:/dev/rmt/1
```

The remote system must be set up to perform Vault tape commands. Refer to “Setting Up a Remote Tape Server” on page 7-8 for instructions on setting up the remote tape server.

Tape Initialization Utilities

Do not use the `tapenlabel` command to initialize 1/2-inch tapes. Use the `tapelabel` command instead.

Exabyte and 4mm DAT tape drives have a performance problem when writing physical tape marks. ANSI standard labeled tapes, which Vault uses, require tape marks. Consequently, use the `tapenlabel` command with Exabyte and 4mm DAT tapes. The `tapenlabel` command indicates to Vault that ANSI-standard tape marks are not being used. This improves the performance of Exabyte and 4mm DAT tape drives.

Using the Correct Device Setting

UNIX numbers tape devices consecutively. To find out the type of a device, load a tape in the device and enter

```
mt -f tape_device_name status (on HP-UX systems, use mt -t)
```

For example, on a Solaris system enter

```
mt -f /dev/rmt/1 status
```

This command returns the type of the device. Vault has no restrictions on the type of device you use, except that when executing the `LOAD` or `UNLOAD` command, the density must be 1600.

On AIX, HP-UX, OSF/1, SGI or Solaris tapes, you can specify tape density by adding one of the values below to the device name.

- Add h to indicate high density. Example: `/dev/rmt/2h`
- Add m to indicate medium density. Example: `/dev/rmt/3m`
- Add l to indicate low density. Example: `/dev/rmt/1l`

In addition, you can add a **c** to the device name to indicate that the contents should be compressed. You can add an **n** to the device name to specify the `nonrewind` option. For example, `/dev/rmt/1hcn.`

For Vault incremental backup and archive tapes, you must also use the b switch to read the tapes. The b switch indicates Berkley style.

On SunOS tapes, you specify tape density by identifying the tape device name as rst4, rst8, rst12, or rst16.

Please note: Do not run CI commands, such as `ciubkup` and `ciubksa`, directly from the scripts directory. During installation, you should have put /$EPD_HOME/`scripts` in your path. This allows you to run `CI` commands from any location, other than the scripts directory.

Using Tapes

This section gives details about the points to be taken into consideration while using tapes.

Using Exabyte Tapes

Restrictions and Limitations

When you use Exabyte tape drives, take into consideration the following points:

- Exabyte tapes written with no tape marks are never filled to capacity. Vault stops the operation before full capacity is reached.
- If the physical end-of-tape (PEOT) is reached before the capacity value has been attained, Vault terminates the tape operation with a fatal input/output error of the following standard Vault format:

  ```
  Processing not done - fatal return code from &1.
  RC = 30448  EC = 0
  Please notify your EDM administrator
  ```

  The fatal return code is from some low-level tape routine. The RC value indicates a premature physical end-of-tape failure. This condition is caused by one of the following user errors:

  ```
  You specified a 5 GB drive when using a 2.2 GB drive.
  You put a short capacity (for example, less than 2.2 GB)
  tape in the drive.
  ```

- Tapes with no tape marks cannot be appended by using the append-to-tape feature. If you try to append to a tape written with zero (0) physical file marks, an error is returned indicating that this option is not supported.

Using HP Servers for Tape Commands

On HP servers use tape drive `/rmt/0m` or `/rmt/0mh` for tape commands. Drive `/dev/rmt/c201d2mn` can not work.

Setting the Capacity Value

The ETAPESIZE environment variable identifies the capacity of the Exabyte tape drive. The allowable values of the variable are:

- 2 indicating a 2.2GB capacity (default)
- 5 indicating a 5GB capacity

If you need a capacity more than the default, use the `setenv` command at the shell prompt (%) to specify the capacity of the ETAPESIZE variable. For example,

```
setenv ETAPESIZE 5
```

Tape Utilities

`cibkup` : When issuing the `cibkup` command with dual tapes, you must provide two tape numbers in addition to two tape units.

```
ciibkup TAPEUNIT="(TAPE1,TAPE2)"
TAPENUM="(SCRTCH,SCRTCH)" TAPPEND=Y
```

`LOAD` : The `LOAD` command with a selection scope of Entire tape or Catalog does not store parts. Files belonging to parts are stored but there is no recognition of any parts.

`UNLOAD:` Issuing `UNLOAD (GET)` for a catalog that spans two tapes displays the following message during the second tape:

```
Processing not done.
```

Scanning the second tape shows a successful copy of the files to tape but the sign out of the files is never accomplished. All files in the catalog have a blank system code.

`UNMARK:` When you unmark (`UNMARK`) for archive a file set and you don't specify a current revision, all revisions of the files associated with the file set are unmarked for archive.

Setting Up a Remote Tape Server

Please note: Remote Tape Server cannot be used in Windows NT environment.

To set up remote tape for on UNIX perform the following steps:

1. In the **.cshrc** file on the Optegra server node, for those accounts which can execute tape functions, add or replace:

   ```
   setenv EDM_TAPE1 REMOTE_NODE:/dev/rstxx
   ```

   where REMOTE_NODE names the node that contains the remote tape server executable `asrtserv` and /dev/`rstxx` is replaced by the actual device address for the tape device on that node. Similarly, EDM_TAPE2, EDM_TAPE3, and EDM_TAPE4 can also be defined.

2. Edit the `nsm.config` file Tape_Server MODEL as follows:

   - Change server_system to the name of the DOMAIN in which the Tape_Server is to run. This is usually the name of the Optegra Vault server.
   - Change the PATH value to define the path to the `asrtserv` executable, as it resides on the remote machine (the machine that has the tape drive).
   - Change the OWNER value to define the user ID under whose account privileges the `asrtserv` executable can run. This user ID should have operator privileges, so that it is capable of mounting tapes.

     ```
     MODEL(Tape_Server)
     DOMAIN(server_system)(EDM domain name)
     AE(EDMTS_CLASSA_V1,tapegrp,1)
     SERVER(NO_SERVER_TIMEOUTS)
     PATH(/$EPD_HOMEclient/bin/asrtserv)
     (path on system with tape drive)
     DEMAND(ANYTIME)
     OWNER(remote_userid)(user ID to execute asrtserv)
     WORKDIR(/tmp)
     CONCURRENCY(1,1)
     CLOSE
     GRPCTL(0,0,2)
     END_MODEL
     ```

   - You must also have the following:

     ```
     NODE(nodename1)(machine that has tape drive/asrtserv)
     ALIAS(NODENAME1)
     @EDMClient
     @Tape_Server
     ```

3. In the `.cshrc` of the account for the remote tape server, establish the environment variable EDM_TAPE1, EDM_TAPE2, EDM_TAPE3, and EDM_TAPE4. These variables should take the standard form:

   ```
   setenv EDM_TAPE1 /dev/rstxx
   ```

4. Place the following files in a directory defined by the path mentioned earlier (for example, `$EPD_HOMEclient/remotetape`):
   - `ansprcoa`
   - `asrtserv`
   - `pm.config` (`pm.config` points to the domain where the Vault server is located.)

Please note: This file must be in the same directory listed in the `nsm.config` file, such as `/$EPD_HOMEclient/bin/asrtserv.`

5. On a remote system where PATH is defined for the `asrtserv` file, enter:

```
%setenv ANAPATH
$EPD_HOMEclient/remotetape/pm.config
%ansprcoa
```

6. On the Vault server system, execute `nsmquery -pca` to make sure that the `pca` process on the remote system executed correctly.

Please note: `cibkup` (Universal Backup) is not supported on remote tape.

Using Vault Tape Labels

This sections gives directions for using labels.

UNLOAD and UBKUP Commands

Tapes to be used for the UNLOAD and UBKUP commands do not need to be labeled.

Please note: All tape commands except UBKUP work in a similar manner for both Windows NT and UNIX. In Windows NT, UBKUP has the backup utility (NTBACKUP) which helps to restore the pool files before running RECSP.

IBKUP and ARCHIVE Commands

Tapes used for the IBKUP (incremental backup) and ARCHIVE commands must be labeled before Vault uses them. If you want to use the label already on the tape, enter SCRTCH as the tape label.

Label Format

Tape labels can have up to six characters (A-Z, 0-9); lowercase letters are converted to uppercase. If the label has less than six characters, Vault left-justifies it. This is the value you enter with the TAPENUM keyword used with the IBKUP and ARCHIVE commands.

Tape Format

Vault tape labels initialize tapes in one of these formats:

- ANSI-standard format (for tapes used on 1/2-inch tape drives).
- Native-Vault format (for tapes used with Exabyte tape drives). There are two kinds of native-Vault formats.
  - The first kind has one tape mark per file that is used for positioning. You can append to a tape labeled this way.
  - The second kind has no tape marks. It uses a logical tape mark instead. You cannot append to this kind of tape.

Warning

Because of the nonconformance of Exabyte and 4mm DAT tapes to industry standards (ANSI), do not use Exabyte or 4mm DAT tapes with the Vault ARCHIVE command for long-term offline storage.

Labeling Vault Tapes for 1/2-Inch Tape Drives

To label Vault tapes with an ANSI-standard label for a 1/2-inch tape drive, use the Vault `tapelabel` utility, as shown below.

```
% tapelabel logical_tape_unit label
```

Where logical_tape_unit is the name of the Vault logical tape unit (TAPE1, TAPE2, TAPE3, or TAPE4) and label is the ANSI tape label (up to six characters long).

For example, to write the ANSI-standard label IBK004 on the tape on Vault logical tape unit TAPE1, enter

```
% tapelabel TAPE1 ibk004
```

If no messages are returned after you execute the `tapelabel` utility, the labeling has been successful.

Labeling Vault Tapes for Exabyte and DAT Tape Drives

To label Vault tapes with a native-Vault label for use on an Exabyte or 4mm DAT drive, use the Vault `tapenlabel` utility, as shown below.

```
% tapenlabel logical_tape_unit label number_of_tapemarks
```

Where logical_tape_unit is the name of the Vault logical tape unit (TAPE1, TAPE2, TAPE3, or TAPE4), label is the native-Vault tape label (up to six characters long), and number_of_tapemarks indicates whether you want one or zero tape marks per file.

For example, to write the native-Vault label IBK004 on the tape on Vault logical tape unit TAPE1 and allow tape append operations, enter

```
% tapenlabel TAPE1 ibk004 1
```

If no messages are returned after you execute the `tapenlabel` utility, the labeling has been successful.

Reading Vault Tape Labels

To read the label of a tape that has an ANSI-standard or native-Vault label, use the Vault `taperead` utility, as shown below:

```
% taperead logical_tape_unit
```

Where logical_tape_unit is the name of the Vault logical tape unit (TAPE1, TAPE2, TAPE3, or TAPE4).

For example, to read the label on the tape on Vault logical tape unit TAPE4, enter

```
% taperead TAPE4
```

Result of Label Utilities

If you execute `tapenlabel` TAPE1 IBK002 1 and then `taperead` TAPE1 you get the following:

```
Tape has native EDM format - version 2 (One tapemark per file)
Tape serial number is  :  IBK002
Tape creation date was :  07/28/93
Tape expiration date is:  07/28/93
Tape labeled using tapenlabel by userid: lila
```

If you execute `tapenlabel` TAPE1 IBK002 0 and then `taperead` TAPE1 you get the following:

```
Tape has native EDM format - version 1 (Zero tapemarks per file)
Tape serial number is  :  IBK002
Tape creation date was :  07/28/93
Tape expiration date is:  07/28/93
Tape labeled using tapenlabel by userid: avi
```

If you execute `tapelabel` TAPE1 IBK002 and then `taperead` TAPE1 you get the following:

```
Tape has ANSI standard format (Three tapemarks per file)
Tape serial number is  :  IBK002
Tape creation date was :  07/28/93
Tape expiration date is:  07/28/93
Tape labeled using tapenlabel by userid:sam
```

Backing Up Oracle and Vault Databases

Files are stored within the Vault database in storage pools owned by Vault. Vault control information regarding these files is held in Oracle control tables. These tables are owned by Vault.

For example, the current status of an Vault file is obtained by referring to the appropriate Oracle control table. This status information indicates whether the file is signed out (and to whom), deleted, or archived. The file itself is stored in one of Vault's storage pools.

Backup mechanisms protect against data loss from media failure or inadvertent data corruption. Vault backup can be divided into two distinct areas:

- Oracle database backup
- Vault database backup

Both the Oracle database containing Vault control data and the storage pools containing file data must be backed up to maintain database consistency and allow recovery in case of a media failure.

Please note: You must also back up Oracle runtime software, Vault executable software, and any other Vault system files (for example, Vault user login files). Use the regular backup procedures on your Vault host system.

The figure on the following page illustrates the recommended minimal schedule for backup routines. It assumes that Oracle's redo log files are used in ARCHIVELOG mode and that the redo log files are automatically archived.

Table 7-1 Recommended Minimal Backup Schedule for EDM

Monday	Tuesday	Wednesday	Thursday	Friday
Vault incremental backup ORACLE image backup	Vault incremental backup	Vault incremental backup	Vault incremental backup	Vault incremental backup
Vault incremental backup	Vault incremental backup	Vault incremental backup	EDM incremental backup	Vault incremental backup UNIX/NT system backup
Vault incremental backup ORACLE image backup	Vault incremental backup	Vault incremental backup	EDM incremental backup	Vault incremental backup
Vault incremental backup	Vault incremental backup	Vault incremental backup	Vault incremental backup	Vault universal backup

Backing Up the ORACLE Database

You must regularly back up Vault's Oracle database tables, redo log files, and control files. The Vault database must be shut down while you do an image backup. The Oracle database is copied to tape using standard UNIX backup utilities (`dump, tar,` or `cpio`).

Image Backup

Run an image backup at least twice a month. Maintain at least two sets of image backup tapes.

- The first set is for the current image backup
- The second set is for the previous image backup in case the system fails during the current backup.

Redo Log Files

Every Oracle database requires a minimum of two redo log files. The redo log records changes to the database. In the event of database corruption, the process of reapplying the redo log to the database is called roll-forward recovery.

Vault's database recovery mechanisms require that you use Oracle's redo log files in ARCHIVELOG mode to provide complete roll-forward recovery in the event of a media failure (disk crash). If you do not use ARCHIVELOG mode, you cannot use the EDM RECSP (recover storage pool) command to restore Vault storage pools, because the Oracle and Vault databases would subsequently be out of sync.

The redo log files should be automatically archived. With automatic archiving, online redo log files are archived when the checkpoint interval (specified in the `init.ora` file) is reached. The online redo log is then copied to an offline redo log (also specified in the `init.ora` file).

Save the offline redo log files on disk back to the last image backup. Save the previous set of redo log tapes in case of a system failure during the backup or taping off of redo log files.

Please note: For more information on backing up the Oracle database, see the Oracle Corporation documentation.

Backing Up the Vault Database

Vault allows you to perform two kinds of backups:

- Incremental
- Universal

Incremental Backups

When you execute the incremental backup command (`IBKUP`), Vault copies to tape only the files that have been added or changed since the last incremental backup. Run incremental backups at least once a day.

Keep incremental backup tapes for at least one complete rotation between universal backups. Place an exterior label on each tape that matches the tape label you entered with the `IBKUP` command. Regularly send these tapes offsite for storage. Refer to Chapter 4, “Performing a Backup of Files” for information about `IBKUP` that applies to all Vault platforms.

Universal Backups

When you execute the universal backup command (`ciubkup`), Vault copies to tape the contents of the storage pools. Run universal backups at least once a month. You must be a member of the Operator group in order to use the `ciubkup` command. If you are not a member, some commands might fail on an access violation. The contents of each storage pool are backed up to tape, one storage pool at a time. Multiple storage pools can be on a tape.

Keep a minimum of two sets of universal backup tapes: one set for the current universal backup, and one set for the previous universal backup. Rotate universal backup tapes annually, and send these tapes offsite for storage.

Please note: SunOS machines use the `bar` command instead of `cpio` in order to span tapes. The `cpio` utility does not support the spanning of tapes in the SunOS environment.

Using the cpio Command

Use the `cpio` command to perform the Vault universal backup. Although the `cpio` command writes headers to identify archived pools, it does not label the tapes. You must physically label universal backup tapes with the date and time you ran the `cpio` command.

Some UBKUP tapes can contain more than one pool, while large pools might span multiple tapes.

Please note: On Windows NT, NTBACKUP utility is used for restoring backup. You can read the tape contents using NTBACKUP for verification before restoring them.

In each step on the next page, the tape_device specified must be a nonrewinding device. On AIX, HP-UX, OSF/1, SGI and Solaris systems, the tape device name must have the format

```
/dev/rmt/device_numbern
```

For example, `/dev/rmt/1n`

On SunOS systems, the tape device name must have the format

```
/dev/ntape_device_name
```

For example, `/dev/nrst18`

You must always include the `n` option to indicate a nonrewinding device when performing a universal backup.

Finding the cpio Archive of a Storage Pool

You should physically label each UBKUP tape spool with the date of the UBKUP and the pool names that are included on that tape. The process for identifying the `cpio` archives on a UBKUP tape is outlined below.

To read the first archive on a UBKUP tape enter the command below.

On any system, except Solaris, enter

```
% cpio -icBt < tape_device_name
```

On Solaris systems, enter

```
% cpio -ict -B -I tape_device_name
```

The system then lists the files in each pool.

```
/pool1/AEM93288.PDM
/pool1/AEN93288.PDM
/pool1/AEO93288.PDM
/pool1/AEP93288.PDM
/pool1/AEQ93288.PDM
```

```
/pool1/AER93288.PDM
/pool1/AES93288.PDM
/pool1/AET93288.PDM
/pool1/AEU93288.PDM
```

To read any subsequent archives on the same tape, you must skip past the tape EOF mark with the UNIX `mt` command and then enter the `cpio` command again.

On AIX, OSF/1, SGI, Solaris, and SunOS systems, enter the following:

```
% mt -f tape_device_name fsf 1
```

On HP-UX systems, enter

```
% mt -t tape_device_name fsf 1
```

Entering the `cpio` command again would list the files in the next pool.

```
/pool2/AAB93183.PDM
/pool2/AAN93201.PDM
/pool2/AAO93201.PDM
/pool2/AAP93201.PDM
/pool2/AAQ93201.PDM
/pool2/AAU93232.PDM
/pool2/AAV93232.PDM
```

Note that Vault 5.0.2 uses the `bar` command on SunOS while Vault 5.1.2 uses the `cpio` command on SunOS.

To read the first archive ID header on a UBKUP tape enter

```
% bar -Rf tape_device
```

The system displays the following:

```
bar: Archive ID = EDMVault.ubkup.POOL3.950904.103743;
date = 09/04/05; time = 15:37; volume number = 1;
```

For example:

```
% bar -Rf /dev/nrst0
bar: Archive ID = Vault.ubkup.POOL3.950904.093743;
date = 09/04/05; time = 09:37; volume number = 1;
% mt -f /dev/nrst0 fsf 1
% bar -Rf /dev/nrst0
bar: Archive ID = EDMVault.ubkup.POOL3.950904.100529;
date = 09/04/05; time = 10:05; volume number = 1;
% mt -f /dev/nrst0 fsf 1
% bar -Rf /dev/nrst0
bar: Insert volume 2 and press Return when ready.
```

At this point during -R option when bar prompts for volume 2, press CTRL-C, since -R works on a per-volume basis.

```
/*** rewound and unloaded first tape; loaded second tape ***/
% bar -Rf /dev/nrst0
Archive ID = EDMVault.ubkup.POOL3.950904.100529;
date = 09/04/05; time = 10:05; volume number = 2;
% mt -f /dev/nrst0 fsf 1
% bar -Rf /dev/nrst0
bar: Archive ID = EDMVault.ubkup.POOL6.950904.102043;
date = 09/04/05; time = 10:20; volume number = 1;
% mt -f /dev/nrst0 fsf 1
% bar -Rf /dev/nrst0
bar: Archive ID = EDMVault.ubkup.POOL6.950904.104528;
date = 09/04/05; time = 10:45; volume number = 1;
% mt -f /dev/nrst0 fsf 1
% bar -Rf /dev/nrst0
bar: Insert volume 2 and press Return when ready.
```

At this point during -R option when bar prompts for volume 2, press CTRL-C, since -R works on a per-volume basis.

Keywords for the ciubkup Command

This section provides a description of the keywords used with the `ciubkup` command and found in the `ubkup.config` file. After you determine the values required by your site, and before you run the first universal backup, edit the `ubkup.config` file. The values in the file are the default values used when you execute the `ciubkup` command.

Please note: The keywords and the `ubkup.config` file work in the similar way on Windows NT.

TAPEUNIT

`TAPEUNIT=logical_tape_unit`

logical_tape_unit identifies the Vault logical tape unit where the tape is to be created. It can be TAPE1 (the default), TAPE2, TAPE3, or TAPE4.

RESTART

`RESTART=Y or N`

If a previous universal backup was aborted, you can restart the backup at the point it was aborted. Enter RESTART=Y to start the backup at the point it stopped during the previous ciubkup command execution. The default is Y (begin a complete backup operation).

CLEAR

CLEAR=Y or N

Enter CLEAR=Y to delete previous entries in the Backup Table. The default is N.

CYCLES

CYCLES=number_of_cycles

If CLEAR=Y (yes), enter the number of universal backup cycles to keep in the Backup Table. The default is 1, which means only the current universal backup can be kept in the Backup Table. If CLEAR=N, the CYCLES keyword is ignored.

The number of cycles to keep in the Backup Table should be the same as the number of universal backup tape sets in your rotation. For example, for an annual rotation of universal backup tape sets, enter the number of universal backups performed in a year.

INPUT

INPUT=L or F

Enter INPUT=L to indicate that the storage pools to be backed up are in a list. Enter INPUT=F if the names of the storage pools to be backed up are in a file. The default value is list.

POOLS

POOLS=* or list_of_pools or local_filename

Use this keyword in conjunction with the INPUT keyword to tell the ciubkup command the name of the storage pools to process.

Use the * (asterisk) value with INPUT=L to mean all storage pools. This is the default value.

Use the list_of_pools with INPUT=L to provide a list of the storage pool names to be processed. If there is more than one pool in the list, the pool names must be separated by commas and the entire list delimited by parentheses.

Use the local_file_name with INPUT=F to provide the name of a local file that contains the storage pool names to be processed.

Please note: The local file containing the names of storage pools to be processed must be a text file. Each storage pool name must be on a separate line in the file.

APPEND

APPEND=Y or N

Append new audit information to audit file? N means that Vault erases the content of the existing audit file. Default is Y.

COMPACT

COMPACT=Y or N

Compress the data as the pools are backed up? Default is N.

PARTIAL

```
PARTIAL=Y or N
```

Continue the backup even if some of the pools are currently unavailable? Default is N.

PAUSE

```
PAUSE=Y or N
```

Query the operator during backup? For example, if a pool is busy and PAUSE=**Y,** you can instruct Vault to try again. When PAUSE=N, the backup aborts. Default is Y.

UBKUPNAME

```
UBKUPNAME=executable
```

Use this keyword to specify the name of an executable to be used for performing a back up of storage pools. Specify a fully qualified path name for the executable. When you do not specify this keyword, Vault uses the default utility shown in the Creating Your Own Backup Utility, later in this chapter. See that section for additional information.

Overriding the Contents of the ubkup.config File

To use the keyword values listed in the `ubkup.config` file, issue the `ciubkup` command with no arguments. To override one or more values in the `ubkup.config` file, enter the `ciubkup` command with the keywords and their values on the command line.

Here is an example of a `ciubkup` command line with keywords that override those in the `ubkup.config` file.

```
ciubkup TAPEUNIT=TAPE2 RESTART=Y CLEAR=Y CYCLES=2
```

This command line

- Creates the backup tape on logical tape unit TAPE2.
- Restarts backup where the previous backup operation stopped.
- Clears the Backup Table of previous entries except for the current and last universal backups. Only entries for the previous universal backup and the backup you are about to perform can remain in the table after the execution of the command.

Using ciubkup Command Lines

The universal backup command creates a backup tape and performs other tasks if you choose.

Please note: This command works in a similar way on Windows NT. The only difference is in the audit trail file in the header which contains the following instead of <cpio archive#3 header:>

```
<ntbackup archive#3 header:>
```

The following are three sample `ciubkup` command lines that you can use as models. The first uses the defaults provided by Vault.

Example One

```
% ciubkup
```

This command line

- Creates the backup tape on logical tape unit TAPE1
- Creates a complete backup tape
- Does not clear the Backup Table of previous entries

These default values come from the `ubkup.config` file, an example of which is shown below.

```
APPEND=YES
Append to the backup audit file?

CLEAR=NO
Clear the backup table?

COMPACT=NO
Compress the data on tape?

CYCLES=1
The number of universal backup cycles to keep.

INPUT=L
L" for list, "F" for file.

PARTIAL=NO
Continue even if some pools are unavailable?

PAUSE=YES
Query the user during the backup?

POOLS=*
```

```
"*" (all pools), OR a list of pools, OR filename

PROMPT=YES
RESERVED FOR FUTURE USE.

RESTART=YES
Start where universal backup stopped last time?

TAPEUNIT=TAPE1
Appropriate backup device.

UBKUPNAME=UBUCPIO
se the default backup utility.
```

Example Two

```
% ciubkup tapeunit=tape4 restart=y clear=y cycles=2
```

This command line

- Creates the backup tape on logical tape unit TAPE4.
- Restarts the backup from where the previous backup operation stopped.
- Clears the Backup Table of previous entries except for the current and last universal backups. Only entries for the previous universal backup and the backup you are about to perform can remain in the table after the execution of the command.

Example Three

```
% ciubkup restart=y cycles=2 input=l pools=(pool1,pool2,pool3)
```

This command line

- Restarts the backup from where the previous backup operation stopped.
- Clears the Backup Table of previous entries, except for the current and last universal backup cycles. Only the entries for the previous universal backup and the backup you are about to perform can remain in the table after the execution of the command.
- Indicates that the storage pools to be backed up are in a list.
- Gives the `ciubkup` command the list of names of the storage pools to process.

Universal Backup Audit File

Vault generates an audit trail containing the storage pool names and the date and time of the backup. Use the audit trail to assist you in restoring the destroyed pools from the universal backup tape(s) prior to running the storage pool recovery command (RECSP).

Vault places a copy of the audit trail in your local storage area under the name UBKUP.EDMAUDIT. The example on the next page illustrates a portion of the audit file.

At the completion of the ciubkup command, print the Universal Backup audit file and attach the audit report to the set of tapes produced by the Universal Backup utility. Unless the RESTART option is set to yes, Vault overwrites the old audit file with the new audit trail each time you execute the ciubkup command.

```
27-Sep-93 15:12:15: About to back up storage pool POOL1
(/EDMPOOLS/pool1).
cpio archive #1 header: ″EDMVault.ubkup.POOL1.910927.151215″
27-Sep-93 15:12:51: Storage pool POOL1 (/EDMPOOLS/pool1) has been
backed up.
27-Sep-93 15:12:55: About to back up storage pool POOL2
(/EDMPOOLS/pool2).
cpio archive #2 header: ″EDMVault.ubkup.POOL2.910927.151255″
27-Sep-93 15:13:15: Storage pool POOL2 (/EDMPOOLS/pool2) has been
backed up.
27-Sep-93 15:13:17: About to back up storage pool POOL3
(/EDMPOOLS/pool3).
cpio archive #3 header: ″EDMVault.ubkup.POOL3.910927.151317″
27-Sep-91 15:13:40: Storage pool POOL3 (/EDMPOOLS/pool3) has been
backed up.
27-Sep-93 15:13:42: About to back up storage pool POOL4
(/EDMPOOLS/pool4).
cpio archive #4 header: ″EDMVault.ubkup.POOL4.910927.151342″
27-Sep-91 15:14:01: Storage pool POOL4 (/EDMPOOLS/pool4) has been
backed up.
27-Sep-93 15:14:03: About to back up storage pool POOL5
(/EDMPOOLS/pool5).
cpio archive #5 header: ″EDMVault.ubkup.POOL5.910927.151403″
27-Sep-91 15:14:17: Storage pool POOL5 (/EDMPOOLS/pool5) has been
backed up.
Universal backup is complete.
```

Creating Your Own Backup Utility

You can write the executable that Vault uses to back up storage pools when you issue the universal backup (`ciubkup`) command. Specify your own executable with the UBKUPNAME keyword when issuing `ciubkup`. You can also specify the name of an executable in the `ubkup.config` file.

Vault invokes your executable for each storage pool being backed up.

What the Executable Must Include

When you write your executable, follow the structure of the default backup utility shown in the example on the next page.

Vault passes the following arguments to your executable. Your executable must be able to handle these arguments correctly.

- TAPEUNIT: Name of the tape device being used by `ciubkup`.
- POOLPATH: Name of a partition on the local disk or name of an NFS-mounted disk partition.
- POOLNAME: Name of the storage pool being backed up.
- COMPACT: Indicates whether or not to compress the data.
- NUM: Process ID for unique names.

Your executable must set up its own execution environment. This includes definition of path requirements, special environment variables, and any shell preferences.

If you want to backup multiple storage pools on a single tape, your executable must be able to detect the end of the tape.

Example: Default Backup Utility

When you do not specify your own backup utility, Vault uses the utility shown below.

```
#!/bin/sh
#
#  %W% %E% PCI
#
# set -x  # Debugging switch
if [ $# -eq 0 ]
then
    echo ”Argument count incorrect.”
```

```
    exit 1
fi
# Determine the OS Type
MACHOS="'uname'"
# If OS is sunOS, determine whether it is SOLARIS
if [ "${MACHOS}" = "SunOS" ]
then
       VERSION='uname -r | awk -F. '{print $1}''
       if [ "$VERSION" -gt 4 ]
          then
               MACHOS="SOLARIS"
       fi
fi
TAPEUNIT=$1  # /dev/rmt/0
POOLPATH=$2  # /EDMPOOLS/pool1
POOLNAME=$3  # POOL1
COMPACT=$4   # YES or NO
HEADER=$5    # Not used by cpio. A bar option
NUM=$6       # The process ID. For unique # names
exit 0
if [ ! -f /bin/cpio ]
then
    echo FILE /bin/cpio MISSING
    exit 1
fi
if [ $COMPACT = "YES" ]
then
    echo "COMPRESSION OF DATA NOT POSSIBLE"
fi
# Use the correct cpio syntax
if [ "${MACHOS}" = "SOLARIS" ]
then
#   ls $POOLPATH/*.* | /bin/cpio -ovc -B -O $TAPEUNIT >
AUDIT.$POOLNAME.$NUM
    find $POOLPATH -name "*.*" -print | /bin/cpio -ovc -B -O
       $TAPEUNIT > AUDIT.$POOLNAME.$NUM
else
# code below is for machines AIX, OSF/1, HP-UX, IRIX6.5 and sunOS
 find $POOLPATH -name "*.*" -print | /bin/cpio -ocB > $TAPEUNIT
    # Save the names of the files to the audit file
    find $POOLPATH -name "*.*" -print > AUDIT.$POOLNAME.$NUM
fi
if [ $? -ne 0 ]
then
    exit 1
fi
```

Creating Universal Backup Tapes

You can run the `ciubkup` command while others continue to use the system; `ciubkup` locks only the storage pool it is backing up. No new files are written to that storage pool, but users can access the storage pool in read mode and write to other storage pools while the `ciubkup` command is running.

Before you run the `ciubkup` command, run the `IBKUP` (incremental backup) command to protect against data loss if you have a system failure during the universal backup.

After running the `ciubkup` command, you might want to perform another incremental backup on any new or modified files that were added to the database during the universal backup.

You must have the `UBKUP` command in your public command list to create a universal backup of the Vault database.

Please note: This command cannot be used with the menus or through the programmatic interface.

Here is the procedure to run the universal backup.

1. Enter the `NEWGRP` command.
2. Log on to the account that owns Vault software.
3. Mount the first tape on the physical tape device mapped to the Vault logical tape unit (TAPE*n)* you want to use.
4. Sign on to Vault using the command-line format.
5. Execute the universal backup command:

```
% ciubkup keywords
```

   After you enter the correct keywords and values and press the `RETURN` key, the `cpio` command takes control and the backup operation begins. If additional tapes are required, the `cpio` command prompts you to mount another tape.

6. Unload the last backup tape.

Warning

Do not use CTRL-C to interrupt the universal backup procedure. When you restart the process, it is possible that a storage pool can be locked.

Perform Backup from the Account Owning Storage Pools

When performing a universal backup, you should be logged in to the account where Vault software is loaded. If you are not, you receive a warning message.

```
Log on under the correct name and reenter the command.
```

Migrating Vault Objects or User Passwords

The following procedures explain how to migrate Vault objects and user passwords from one operating system to another using the `adpwmig` utility. The `adpwmig` utility exists in the `$EPD_HOME/install` directory.

Please note: Only the Vault administrator can execute this utility.

Prerequisites

The required Oracle and Vault versions must be installed and set up on the source and destination systems.

Source System

Do the following on the source system:

1. Change directory to the `$EDM_HOME/install` directory.

   ```
   %> cd $EDM_HOME/install
   ```

2. Execute the `adpwmig` utility with the `export` option.

   ```
   %> adpwmig export
   ```

3. Export the entire database from the Oracle account. For more information, refer to *Installing Vault and Locator.*
4. Transfer the contents of the storage pools to the respective destination storage drives.

Please note: The contents of the storage pools must be moved physically.

Destination System

Do the following on the destination system:

1. Stop Vault processes from `$EDM_HOME` as `edmapl`.

   ```
   %> nsmstop -pca
   %> nsmstop -all
   ```

2. Drop the Oracle users from the Oracle account.

   ```
   %> Drop user asm cascade;
   %> Drop user edmui cascade;
   %> Drop user pdmdm cascade;
   %> Drop user pdmqf cascade;
   %> Drop user edmdv cascade;
   %> Drop user edmattr cascade;
   ```

3. Import the entire database from the Oracle account. For more information, refer to *Installing Vault and Locator.*
4. Change the directory to $EDM_HOME/install.

```
%> cd $EDM_HOME/install
```

5. Execute the adpwmig utility with the import option.

```
%> adpwmig import
```

6. Update the DM_POOL_INFO table to reflect the destination storage drives.
7. Change the status of all the storage pools.

```
%> cichgsps poolname=storage-pool-name
poolstat=storage-pool-status
```

8. Update the DM_VAULT_CONFIG table to add an entry of the destination Vault as a Self Vault.

   As a result of the import operation, the DM_VAULT_CONFIG is identical to the source system.

   a. Remove the destination Vault entry as the Self Vault from the DM_VAULT_CONFIG table as Oracle user pdmdm.

```
%> delete from dm_vault_config where
dm_vault_id=source_vault_id

%> commit;
```

   b. Add the destination Vault as the Self Vault in the DM_VAULT_CONFIG table from $EPD_HOME.

```
%> ciaddvault vaultid=dest_vault type=S
seqno=seq_no node=dest_node
```

9. In a Distributed Vault setup, if the destination Vault is a DOD, make an entry of type D in the DM_VAULT_CONFIG.

   a. If there already is an entry of some other Vault as a Self DOD, remove it by giving the command as Oracle user pdmdm.

```
%> delete from dm_vault_config where dm_vault_type='D'
%> commit;
```

   b. Add the destination Vault as the DOD Vault.

```
%> ciaddadod vaultid=dest_vault_id node=dest_vault_name
type=SELF
```

10. Stop and start the Vault servers.

The edmdv user must be exported, dropped, or imported only if the source or destination Vault is a DOD.

Recovering from a Media or Power Failure

You can recover from media failures if you maintain the three types of regular backup tapes. If a media failure damages disks containing Vault's Oracle control tables or storage pools, you can restore the Vault database at least up to the last incremental backup.

Files stored within a defective storage pool not previously backed up cannot be recovered. These are files that were stored in Vault between the time the last backup (incremental or universal) was performed and the time the storage pool was destroyed or became defective.

Please note: Modifications made to files during this time are lost.

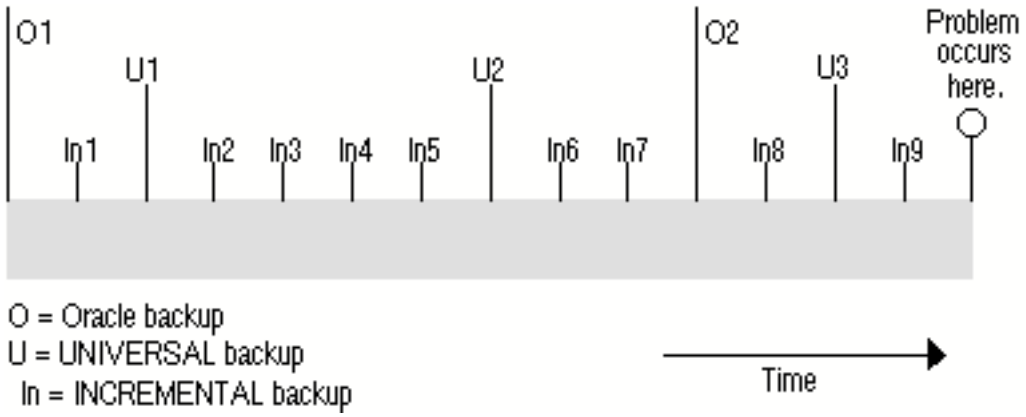
Recovering the Oracle Database and Vault Storage Pools

To recover from a media failure affecting both ORACLE disk(s) and storage pool disk(s), you need

- The most recent set of Oracle backup tapes
- The most recent set of Vault universal backup tapes
- The most recent Vault incremental backup tapes

After repairing or replacing the failed disk device(s), restore both the Oracle database and Vault storage pools as described below. This procedure synchronizes the Oracle control tables and the storage pools. The following figure illustrates Oracle and Vault backups.

Figure 7-1 Chronological Chart of Oracle and EDM Backups



Procedure

Given the chart of Oracle and Vault backups shown in the previous figure, if both an Oracle disk and a storage pool disk are lost at the same time, perform the following steps to recover the Vault database.

1. Restore the Oracle database from ARCHIVE tape O2. The roll-forward of Oracle takes the database forward from point O2 to the moment before Oracle aborted.
2. Load the last Vault universal backup tape (U3) containing the lost storage pool(s) to recover the Vault database to point U3.
3. Restore the storage pools using the `RECSP` command. This command corrects the database from point U3 onward until at least point In9 where the last incremental backup was performed. Vault then attempts to recover the database from point In9 to where the problem occurred. The Oracle database is synchronized with Vault's data disks following point In9 to reflect the recovered state of the database.

   You can lose data from the most current copies of files between point In9 and where the problem occurred. This is largely dependent on the number of disks corrupted or destroyed and the frequency of your incremental backups.

When recovering a storage pool, you should be logged in to the account where Vault software is loaded. If you are not, you receive a warning message.

Log on under the correct name and reenter the command.

Recovering the Oracle Database

Restore the Oracle database from the backup tapes by following these instructions.

To recover from a media failure on an Oracle disk, you need the most recent set of Oracle backup tapes. After repairing or replacing the failed disk device(s), follow these steps to restore the Oracle database.

1. Log on as root or log on to the Oracle account that owns Vault's control tables.
2. Load the most recent Oracle database backup tape.
3. If you used the `tar` command to create the tape, restore Vault control tables by entering

   ```
   % tar xvf tape_device_name
   ```

   where tape_device_name is the name of the physical tape device on which you are loading the backup tape.

If you used the dump utility to create the tape, restore Vault control tables by entering

```
% restore -rf tape_device_name directory
```

where tape_device_name is the name of the physical tape device on which you are loading the backup tape and directory is the name of the directory from which you copied the files to tape.

Here is an example of the command you would enter on any system except a SunOS system.

```
% restore -rf /dev/rst18 /$EPD_HOME_dbs
```

4. Apply Oracle redo log files from tape to restore the Oracle database to the moment of failure.

Recovering Vault Storage Pools

Restore Vault storage pools from the backup tapes with these instructions.To recover from a media failure on a storage pool disk, you need

- The most recent set of Vault universal backup tapes
- The most recent Vault incremental backup tapes

Use the CHGSPS command to change the status of the destroyed pools to 5 (unavailable due to media failure). After repairing or replacing the failed disk device(s), follow these steps to recover one or more Vault storage pools.

1. Log on to the account that owns Vault software.

   It is important that you physically label each UBKUP tape with the date of the UBKUP and the pool names that are included on that tape. (See Chapter 4, “Backing Up Files”, for more information.)

Please note: Some UBKUP tapes can contain more than one pool, while large pools can span multiple tapes.

   Before beginning the restoration process, assemble all the UBKUP tapes needed to restore that pool.

   There are two main steps in restoring a pool from a universal backup: identifying the archives on those tapes, and restoring the archived storage pool from those tapes.

2. Load the tape on the tape device.

   In each step below, the tape_device specified must be a nonrewinding device**.** On AIX, HP-UX, OSF/1, and Solaris systems, the tape device name must have the format

```
/dev/rmt/device_numbern
```

For example, `/dev/rmt/1n`

On SGI and SunOS systems, the tape device name must have the format

```
/dev/ntape_device_name
```

For example, `/dev/nrst18`

You must always include the **n** option to indicate a nonrewinding device when recovering a storage pool.

3. Read the first archive on a UBKUP tape by entering the command below.

   On any system, except Solaris, enter

   ```
   % cpio -icBt < tape_device_name
   ```

   On Solaris systems, enter

   ```
   % cpio -ict -B -I tape_device_name
   ```

   The system then lists the files in each pool.

4. To read any subsequent archives on the same tape, you must skip past the tape EOF mark with the UNIX `mt` command before invoking the `cpio` command again.

   On any system except HP-UX, enter

   ```
   % mt -f tape_device_name fsf 1
   ```

   On HP-UX systems, enter

   ```
   % mt -t tape_device_name fsf 1
   ```

5. Restore an archived storage pool from a UBKUP tape.

   Once the wanted pool's location on the tape has been determined, it can be restored. If the pool is not the first archive on the tape, use the UNIX mt command to position the tape. For example, if the wanted pool is the third archive on the tape, enter:

   ```
   % mt -f tape_device_name fsf 2 (use mt -t on HP-UX)
   ```

   which would space forward two tape marks to reach the third tape archive.

   The pool can then be restored to disk:

   ```
   cpio -icB < tape_device_name
   ```

6. After restoring the files with the UBKUP tapes, use the CHGSPS command to change the status of each pool to 6 (tagged for roll forward by RECSP).

7. Use the RECSP (recover storage pool) command to recover the contents of a destroyed or defective storage pool. The RECSP command can prompt you to load specific incremental backup tapes containing the most current versions of particular files.

   While you restore files from the incremental backup tapes, Vault can service user requests.

8. After running RECSP, change the status of each pool to 0 (available) or 2 (read only).

If you have accidentally deleted a file or corrupted the latest version of a file and want to recover an earlier version of it, you can use the recover single file command, RECSF.

This command brings back an old version of a file either to replace the current (corrupted) version or to reactivate a deleted file. The file's old version is recovered from an incremental backup tape or from the storage pool where it awaits incremental backup.

If you have deleted or corrupted an entire part and want to recover it, recover each file in the part individually.

Restoring an Archived Storage Pool from a UBKUP Tape on SunOS Machines

Once the wanted pool's location on the tape has been determined, it can be restored. If the pool is not the first archive on the tape, the tape must first be positioned using the mt command. For example, if the wanted pool is the second archive on the tape, enter:

```
% mt -f tape_device fsf 1
```

which would space forward one tape mark to reach the second tape archive.

The pool can then be restored to disk:

```
% tar XVF tape_device
```

For instance, to recover storage pool POOL4 from the previous example, mount the second UBKUP tape, and position it:

1. Enter `% mt -f /dev/nrst0 fsf 1` and then restore the pool.
2. Enter `% bar -xvf /dev/nrst0.` The following output appears:

```
X /EDMPOOLS/pool4/POOL4.EDMVAULT, 20 bytes, 1 tape blocks 1
X /EDMPOOLS/pool4/AAD95241.PDM, 24576 bytes, 48 tape blocks 1
X /EDMPOOLS/pool4/AAG95241.PDM, 16 bytes, 1 tape blocks 1
X /EDMPOOLS/pool4/AAH95241.PDM, 16 bytes, 1 tape blocks 1
X /EDMPOOLS/pool4/AAI95241.PDM, 16 bytes, 1 tape blocks 1
X /EDMPOOLS/pool4/AAM95241.PDM, 24576 bytes, 48 tape blocks 1
bar: Insert volume 2 and press Return when ready.
/*** rewound and unloaded first tape; second tape loaded; press
Return ***/
X /EDMPOOLS/pool4/AAN95241.PDM, 16 bytes, 1 tape blocks 1
X /EDMPOOLS/pool4/AAO95241.PDM, 24576 bytes, 48 tape blocks 1
%
```

Recovering Part Files

As you recover part files, they might or might not be automatically reattached to their original part. If the recovery is performed before the incremental backup, which deletes the old version, the files can be reattached to their part. If the entire part was deleted, it is made active again during the recovery process.

If the part is not recovered intact, you can reassemble it by reading out the orphaned files after they are recovered, purging them from the file directory, then storing them back using the add-file-to-part option on the `Store` command.

If an entire part is disassembled, read out all its files, purge them, then store the part back into Vault.

Command-Line Format Example

To recover a part with two files, enter the following:

```
%cirecsf pfname=GENERATOR.A date=11/07/93 time=15:37:11
revision=3
%cirecsf pfname=GENERATOR.&PD date=11/07/93 time=15:37:13
revision=3
```

where `GENERATOR` is the part that was accidentally deleted. This part was recovered from the storage pool because no `DELOV` command had been run. The two files in the part were automatically reattached to their part header.

Dropping and Recreating Indexes

When running Vault on a UNIX platform, you must occasionally drop and recreate indexes for Vault's Oracle control tables. This allows the indexes of these tables to continue to work efficiently.

If you notice a degradation in system performance (particularly if this occurs after many files, users, and/or projects have been added to the database), drop and recreate the table indexes. This should make Vault work more efficiently.

You should also drop and recreate indexes after deleting many rows from the file or part directories or the attribute data table. Run `edmindex` to drop and recreate the Vault control table indexes. Run `attrindx` to drop and recreate the attribute control table indexes.

Activating the E-mail Trigger

The e-mail trigger sends UNIX e-mail in addition to Vault messages when users execute the REQRVW (request review), RSVP (respond to review), or SENDMSG (send message) commands. UNIX e-mail is sent at the same time as the duplicate Vault message.

You must have previously loaded Projects and Programming to use this feature. You can turn this feature on or off during software installation or at any time after installation.

Turning On the E-mail Trigger

Follow the instructions below to activate the e-mail trigger. The results of this procedure are that any command triggers for the SENDMSG, REQRVW, and RSVP commands are changed to the e-mail trigger, and the command trigger list is activated.

1. Log in to the Vault account:

```
# rlogin system_name -l edm_account_name
```

2. Change to the installation directory:

```
# cd /$EPD_HOME/install
```

3. Initiate the Vault e-mail trigger activation module by entering the following:

```
# edmaetm -t y
```

Please note: The -t option in both cases stands for TRACE ON. If you do not require this option, enter a dummy variable as follows:

```
# edmaetm -f y
```

4. When prompted to indicate if you want to continue, enter yes or press RETURN. (Yes is the default.)

Turning Off the E-mail Trigger

You can also use the CHGCTL (change command trigger list) command to turn the e-mail trigger off. To deactivate the e-mail trigger, enter the following:

```
# edmaetm -t n
```

You can turn it off for the RSVP, REQRVW, and/or SENDMSG command. Specify the command name and N for the Active parameter. See *Vault Programmer Guide*.

Alias for UNIX E-mail Recipients

UNIX e-mail recipients must have an alias in the `/etc/aliases` file that maps their Vault user ID to the UNIX `/etc/passwd` file ID on the node that the triggered process executes on.

Changing Password for E-mail Triggers

To change the `edmadmin` password for email triggers, do the following:

1. Open the `nsm.config` file located in the `$EDM_HOME/data` directory.
2. Locate the email trigger information in this file.

   A sample of the information is as follows:

```
# E-Mail Command Trigger
   AE(COMMAND_TRIGGER,edmgrp,3)
     ALIAS(TRIGGER)
     PATH($EDM_HOME/trig/start-COMMAND-TRIGGER)
     OWNER(edm)
     WORKDIR(/tmp)
     CLOSE
     MAXINST(2)
     GRPCTL(1,1,0)
```

   In the above example, the PATH is `$EDM_HOME/trig/start-COMMAND-TRIGGER`.
3. Open the `start-COMMAND-TRIGGER` file and check for the following line:

```
$TRIG_DIR/dtrigger edmadmin edmadmin
```

   The second `edmadmin` is the email trigger password that you can change.
4. Change the email trigger password as required.

   If you cahnge it to `admin12`, the line will be as follows:

```
$TRIG_DIR/dtrigger edmadmin admin12
```

Existing RSVP, REQRVW, and SENDMSG Triggers

If you already have triggers for the RSVP, REQRVW, and/or SENDMSG commands, performing the steps on the previous page replaces your trigger for each of these commands. To set the trigger for one or two of the commands, do not perform those steps. Instead, execute the CHGCTL (change command trigger list) command.

When you execute the CHGCT command, first issue the CHGCTL command with ALL as the Vault command name and Active set to Yes. Then execute CHGCTL for REQRVW, SENDMSG, or RSVP with the values shown below.

- Active = Y
- Trigger at beginning = N Timeout = 0
- Trigger at end = Y Timeout = 0
- Application entity name = COMMAND_TRIGGER

You can also merge your trigger with the UNIX e-mail trigger. See *Vault Programmer Guide* for more information about the CHGCTL command and for information about command triggers.

Appending an Vault File to the UNIX E-mail

When users execute the SENDMSG command, and the e-mail trigger is activated, they can append an Vault file to the UNIX message. To do so they end the Vault message with

EDM file: EDMfilename

being sure to put a space between the colon and the file name.

The user ID signed on to Vault must have access to any appended files or the contents of these files cannot be included in the UNIX e-mail message.

Please note: Vault messages sent via UNIX mail use SQL servers. When the number of servers defined by maxinst are all in use, the Vault UNIX e-mail trigger does not work. No message is returned.

Changing Storage Pool Selection Logic

Chapter 3, “Changing Storage Pool Selection” provides the information you need to change storage pool selection. This section provides the details specific to UNIX.

Designing and Installing Custom Logic

When designing your own template for storage pool selection, the files you create must have the following names.

- FD1 is the name of the control file.
- FD2 contains the selection queries.
- FD3 contains the pool filters.

Chapter 8

Summary of Network Services Commands

This chapter contains a brief description of the commands associated with Vault Network Services.

- Overview
- nsmquery Command
- nsmflush Command
- nsmstop Command
- nsmstart Command

Overview

NSM commands can be issued with options and/or parameters. Optional parameters are enclosed in square brackets [ ].

So instead of entering `run nsmquery`, you enter `nsmquery -ae` with the required option. The partial and full ae_names should be in quotes, for example:

```
nsmquery -ae “blazer:pdm:ae1:0” or nsmquery -ae “:pdm:ae1:”
```

Please note: The NSM Configuration File now has up to 512 instances. Any application entity can have up to 512 instances within a domain. Specify the number of instances with the MAXINST parameter in the `nsm.config` file.

nsmquery Command

The `nsmquery` command queries the PM_BIND and DS_BIND Entity Trees of Directory Services.

The `nsmquery` command and its options are listed below.

`nsmquery -pm`

Writes all the pmbind table information to the screen. The pmbind table contains process name, protocol, port name, process state, and process error state.

`nsmquery -pm -f`

Writes all the pmbind table information to the `querypm.log` file, which is created in the directory from which the command is issued. The pmbind table contains process name, protocol, port name, process state and process error state.

`nsmquery -ds`

Writes all the dsbind table information to the screen. The dsbind table contains process name, maximum concurrences, number of connections, number of ports, protocol, and port names.

`nsmquery -ds -f`

Writes all the dsbind table information to the `queryds.log` file. The dsbind table contains process name, maximum concurrences, number of connections, number of ports, protocol, and port names.

`nsmquery -ae`

Writes all the pmbind and dsbind table information to the screen.

- pmbind — This table contains process name, protocol, port name, process state and process error state.
- dsbind — This table contains process name, maximum concurrences, number of connections, number of ports, protocol, and port names.

`nsmquery -ae -f`

Writes all the pmbind and dsbind table information to the `queryae.log` file.

- pmbind — This table contains process name, protocol, port name, process state and process error state.
- dsbind — This table contains process name, maximum concurrences, number of connections, number of ports, protocol, and port names.

`nsmquery -ae full_aename`

Writes all information concerning the named AE to the screen.

Example:

```
full_aename example - blazer:pdm:ae1:0
```

`nsmquery -ae partial_aename`

Writes all information concerning any AE whose name contains the partial AE name to the screen.

Example:

```
artial_aename examples -
:pdm:ae1:, ::ae1:, blazer:::, ae1, :pdm::,
The partial_aename, :pdm:ae1:, is part of the full_aenames,
blazer:pdm:ae1:0 and pandora:pdm:ae1:1.
```

`nsmquery -ae -f partial_aename or full_aename`

Writes all information concerning the named AE to the `queryae.log` file.

Example:

```
partial_aename examples -
:pdm:ae1:, ::ae1:, blazer:::, ae1, :pdm::,
full_aename example - blazer:pdm:ae1:0
```

`nsmquery -pca`: Displays a list of all running PCAs.

`nsmquery help or nsmquery h`

Online help is available by typing the command followed by the word help or the character h and pressing `RETURN.`

nsmflush Command

The `nsmflush` command deletes the entities currently recorded in the dynamic portions of Directory Services

The `nsmflush` command and its options are listed below.

Please note: This command should only be used by an authorized Vault administrator and only in extreme conditions in order to recover from an error condition.

`nsmflush -a`

Deletes all entries in the pmbind and dsbind tables.

- pmbind — This table contains process name, protocol, port name, process state and process error state.
- dsbind — This table contains process name, maximum concurrences, number of connections, number of ports, protocol and portnames.

`nsmflush full_aename`

Deletes all entries for a specified AE from the pmbind and dsbind tables.

Example: `full_aename example - blazer:pdm:ae1:0`
`nsmflush partial_aename`

Deletes information concerning the named AE from the pmbind and dsbind tables.

Example: `partial_aename example - nsmflush :pdm::`

This entry would cause all entries that have the domain pdm in their aename to be deleted from the pmbind and dsbind tables.

`nsmflush help or nsmflush h`

Online help is available by typing the command followed by the word help or the character h and pressing `RETURN.`

nsmstop Command

The `nsmstop` command stops specific processes that are running on the network. This command requires a single parameter.

The `nsmstop` command and its parameter formats are listed below.

`nsmstop` node_name:domain_name:ae_name:instance

Sends a stop signal to a specific AE instance. The optional parameters are used to define the specific AE ([node]:domain:[ae]:[instance]).

`nsmstop` :domain_name:

Sends a stop signal to all AEs in the specified domain that are running on the local node (the node from which the command was issued).

`nsmstop` :domain_name.start_group_name

Sends a stop signal to all AEs defined for the specified start_group within the specified domain, regardless of node.

`nsmstop` :domain_name

Sends a stop signal to all AEs defined for the specified domain, regardless of group or node.

`nsmstop` :.start_group_name

Sends a stop signal to all AEs defined for the specified start_group, regardless of node or domain.

`nsmstop :`

Sends a stop signal to all AEs. All domains and all groups is the default value when issued with no optional parameters specified:

nsmstop : [domain].[group]

`nsmstop` -pca

Sends a stop signal to all the Process Control Agents (PCAs).

`nsmstop` -pm

Sends a stop signal to the Process Manager (PM).

`nsmstop` -all

Sends a stop signal to all AEs and to the Process Manager (PM).

`nsmstop help or nsmstop h`

Online help is available by typing the command followed by the word help or the character h and pressing `RETURN`.

nsmstart Command

The `nsmstart` command starts specific processes that are running on the network. This command requires a single parameter.

The `nsmstart` command and its parameter formats are listed below.

`nsmstart` node_name:domain_name:ae_name:

Sends a start signal to the specified AE. Domain_name and ae_name are required (:domain_name:ae_name:). Instances are not allowed.

`nsmstart :`domain_name`.`start_group_name

Sends a start signal to all the AEs in the specified group within the specified domain, regardless of node.

`nsmstart` :domain_name

Sends a start signal to all the AEs in the specified domain, regardless of node.

`nsmstart` :.start_group_name

Sends a start signal to all the AEs in the specified group, regardless of node or domain.

`nsmstart help or nsmstart h`

Online help is available by typing the command followed by the word help or the character h and pressing `RETURN`.

Chapter 9

Editing the Vault Configuration Files

This chapter provides instructions to aid in the management of the most basic aspects of the Vault configuration files, including how to edit them to support a multiple vault installation.

- Editing the NSM Configuration File
- Editing the PM Configuration Files

Editing the NSM Configuration File

The installation tool edmcinstall sets up both the `nsm.config` and the `pm.config` files automatically. If the network configuration files at your site require adjustments, edit the NSM and PM configuration files as described in the next two sections.

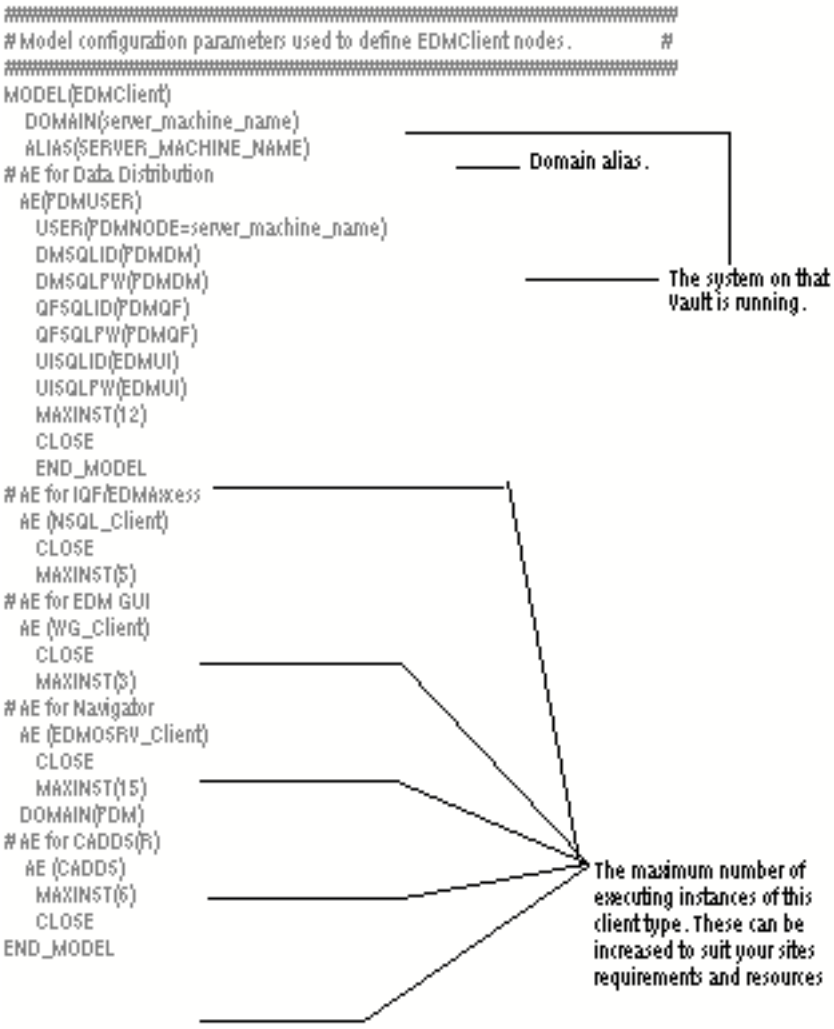
Description of the NSM Configuration File

The NSM configuration (nsm.config) file contains the node configuration of a Vault including key information such as port addresses, node names, a listing domains (such as Vault application domains and the domain of the Process Manager).

Add to NSM configuration file names of nodes which can access the Vault server (the machine where the Process Manager lives). Only those nodes recorded in the NSM file is visible to Vault.

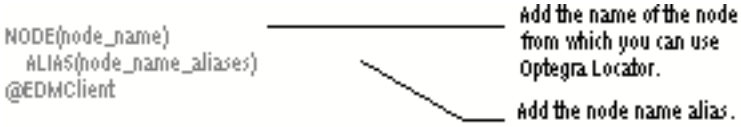
Please note: After you have edited the network configuration files, stop and restart your Vault processes. Otherwise, updated client nodes can not be recognized by Vault.

Figure 9-1 The EDMClient Model entry in the nsm.config



Please note: You can also edit the MAXINST value for the following servers: PDMDD, PDMADMN, ADMIN_SERVER, NSQL_Server, OAXIS, SQL_SERVER, QUERY_SERVER, DOD_QUERY_SERVER, and DIST_SERVER.

Figure 9-2 The EDMClient Node entry in the nsm.config



Please note: Verify that you have the correct NODE name and ALIAS name in the `nsm.config` template. Running `edmsnsm` during the installation session sometimes does not correctly revise these entries. This can cause startup failure.

Editing the PM Configuration Files

The PM configuration (pm.config) file specifies the network address of the Process Manager (PM) for each Vault. Each node must have a pm.config to provide this address. The RESOURCE statement in the pm.config maps to corresponding fields in the NSM configuration file. For example, to start up Vault in a Sun cluster environment, the pm.config file on the system where the Vault is started must have the following format:

```
RESOURCE(CLUSTER:::cluster:process_manager_domain:process_manager
_AE:0, udp, 9000 NODE (cluster)
```

Where CLUSTER or cluster is the name of the node.

Note that the Process Manager looks for the ANSPATH UNIX environment variable to find pm.config.

This section describes:

- The Process Manager's search order for the pm.config file.
- Using the systems pm.config file in $EPD_HOME/data/pm.config.
- Editing the pm.config_template to run Vault Client remotely from a single Vault.
- Editing the pm.config_template so that the Vault Client can access a number of Vaults.

Searching for the PM Configuration File

The search order for a pm.config file is:

1. A path defined by the ANSPATH UNIX environment and set in either the user's .cshrc or .login file:

   ```
   setenv ANSPATH /absolute_path/pm.config
   ```

2. A pm.config file in the user's home directory.
3. The default directory /usr/apl/edm/data

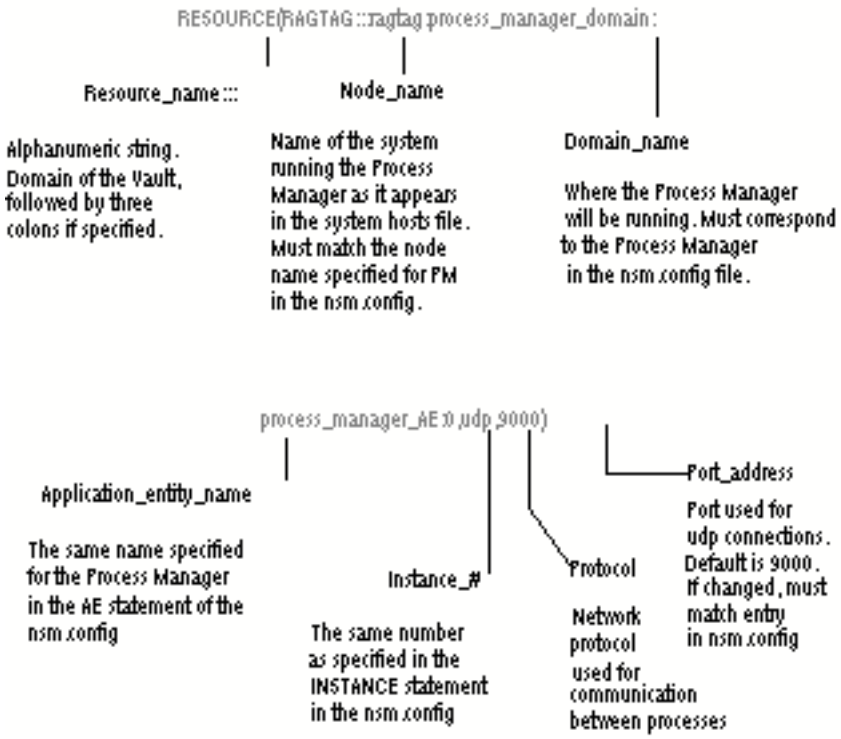
Ensure a valid copy of the pm.config file exists in one of these three locations.

Using the System's PM Configuration File

If the Vault Client is running on the same node as the Process Manager, use the system `pm.config` file**,** by adding this line to the user's `.login or .cshrc` file:

```
setenv ANSPATH $EPD_HOME/data/pm.config
```

Figure 9-3 A sample pm.config file



Please note: If you omit the vault name, the system assumes that this RESOURCE statement is the default. The vault name must be the first RESOURCE statement in the `pm.config` file. In this way, the `pm.config` file is upwardly compatible with the previous versions of Vault.

Editing the PM Configuration File for a Single Vault

Follow this steps to run Vault Client remotely from a single Vault:

1. Copy the template to your home directory:

   ```
   cp $EPD_HOME/install/pm.config-template ~/pm.config
   ```

2. Copy the Process Manager address from the NSM configuration file into your PM configuration files (on each node).

3. Place `pm.config` where it can be found by your Vault Client.

   ```
   setenv ANSPATH /absolute_path/pm.config
   ```

4. Set the permissions on the `pm.config` file:

   ```
   chmod 644 pm.config
   ```

The following is a sample `pm.config` file setup for a single Vault:

```
RESOURCE(JOHN:::john:process_manager_domain:process_manager_AE:0,
udp,9000
```

Editing the PM Configuration File

For Multiple Vaults: The syntax for the `pm.config` file supports multiple (serial) Vault access. You must include a `RESOURCE` statement in the `pm.config` file to identify each Vault to which a client can have access. For example, a distributed Vault needs multiple resource statements in the `nsm.config` file.

Follow this steps to run Vault Client remotely with access to more than one Vault:

1. Copy the template to your home directory:

   ```
   cp $EPD_HOME/install/pm.config-template ~/pm.config
   ```

2. Copy the Process Manager address from each NSM configuration file on each Vault, into your PM configuration file (on each node).

   A multiple Vault setup has multiple `RESOURCE` statements, one corresponding to each Vault.

3. Place `pm.config` where it can be found by your Vault Client.

   ```
   setenv ANSPATH /absolute_path/pm.config
   ```

4. Set the permissions on the `pm.config` file:

   ```
   chmod 644 pm.config
   ```

The following is a sample `pm.config` file setup for multiple Vaults:

```
RESOURCE(JOHN:::john:process_manager_domain:process_manager_AE:0,
udp,9000
RESOURCE(PAUL:::paul:process_manager_domain:process_manager_AE:0,
udp,9000
RESOURCE(JANE:::jane:process_manager_domain:process_manager_AE:0,
udp,9000
RESOURCE(JOAN:::joan:process_manager_domain:process_manager_AE:0,
udp,9000
```

This example shows the RESOURCE statement syntax expanded to enable the correlation of a Vault name to a Process Manager's AE address.

- The first parameter, separated by three colons (for example, `MATTHEW:::`) represents the RESOURCE name. This name represents an Optegra DOMAIN, as it is configured in the `nsm.config` file.
- The remaining portion of the RESOURCE statement is identical to previous releases of EDMVault, now known as Vault.

Please note: If you omit the Vault name, the system assumes this RESOURCE statement is the default. The Vault name must be the first RESOURCE statement in the `pm.config` file. In this way, the `pm.config` file is upwardly compatible with previous versions of Vault.

For Accessing Vault from a Different Domain: If you are running Vault and EPD.Connect from different network domains, you must specify the full name of the node in the `pm.config` file, that is the ANSPATH. The following is an example of a `pm.config` file in such a case:

```
RESOURCE(APHIDS:::aphids:process_manager_domain:process_manager_A
E:0, udp, aphids.usa.mycom.com, 9000)
```

Where, `aphids.usa.mycom.com` is the full node name.

In addition, set the environment variable EDMOANS to 1.

On the UNIX platform, set the environment as follows:

```
setenv EDMOANS 1
```

On the Windows NT platform, set the EDMOANS variable using Start > Settings > Control Panel > System > Environment.

Chapter 10

Vault Domain Management

This chapter provides information on Vault server processes and their function.

- Vault Servers
- Vault Distributed Object Directory Servers

Vault Servers

This section gives an overview of the standard vault server processes developed for Optegra with the intent of explaining their purpose and function.

Vault Domain Management (PMGR & PCA)

The collection of Vault client and server nodes, and processes is referred to as the Vault domain. Vault Domain Management coordinates the interaction between the various client and server processes within that domain. The running processes are the Process Manager and the Process Control Agent. The Process Manager authorizes access to the Vault and server nodes, and processes is referred to as the Vault domain. Vault Domain Management coordinates the interaction between the various client and server processes within that domain. The running processes are the Process Manager and the Process Control Agent. The Process Manager authorizes access to the Vault domain for any client or server process which wants to use the domain. It also coordinates startup and shutdown of the servers that run in the domain. It also assigns specific servers to clients when a client requests access to the domain.

The Process Control Agent is a server whose purpose is to start server processes within the Vault domain, at the instruction of the Process Manager.

Data Manager (PDMDM)

Vault must have one Data Manager server running at all times.

The Data Manager coordinates access to Vault data (files, parts, filesets, catalogs, and so forth). It has connections from each of the Data Distribution servers. The Data Manager serializes access to these data, thus ensuring that service requests can not collide when attempting to acquire or store a particular data item.

Data Distribution Server (PDMDD)

The Data Distribution Server processes client requests for files, parts, catalogs, and so forth. It provides file transfer services between the Vault domain and the client's file system. Client transactions such as `GET, REPLACE, STORE`, and `UPDATE` are directed to this server. The Data Distribution server has connection to the Data Manager, and uses the Data Manager to authorize access to the requested data. It also has connection to the Attribute Manager, where it can transfer attributes for association, classification, and storage/retrieval.

Each Data Distribution Server can support a single client connection at a time. A configurable maximum of 512 such servers is supported per Vault domain.

Administrative Server (PDMADMN)

The Administrative Server processes client requests for such administrative activities as adding or modifying user entries, creating filesets, administering projects, and so forth. It applies updates directly to the RDBMS. Each Administrative Server can support a single client connection at a time. A configurable maximum of 512 such servers is supported per Vault domain.

DV/Binder Administrative Server (ADMIN_SERVER)

The DV/Binder Administrative Server processes client requests relating to registration and subscription for the Optegra Distributed Vault option and for the administration of Optegra Binders.

Each DV/Binder Administrative Server can support a single client connection at a time. A configurable maximum of 512 such servers is supported per Vault domain.

Attribute Management Server (EDMATTR)

An Vault must have one Attribute Management Server running at all times. The Attribute Management Server is used for the storage and maintenance of customer-defined attributes associated to files and parts. It has connections from client processes for Optegra commands which are used to define attributes and attribute groups, and to modify file and part attributes.

It also has connections from the Data Distribution Server (PDMDD) for the purposes of classification, association, storage, and retrieval of the attributes of files and parts.

Remote Query Server (NSQL_Server)

The Remote Query Server supports the Vault remote IQF facility, and the VaultAxcess client interface. A maximum of 512 such servers are supported per Vault domain.

Export/Import Server (OAXIS)

The Export/Import Server is accessed by client processes to execute Distributed Vault `EXPORT` and `IMPORT` commands. This is a single-use server (it terminates itself after completion of the transaction) and serves one client connection. This server is also used in conjunction with the Import Manager Server (EDMIMGR) to enable automated Import operations.

A maximum of 512 such servers are supported per Vault domain.

Audit Logger (PDMLOG)

The Audit Logger writes transaction audit information to the Vault Audit Log table. It has connections from all other servers in the Vault domain, as well as connections from client processes.

There can only be one Audit Logger configured in an Vault domain, and it is optional.

Event Manager (EDMEMGR)

The Event Manager is an automated Distributed Vault server process which handles events related to files and parts that either have subscriptions applied to them explicitly or are registered in the Distributed Vault environment. The Event Manager can process these events into the appropriate actions, which the Action Manager (EDMAMAN) process can execute.

The Event Manager has no client connections. There can be only one Event Manager process configured per Vault domain. It is required by the Distributed Vault option.

Action Manager (EDMAMAN)

The Action Manager is an automated Distributed Vault server process which executes actions produced by the Event Manager. Actions executed by the Action Manager can include updating the Distributed Object Directory for registered files and parts, issuing an Vault `EXPORT` command, or running `SENDMSG` commands for files and parts which have active subscriptions.

The Action Manager has no client connections. There can be only one Action Manager process configured per Vault domain. It is required by the Distributed Vault option.

Import Manager (EDMIMGR)

The Import Manager is an automated Distributed Vault process which executes IMPORT operations to an Vault's peer distributed Vaults.

It has no client connections. Rather, it connects as a client, to an OAXIS server in its peer domains. The Import Manager process is optional, but highly recommended, and is supported by the Distributed Vault option only. There can be only one Import Manager process configured per Vault domain.

DataBase Update Server (SQL_SERVER)

The DataBase Update Server process is used by the Vault GUI and the Optegra Navigator for the purpose of applying updates to high level queries, conditions, and expressions to the RDBMS.

This server allows connection by only one client at a time, and a configurable maximum of 512 such servers are supported per Vault domain.

DataBase Query Server (QUERY_SERVER)

The DataBase Query Server is the query-only counterpart to the DataBase Update Server. It is a multi-user server, supporting up to 15 concurrent client connections. This server has connections from Vault GUI processes, as well as connections from Vault client commands which use Distributed Vault facilities.

A configurable maximum of 512 such servers are supported per Vault domain.

Vault Distributed Object Directory Servers

This section explains the purpose and function of the Vault Distributed Object Directory (DOD) server processes. The following servers reside on the Vault domain that is configured with a Distributed Object Directory.

Distributed Object Directory Query Server (DOD_QUERY_SERVER)

The Distributed Object Directory Query Server is a query-only server that accepts connections from Vault client processes using Distributed Vault features. It is also used by the Vault GUI, when the GUI is displaying the Distributed Files/Parts view.

This server supports up to 15 concurrent client connections. A configurable maximum of 512 such servers are supported per Vault domain.

Distributed Object Directory Distribution Server (DIST_SERVER)

The Distributed Object Directory Distribution Server is used by the Vault Action Manager (EDMAMAN) to apply inserts, updates, and deletions to the Distributed Object Directory, as they relate to registered files and parts that reside in the Vault. It accepts only a single client connection.

A configurable maximum of 512 of these servers are supported per Vault domain. As a practical matter, however, a minimum of one of these servers should be configured for each other Vault domain that can access the Distributed Object Directory.

Figure 10-1 Optegra Vault Servers

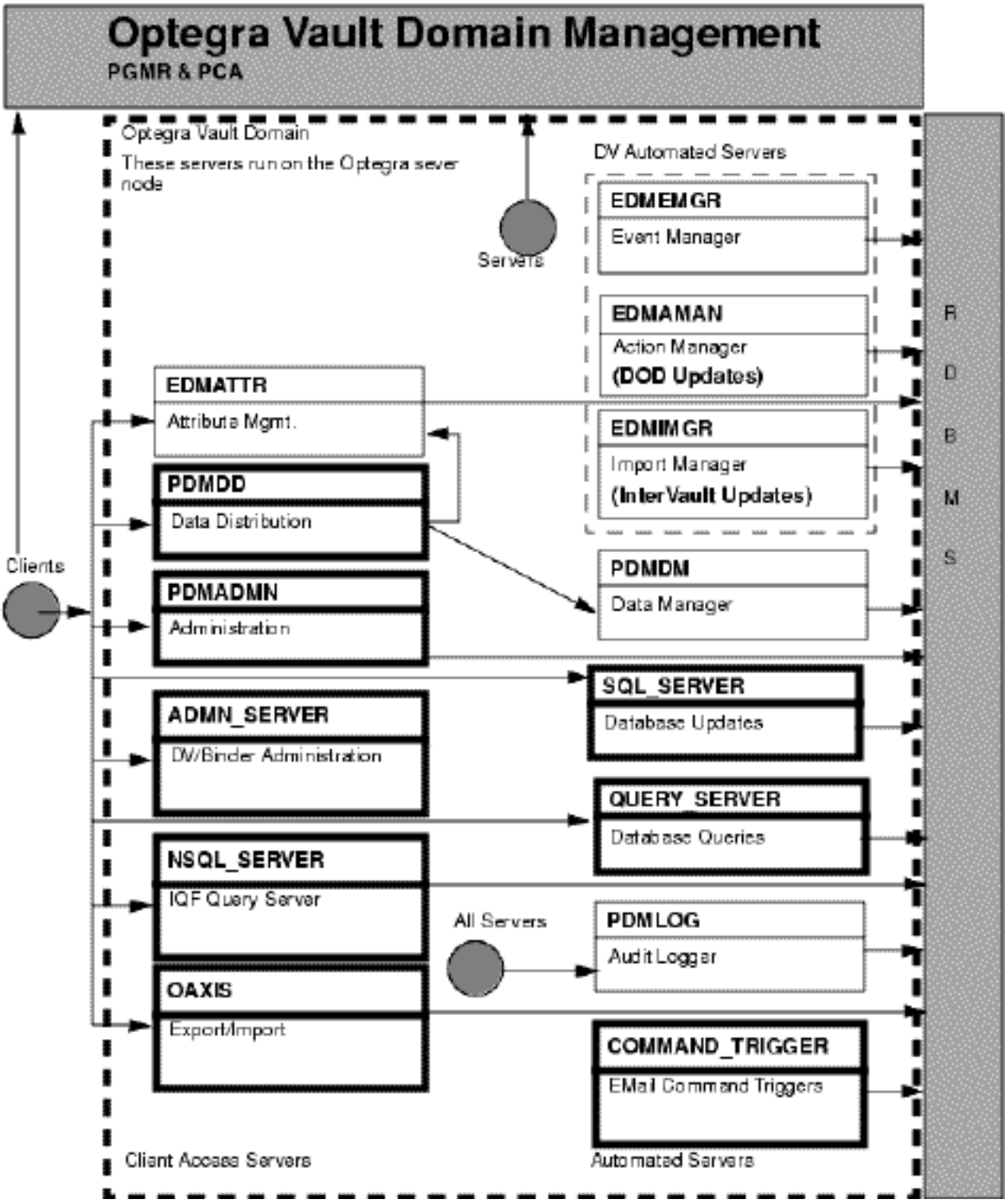


Figure 10-2 Vault DOD Servers

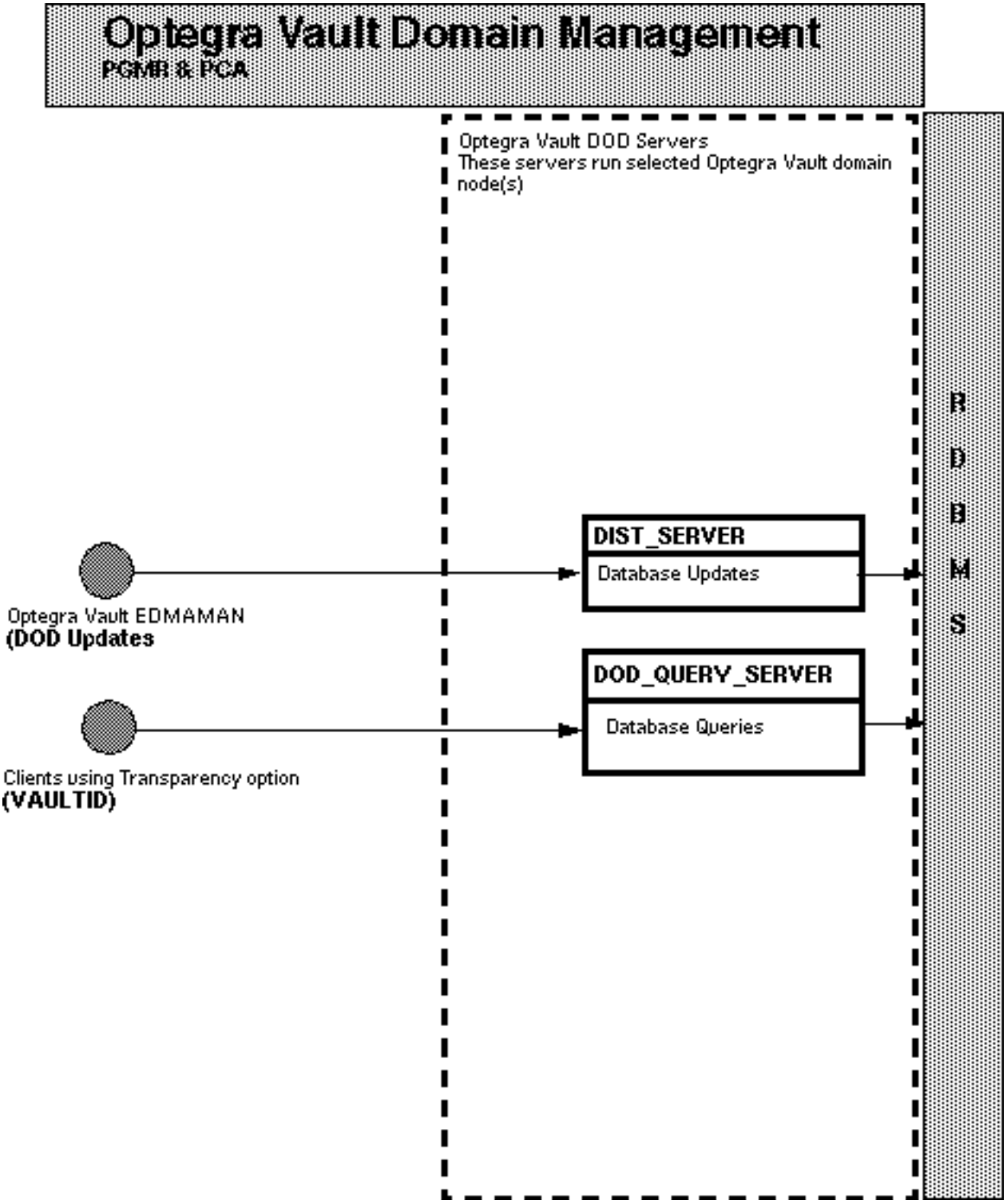
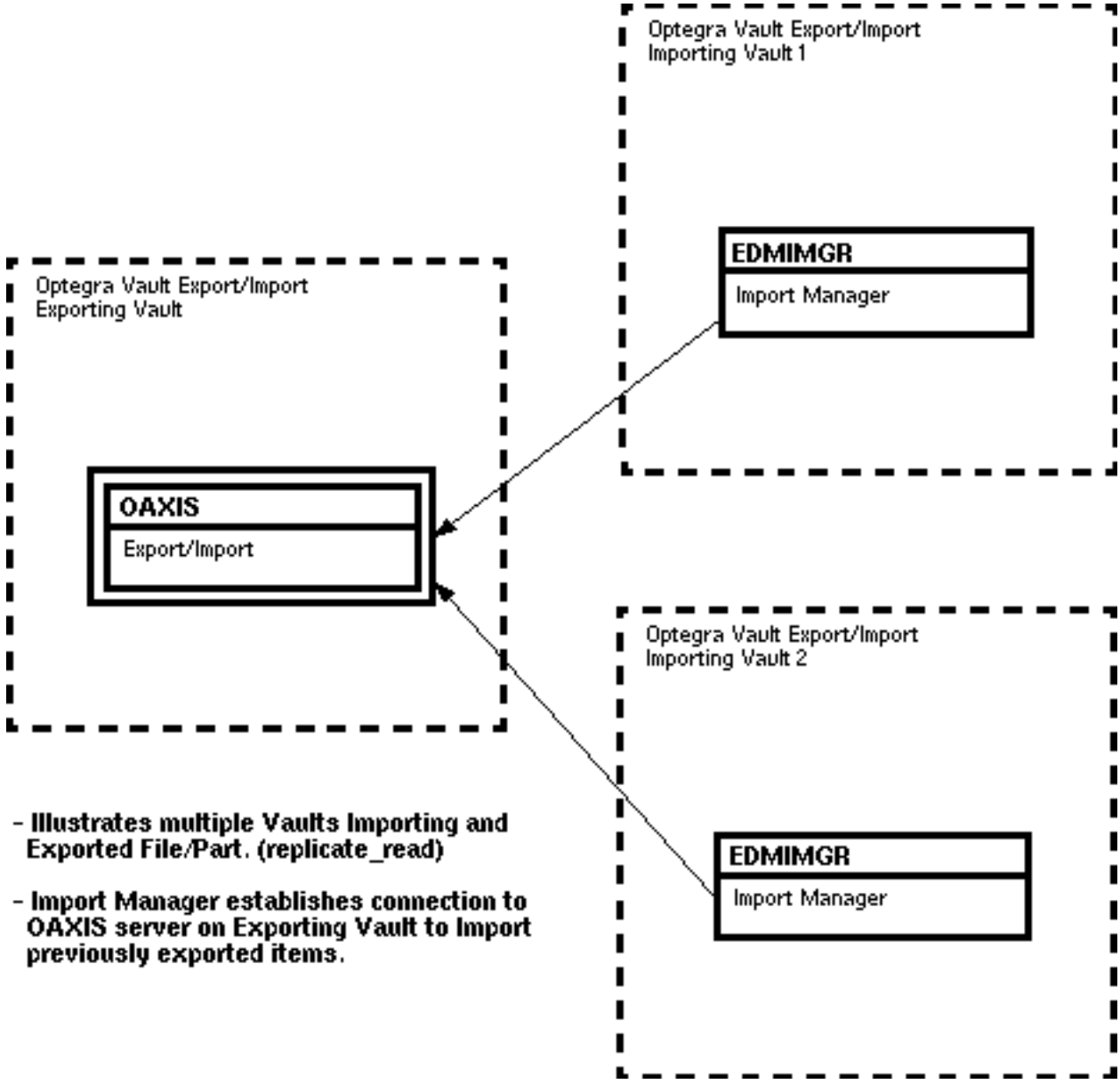


Figure 10-3 Vault Excanport/Import Servers

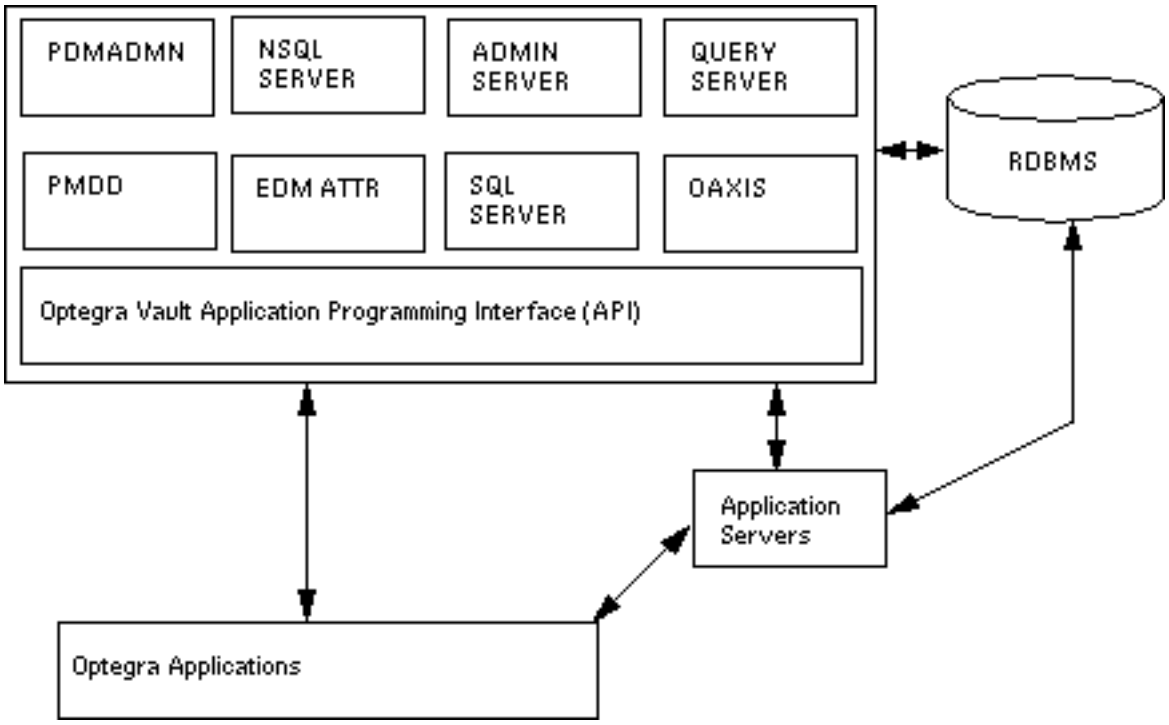


Application Servers

Application Servers provide additional methods for applications to access Optegra Vault functions and/or information stored in the Relational Database Management System (RDBMS). Applications use the Optegra Vault Application Programming Interface (API) to perform Vault functions. Some applications, however, can require capabilities not provided by the API or can operate in environments where the API is not available. In these cases, Application Servers ensure that the applications can communicate effectively with the Optegra Vault.

An application can perform Optegra Vault operations by accessing the Optegra Vault directly, by using an Application Server, or both. This depends on how the application operates, as illustrated below.

Figure 10-4 Optegra Vault operations



Desktop Server (DESKTOP_SERVER)

The Desktop Server (DESKTOP_SERVER) provides access to Optegra Vault functionality from personal computers and other systems where there is no direct interface to the Vault. The application communicates requests to the DESKTOP_SERVER, which executes them using the standard Vault application programming interfaces.

Desktop EDM Oracle Server (DESKTOP_EDMOSRV)

The Desktop EDM Oracle Server (DESKTOP_EDMOSRV) provides SQL operations to the Oracle RDBMS from personal computers and other systems where a direct link to the RDBMS is not available. SQL operations are communicated by the application to the DESKTOP_EDMOSRV, which executes them against the RDBMS.

Vault Command Server (VaultCmdServer)

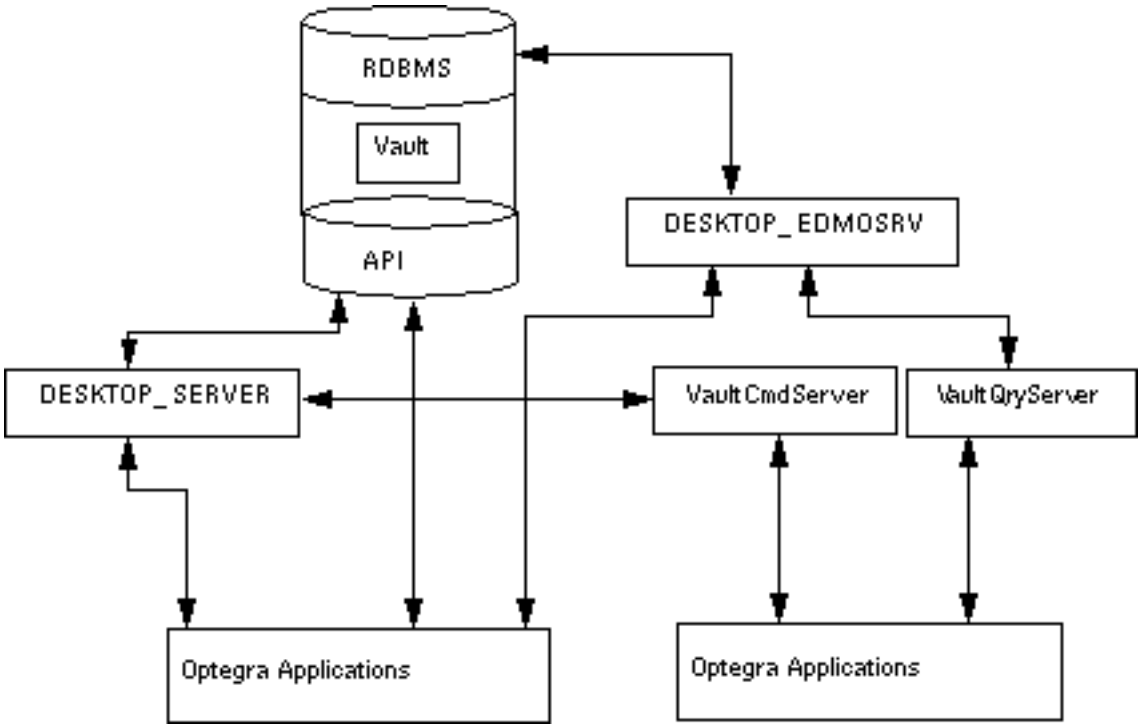
The Vault Command Server (VaultCmdServer) performs Optegra Vault operations from the EPD.Connect Java Information Browser. The VaultCmdServer communicates requests to Optegra directly using the Optegra Vault API and the Desktop Server. The VaultCmdServer is located on each user's computer.

Vault Query Server (VaultQryServer)

The Vault Command Server (VaultQryServer) performs Optegra Vault operations from the EPD.Connect Java Information Browser. The VaultQryServer communicates requests to the Desktop EDM Oracle Server for execution. The VaultQry Server can be located on each user's computer or on a centralized server to handle multiple requests.

The following illustration shows the interaction between Optegra applications, Application Servers, and the Vault.

Figure 10-5 Interaction between Optegra applications, Application Servers, and the Vault.



Chapter 11

Problem Solving

This chapter provides an overview of how to identify and resolve problems in an Optegra Vault environment. Topics presented include:

- Problem Solving Overview
- Checking for Required Software Components
- Using the Trace Utilities

Problem Solving Overview

An Optegra Vault installation can include a large number of computers and applications all working together. The complexity of the installation combined with the wide variety of capabilities that are provided by Optegra applications can make problem solving a challenging task.

Therefore, the key steps in resolving a problem are to:

1. Reproduce It—If a problem can be readily reproduced, its underlying cause can be more easily determined. If the problem cannot be reproduced, it is much more difficult to determine which solution, if any, resolved it.
2. Isolate It—If a problem can be isolated, it can be more easily identified and resolved.
3. Identify It—Once a problem has been isolated, it can be identified.
4. Resolve It—Apply appropriate solutions to the problem until it is resolved. A solution can require a change in the way the application(s) is used, a change to one or more configuration parameters, a software correction, or a change in the documentation that explains how to use the application.
5. Document It—Once a problem has been resolved, it is important to document the nature of the problem and the resolution that was applied, so that future problems that are similar can be more easily identified and resolved.

Checking for Required Software Components

When a problem occurs, one of the first things to check is whether all of the software components, or "Server Processes," needed to perform the required function are available and operating properly.

The following chart provides an overview of the type of processing performed by each component of an Optegra Vault. Use this chart to assist in identifying the software component source of problems.

Please note: Use the ps command (UNIX) or the Task Manager (Windows NT) to check for the process running on a specific system. Consult your operating system documentation for more information.

If a needed server process is not operating, start the process using the procedures described in *Installing Optegra Applications*.

Table 11-1 Processing Performed by Optegra Vault Components

Application		Server Process	Login	Admin	End User	Query	File Xfers	Attributes	Other
EPD.Connect									
	Main Application	PDMADMN	X						
	3-D Viewer	(none)							
	Information Browser	PDMDD			X		X	X	
		DESKTOP_ EDMOSRV				X			
		EDMATTR						X	
	Structure Browser	PDMADMN		X					
		PDMDD			X		X	X	
		EDMATTR						X	
		DESKTOP_ EDMOSRV	X		X	X		X	
	Visualizer/ Explorer	(none)							

Table 11-1 Processing Performed by Optegra Vault Components

Application		Server Process	Login	Admin	End User	Query	File Xfers	Attributes	Other
EPD.Connect Interfaces									
	CATIA Interface	PDMDD			X		X	X	
		DESKTOP_ EDMOSRV				X			
		EDMATTR						X	
	Pro/ENGINEER Interface	PDMDD			X		X	X	
		DESKTOP_ EDMOSRV				X			
		EDMATTR						X	
	MEDUSA Interface	PDMDD			X		X	X	
		DESKTOP_ EDMOSRV				X			
		EDMATTR						X	
	AP203	PDMDD			X		X	X	
		DESKTOP_ EDMOSRV				X			
		EDMATTR						X	
	CADDS/CAMU Integration	PDMDD			X		X	X	
		DESKTOP_ EDMOSRV				X			
		EDMATTR						X	
Locator PC		DESKTOP_ SERVER	X	X	X	X	X	X	Binders
		DESKTOP_ EDMOSRV				X			
Admin/PC		DESKTOP_ SERVER	X	X	X				Binders
		DESKTOP_ EDMOSRV				X			
EDMGUI		PDMDD	X		X		X		
		EDMATTR						X	
		PDMADMN		X					
		QUERY_ SERVER				X			GUI formula tion
Information Browser		VaultCmd Server	X		X		X	X	
		VaultQry Server				X			

When using this chart, keep the following definitions in mind:

- Login—All Optegra Vault functions require the user to log in to Optegra Vault. Therefore, users can not perform any Optegra Vault functions until they successfully log in to the system.
- Admin—Administrative functions include such tasks as creating users, adding users to groups, establishing authorization privileges, and manipulating how the system is set up.
- End User— End user functions are performed by a user who does not have administrative privileges. These functions include changing status codes, sending messages, reviewing change requests, voting, etc.
- Query—Query functions ask questions of the relational database management system (RDBMS) used by Optegra. Operations such as “find all parts checked out to me” are typical queries.
- File Xfers—File transfer functions involve the movement of one or more files between the user’s area and the secured storage area(s) of the Optegra Vault. These include “Read”, “Get”, “Update”, “Replace”, and “Store” operations.
- Attributes—Attribute functions are those that involve the manipulation of user-defined attributes. In addition, because attribute manipulations are performed on objects that have been “checked out,” there is a heavy interdependency between the file transfer and attribute server processes.

Because of the distributed nature of Vault applications, troubleshooting is often a matter of tracing the source of a specific problem. Different actions are required based on the problem source, so discovering this information is an important first step.

Using the Trace Utilities

This section describes how to set up trace files to diagnose problems with Vault and network services. Make sure you send the output of the trace utilities to a file, rather than to the screen (standard output). This way, you can keep this record and forward it to the Customer Service Center.

The utilities described in this section are internal utilities. They are provided to help you diagnose a Vault problem. It is expected that you can use them when talking to the Customer Service Center.

The following trace utilities are provided with the Vault:

- `ANSTLEVL` (for NSM problems)
- `NASCNFIG` (for network services problems)
- `CVTRACE` (for Vault problems)

You can use the `ANSTLEVL`, `NASCNFIG`, and `CVTRACE` utilities alone, together, or in any combination. You can set traces on any number of network processes, as well as on user accounts.

For example, if a user has a problem with a Vault administrative command (for example, `ADDU`, `ADMCOPY`, `CHGAG`, `DELUL`), you can set `ANSTLEVL` and `CVTRACE` on both the user account and the DM process. When the user runs the command again, you and the Customer Service Center can use the trace output to determine the problem.

The following are some general notes about the trace utilities:

- If you cannot determine whether a problem exists in network services or Vault code, set all three traces on a user account and/or the Vault network.
- Vault problems are rarely caused by the PCA or the LOG process. Unless you suspect that a problem exists with the PCA or the LOG process, you do not need to run traces on the PCA or LOG.
- Because no Vault code is involved in the Process Manager or the PCA, do not run `CVTRACE` on PMGR or PCA. Use `ANSTLEVL` and/or `NASCNFIG` to diagnose problems with the PMGR or PCA.

Tracing NSM Problems with ANSTLEVL

If you receive an NSM return code (these return codes begin with 20) and you cannot resolve the problem using the information on return codes, run a trace on NSM.

To run a trace on NSM, set an environment variable called ANSTLEVL with a numeric value that represents the type of NSM trace you want. Table 11-2 describes ANSTLEVL values. As shown in the table, this value can be the sum of the set of trace types you want. In general, it is recommended that you set ANSTLEVL with a value of 7.

You can run ANSTLEVL with NASCNFIG or CVTRACE. To do this, set the ANSTLEVL environment variable before you begin the NASCNFIG or CVTRACE utility.

Table 11-2 ANSTLEVL Values

ANSTLEVL	Meaning	What Is Traced
1	Exceptional conditions	This value is used to report errors that can occur during the operation of the process.
2	Major events	This value is used to report events of primary significance. For example, the message `ANS:Sending a PM_Bind Request for%s` is a major event in NSM processing.
4	Minor events.	This value provides additional information about NSM processing, supporting the major events.
7	All events traced by ANSTLEVL	Values 1, 2, and 4.
8	Low-level events	This value is often used by lower NSM layers of code to describe internal events of potential significance. This value is not recommended for general problem determination.
8	Low-level events	This value is often used by lower NSM layers of code to describe internal events of potential significance. This value is not recommended for general problem determination.
15	All events traced by ANSTLEVL	Values 1, 2, 4, and 8.
16	Entry/exit trace	This value traces entry and exit to the significant NSM processing routines. In many cases, exit return codes are displayed and can be useful for problem determination. This value produces a great deal of output and can be unnecessary for initial problem determination.
31	All events traced by ANSTLEV	Values 1, 2, 4, 8, and 16.
32	Performance statistics	This value traces the use of NSM message subsystem resources.
39	All events traced by ANSTLEVL	Values 1, 2, 4, and 32.

Where to Set Up the ANSTLEVL Environment Variable

If you have diagnosed a problem to a particular network process (Process Manager, PCA, LOG, DM, DN, or DD), you can set the ANSTLEVL environment variable on that process. You can also set ANSTLEVL on any number of network processes.

If you do not know which network process is causing a problem, you can set ANSTLEVL on the Vault account. This way, the trace file can contain all NSM activity. You can also set ANSTLEVL on an individual user account on a Vault Client node.

Setting Up Trace Utilities on UNIX Systems

On UNIX systems, use the following process to set up the trace utilities and save the output in a file.

1. Log on to the account on which you can set the trace. This can be a user account or the Vault account. If you are running a trace on a network process, log on to the edm account and change the directory to `/usr/apl/edm/scripts`.
   a. If you are running NSM trace, set the ANSTLEVL environment variable. Enter the word ANSTLEVL in uppercase.

      ```
      % setenv ANSTLEVL 7
      ```

   b. If you are running a network services trace, create or modify the `nascnfig.data` file. (The file name must be in lowercase.)
   c. If you are running CVTRACE, create or modify the `CVTRACE.PARAMS` file. The file name must be in uppercase.
2. Unless you are running only a network services trace and you set the TRACE_FILE keyword to FILE in the `nascnfig.data` file, set up a file to store the output from the trace:

   ```
   % script filename
   ```

3. If you are tracing one or more network processes and want to restart all of them, shut down and restart the Vault network by entering:

   ```
   % nsmstop -all
   % pmgr
   % pca
   ```

4. If you are tracing one or more network processes and want to restart only the network process(es) you are tracing, shut down and restart the Process Manager by entering:

   ```
   % nsmstop -all
   % pmgr
   ```

5. If you are tracing the DM, DN, and/or the LOG process and want to start only the one(s) you are tracing, now enter one or more of these commands:

```
% admcntl (for the DM)
% adncntl (for the DN)
% adllog (for the LOG)
```

6. If you are tracing the DD process and want to start only the DD, enter these commands. (The DM must be started to run the DD.)

```
% admcntl (for the DM)
% addcntl (for the DD)
```

7. If you logged on to a user or Optegra account, run the command(s) with which you have a problem.

8. Once you have captured errors or whenever you are ready to close the trace file, do one of the following:

   If you are running network services trace and you set the TRACE_FILE keyword to FILE in the `nascnfig.data` file, stop the network by entering:

```
% nsmstop -all
```

   If you used the `script` command to send trace output to a file, enter `CTRL-D`.

Setting Up Trace Utilities on NT

The following describes how to set up the trace utilities and save the output in a file on NT systems.

1. Log on to the account on which you can set the trace. This can be a user account or the Vault account. If you are running a trace on a network process, log on to the Vault account and change to the `$EPD_HOME\scripts` directory.

   a. If you are running NSM trace, set the ANSTLEVL environment variable. (Enter ANSTLEVL in uppercase.)

```
set ANSTLEVL=7
```

   b. If you are running a network services trace, create or modify the `nascnfig.data` file. (The file name must be in lowercase.)

   c. If you are running `CVTRACE`, create or modify the `CVTRACE.PARAMS` file. (The file name must be in uppercase.)

2. Unless you are running only a network services trace and you set the TRACE_FILE keyword to FILE in the `nascnfig.data` file, set up a file to store the output from the trace.

Tracing edmgui (EDMHLI_DEBUG)

To run a trace on edmgui,

1. Stop the currently running edmgui.
2. setenv DEBUG 1
3. Restart edmgui.

Now for every Vault command that you submit through the edmgui, you can view the equivalent Vault command string on your screen.

Tracing the edmosrv Server

1. Stop the currently running edmosrv.
2. setenv DEBUG 1
3. Restart edmosrv.

The results are displayed in the same window on your screen. For more details on edmosrv, refer to Customizing EPD.Connect.

Tracing the dtserver

To start tracing the dtserver, verify that the $EPD_HOME/data/dtconfig file contains the necessary settings for tracing parameters.

Tracing Parameters

The Optegra Vault Desktop Server logs all significant events in “server” and “client” log files in the /usr/tmp/dtserver_<vault_owner_account>l directory. More detailed logging can be turned on with the various tracing parameters.

Please note: Trace logging can impact both disk storage and system performance. In general, you can probably only want to set tracing parameters when suggested by support staff to gather detailed diagnostic information.

Trace Commands

[ON | OFF]: The Optegra Vault Desktop Server logs a brief description of each Vault command that is executed. Turning this parameter ON causes a hexadecimal byte-by-byte dump of the contents of each Vault command's input and output structures. The default is OFF.

Trace-recv-data [ON | OFF]: Turning this parameter ON causes a hexadecimal byte-by-byte dump of all server data as it is received on the TCP/IP connection from the PC.

Trace-send-data [ON | OFF]: Turning this parameter ON causes a hexadecimal byte-by-byte dump of all server data as it is transmitted on the TCP/IP connection to the PC.

Trace-Signed-Out [ON | OFF]: For certain Vault commands, the host server must query the Oracle database, in the process building a temporary list of currently signed out Vault entries. Turning this parameter ON causes the list to be printed to the “client” log file each time the list is built. The default is OFF.

Trace-Custom [ON | OFF]: Turning this parameter ON causes a hexadecimal byte-by-byte dump of the contents of “remote-custom” commands and their replies.The default is OFF.

The following is an example of a `usr/apl/edm/data/dtconfig` file:

```
Trace-commands   OFF
Trace-recv-data  OFF
Trace-send-data  OFF
Trace-Signed-Out OFF
Trace-Custom     OFF
```

Format of Server/Client Log Files

A server log file is

```
server.<server-processid>.<server-parentid>
```

A client log file is

```
client.<month-day-time>.<server-processid-forked>.<server-process
id>
```

The following is an example of server/client log files:

```
client.612102716.21317.21299  server.21299.1
client.612105002.23604.21299  server.25078.1
Client connection requested on 12 June, 10:27:16 on 21317 server.
[ first case ]
```

Sample Trace Output from the ANSTLEVL Utility

The following are two samples (UNIX, followed by NT) of the output you receive from the ANSTLEVL utility. In this example, ANSTLEVL was set with a value of 7. If you run ANSTLEVL with NASCNFIG and/or CVTRACE and send the output to the same file, you can see output from each trace interspersed in the file.

UNIX

```
ANS: Tracing enabled (trace level is 7) @ Thu Jun 11 12:57:46 1998
ANS:    Exceptional Conditions   ON (1)
ANS:              Major Events   ON (2)
ANS:              Minor Events   ON (4)
ANS:          Low Level Events  OFF (8)
ANS:          Entry/Exit Trace  OFF (16)
ANS:    Performance Statistics  OFF (32)
ANS:         Transaction Trace  OFF (64)
ANS:                Reserved_1  OFF (128)
ANS:                Reserved_2  OFF (256)
ANS:                Reserved_3  OFF (512)
ANS:                Terse Mode  OFF (1024)
ANS:              Verbose Mode  OFF (2048)
ANS:                 Timestamp  OFF (4096)
ANS:                Reserved_4  OFF (8192)
ANS:                Reserved_5  OFF (16384)
ANS:                Reserved_6  OFF (32768)
PMMS: Obtaining Process Manager Address
GETR: Trying Environment Variable(/usr/apl/edm/data/pm.config)
GETR: /usr/apl/edm/data/pm.config found via Environment Variable
NETWORK: $$NSM$BACKBONE$$,AE,AE,AF_INET Operational
NETWORK: $$NAS$BACKBONE$$,TCP,STCP,AF_INET Operational
NETWORK: $$NAS$LOCALNET$$,LOC,LOC,AF_UNIX Operational
PMMS: Starting up PMMS
PMMS: Establishing Channel for Default   RESOURCE
PAVAN:::pavan:process_manager_domain:process_manager_AE:0
PMMS: Opening AE Channel pavan:process_manager_domain:LOGON29646:0
ANS: RESOURCE
PAVAN:::pavan:process_manager_domain:process_manager_AE:0
Allocated to ::PDMUSER: by DEFAULT
ANS: Sending a PM_Bind Request for ::PDMUSER:
ANS: Waiting for a PM_Bind Response
ANS: Issuing pmms_wait for ACK to msgtype = 0
ANS: Issuing pmms_wait (interval is 0.00 seconds)
ANS: Receiving the PM_Bind Response
```

```
ANS: Process Management Version is V5.0.0.0
ANS: Opening the AE Channel
ANS: Opening the Control (O-O-B) Channel
ANS: Sending PM_Extract Request
ANS: Waiting for PM_Extract Response
ANS: Issuing pmms_wait for ACK to msgtype = 6
ANS: Issuing pmms_wait (interval is 0.00 seconds)
ANS: Reading PM_Extract Response
ANS: Sending the PM_Unbind Request
ANS: Waiting for the PM_Unbind Response
ANS: Issuing pmms_wait for ACK to msgtype = 4
ANS: Issuing pmms_wait (interval is 0.00 seconds)
ANS: Receiving the PM_Unbind Response
ANS: Closing the AE Channel
ANS: Stopping the Message subsystem
CDMINI072I 12:57:46 initializing storage pool POOL1.
CDMINI072I 12:57:46 initializing storage pool POOL2.
CDMINI455I The EDM initialization process has completed
successfully.
```

NT

```
ANS: Tracing enabled (trace level is 7) @ Thu Jun 11 12:13:23 1998
ANS:    Exceptional Conditions   ON (1)
ANS:               Major Events   ON (2)
ANS:               Minor Events   ON (4)
ANS:          Low Level Events  OFF (8)
ANS:          Entry/Exit Trace  OFF (16)
ANS:    Performance Statistics  OFF (32)
ANS:         Transaction Trace  OFF (64)
ANS:                 Reserved_1  OFF (128)
ANS:                 Reserved_2  OFF (256)
ANS:                 Reserved_3  OFF (512)
ANS:                 Terse Mode  OFF (1024)
ANS:               Verbose Mode  OFF (2048)
ANS:                  Timestamp  OFF (4096)
ANS:                 Reserved_4  OFF (8192)
ANS:                 Reserved_5  OFF (16384)
ANS:                 Reserved_6  OFF (32768)
PMMS: Obtaining Process Manager Address
GETR: Trying Environment Variable(D:\EPD\dm\v220\data\pm.config)
GETR: D:\EPD\dm\v220\data\pm.config found via Environment Variable
NETWORK: $$NSM$BACKBONE$$,AE,AE,AF_INET Operational
NETWORK: $$NAS$BACKBONE$$,TCP,STCP,AF_INET Operational
NETWORK: $$NAS$LOCALNET$$,LOC,LOC,AF_UNIX Inoperative, Does not
respond
PMMS: Starting up PMMS
PMMS: Establishing Channel for Default   RESOURCE
DHUMKETU:::dhumketu:process_manager_domain:process_manager_AE:0
PMMS: Opening AE Channel
dhumketu:process_manager_domain:LOGON501:0
PMMS: Establishing Channel for Secondary RESOURCE
AMEY:::amey:process_manager_domain:process_manager_AE:0
```

```
ANS: RESOURCE
DHUMKETU:::dhumketu:process_manager_domain:process_manager_AE:0
Allocated to ::PDMUSER: by DEFAULT
ANS: Sending a PM_Bind Request for ::PDMUSER:
ANS: Waiting for a PM_Bind Response
ANS: Issuing pmms_wait for ACK to msgtype = 0
ANS: Issuing pmms_wait (interval is 0.00 seconds)
ANS: Receiving the PM_Bind Response
ANS: Process Management Version is V5.0.0.0
ANS: Opening the AE Channel
ANS: Opening the Control (O-O-B) Channel
ANS: Sending PM_Extract Request
ANS: Waiting for PM_Extract Response
ANS: Issuing pmms_wait for ACK to msgtype = 6
ANS: Issuing pmms_wait (interval is 0.00 seconds)
ANS: Reading PM_Extract Response
ANS: Sending the PM_Unbind Request
ANS: Waiting for the PM_Unbind Response
ANS: Issuing pmms_wait for ACK to msgtype = 4
ANS: Issuing pmms_wait (interval is 0.00 seconds)
ANS: Receiving the PM_Unbind Response
ANS: Closing the AE Channel
ANS: Stopping the Message subsystem
CDMINI072I 12:13:23 initializing storage pool POOL1.
CDMINI072I 12:13:23 initializing storage pool POOL2.
CDMINI455I The EDM initialization process has completed
successfully.
```

Tracing Network Services Problems with NASCNFIG

If you receive a network services return code (these return codes begin with 25) and you can not resolve the problem using the information on the return code, run a trace on network services.

To run a trace on network services, create a file called `nascnfig.data`. In the file, enter the following keywords with one of the values shown in parentheses:

- TRACE (INFO or ERROR)
- TRACE_COMPONENT (ALL, DRIVER, FILE_XFER, NETWORK, MESSAGES, PROTOCOL, SCHEDULER, or UTILITIES)
- TRACE_DEST (FILE or TERM)
- TRACE_FILE filename

It is recommended that you set up the nascnfig.data file as follows:

- Set the trace mode to ERROR.
- Set the trace component to ALL.

You can define a file to capture the output of the NASCNFIG trace as follows:

- Set the trace destination to FILE.
- Set the trace file name to any name you want.

Please note: When you run network services tracing on a network process and save network services trace output to a file with the TRACE_DEST FILE keyword, this file is overwritten each time the network process you are tracing is started up.

You can also set TRACE_DEST to TERM. If you do this, send the output from NASCNFIG to a file with an operating system command.

The following is an example of a nascnfig.data file:

```
TRACE ERROR
TRACE_COMPONENT ALL
TRACE_DEST FILE
TRACE_FILE NETWORK.TRACE
```

Running NASCNFIG with ANSTLEVL or CVTRACE

You can run NASCNFIG with ANSTLEVL and/or CVTRACE. To do this, set the ANSTLEVL environment variable before you begin the NASCNFIG and CVTRACE traces.

To capture the output from NASCNFIG, ANSTLEVL, and CVTRACE in the same file, do one of the following:

- Use operating system commands to send the output from ANSTLEVL and/or CVTRACE to the file name defined in the nascnfig.data file with the TRACE_FILE keyword.
- In the nascnfig.data file, set the TRACE_FILE keyword to TERM and use an operating system command to send the output from NASCNFIG, CVTRACE, and/or ANSTLEVL to a file.

If you run NASCNFIG with ANSTLEVL and/or CVTRACE and send the output to the same file, you can see output from each trace interspersed in the file.

Placing the NASCNFIG.DATA File

If you have diagnosed a problem to a particular network process (Process Manager, PCA, DM, DN, or DD), you can run the network services trace on that process. You can also run a network services trace on any number of network processes.

If you do not know which network process is causing a problem, you can place the `nascnfig.data` file under the edm account. In this way, the trace file can contain all network services activity. You can also place the `nascnfig.data` file under an individual user account on an Vault Client node.

Sample Output from the NASCNFIG Utility

The following are two samples (UNIX, followed by NT) of the output you receive from the NASCNFIG utility. If you run NASCNFIG with ANSTLEVL and/or CVTRACE and send the output to the same file, you can see output from each trace interspersed in the file.

UNIX

```
CDMSON000I

###UTILITIES anapmgmt : Entering with sq_ptr -> SQ_event_code 1###
!!!UTILITIES anapmgmt : Scheduler Queue Element
000000 00000000 00000000 00000000 00000000  *................*
000010 00000000 00000000 ef638ea8 efff9d7c  *.........c.....|*
000020 00000001 00000000 00000000 00000000  *................*
000030 00000000 00000000 00000000 00000001  *................*
000040 00200000 efffb2b0 efff9f94 efffb3b8  *. ..............*
000050 00000000 efffb2b0 00000000 00000000  *................*
000060 00000000 00000000 00000000 00000000  *................*
###NETWORK initialize_drivers: Entering with 0 0###
###NETWORK nsm_typename: Entering with net 7###
###NETWORK nsm_typename: Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###NETWORK nsm_typename: Entering with net 1###
###NETWORK nsm_typename: Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###NETWORK initialize_drivers: Exiting with any_functional 2###
###UTILITIES anapmgmt: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler: Scheduler entered with event 5 on conid 147
###SCHEDULER gather_work: Entering with HOS 0###
###SCHEDULER gather_work: Exiting with NO_EVENT_PENDING 25024###
###SCHEDULER dispatch_work : Entering with SQHOS 2240152###
```

```
SCHEDULER dispatch_work: Scheduler Dispatching Event 5 for Conid 0
MACRO print_sq: user connection ID 0
MACRO print_sq: event
MACRO print_sq: initiate request
###PROTOCOL protfsm: Entering with sq_p->SQ_event_code 5###
###PROTOCOL setup_connect: Entering with con_p->ccb_id 1###
###UTILITIES bldpdu: Entering with sq_element->SQ_event_code 5###
###UTILITIES anaintpd: Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu: Exiting with status 0###
###PROTOCOL verify_locals: Entering with 0 0###
###PROTOCOL verify_locals: Exiting with SUCCESSFUL_COMPLETION 0###
###NETWORK nsm_typename: Entering with net 7###
###NETWORK nsm_typename: Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###PROTOCOL parse_addr: Entering with netp->nas_type 7###
###PROTOCOL parse_addr: Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL do_connect: Exiting with ret_code 0###
###PROTOCOL setup_connect: Exiting with se_p->se_return 0###
###PROTOCOL protfsm: Exiting with ret_code 0###
SCHEDULER dispatch_work: Dispatcher processing complete for event
5 Conid 1

SCHEDULER dispatch_work:Scheduler Dispatching Event 12 for Conid 1
MACRO print_sq: user connection ID 1
MACRO print_sq: event
MACRO print_sq: positive event
###PROTOCOL protfsm: Entering with sq_p->SQ_event_code 12###
###PROTOCOL send_init_pdu: Entering with con_p->ccb_id 1###
###NETWORK logical_write: Entering with con_p->ccb_id 1###
###NETWORK net_write : Entering with con_p->ccb_id 1##
###NETWORK Check_PCI : Entering with dir 111##
###NETWORK Check_PCI : Exiting with pdu_type 101###
NETWORK Poll_Streams : table count: 1 poll count: 1
###NETWORK net_write : Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###PROTOCOL initiate_reads : Entering with con_p->ccb_id 1###
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK logical_read : Exiting with ret_code 25056###
###PROTOCOL initiate_reads : Exiting with ret_code 0###
###PROTOCOL send_init_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm: Exiting with ret_code 0###
SCHEDULER dispatch_work: Dispatcher processing complete for event
12 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler: Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work: Entering with HOS 2347752###
SCHEDULER gather_work: CLEARING ECB 2 for CONID 1
```

```
SCHEDULER gather_work: Status Code 25056 Returned by SERVICE
SPECIFIC EXIT
###NETWORK net_write: Entering with con_p->ccb_id 1###
###NETWORK net_write: Exiting with ret_code 0###
SCHEDULER gather_work: Status Code 0 Returned by NETWORK WRITE
ROUTINE
###PROTOCOL write_handler: Entering with con_p->ccb_id 1###
###PROTOCOL write_handler: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work: Status Code 0 Returned by NETWORK WRITE
EXIT
###NETWORK net_read: Entering with con_p->ccb_id 1###
###NETWORK net_read: Exiting with ret_code 0###
###NETWORK convert_pdu_to_event: Entering with con_p->ccb_id 1###
###NETWORK Check_PCI: Entering with dir 105###
###NETWORK Check_PCI: Exiting with pdu_type 102###
###NETWORK convert_pdu_to_event: Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work: Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work: Entering with SQHOS 2395864###

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 0###
SCHEDULER dispatch_work: Dispatcher processing complete for event
122 Conid 1

SCHEDULER dispatch_work:Scheduler Dispatching Event 102 for Conid
1

MACRO print_sq: user connection ID 1
MACRO print_sq: event
MACRO print_sq: initiate response pdu
###PROTOCOL protfsm: Entering with sq_p->SQ_event_code 102###
###PROTOCOL split_pdu: Entering with 0 0###
###UTILITIES analzpdu: Entering with 0 0###
###UTILITIES analzpdu: Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL split_pdu: Exiting with se_p->se_return 0###
###PROTOCOL protfsm: Exiting with ret_code 0###
SCHEDULER dispatch_work: Dispatcher processing complete for event
301 Conid 1

SCHEDULER dispatch_work:Scheduler Dispatching Event 122 for Conid
1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
```

```
###PROTOCOL protfsm: Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm: Exiting with se_p->se_return 100###
SCHEDULER dispatch_work: Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
SCHEDULER scheduler: returning 301
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler: Scheduler entered with event 22 on conid 1
###SCHEDULER gather_work: Entering with HOS 2347752###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work: Entering with SQHOS 2381120###

SCHEDULER dispatch_work:Scheduler Dispatching Event 22 for Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : consume event
###UTILITIES consume : Entering with sq_element->SQ_conid 1###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###UTILITIES consume : Exiting with status 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler: Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work: Entering with HOS 2347752###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler: Scheduler entered with event 20 on conid 1
###SCHEDULER gather_work: Entering with HOS 2347752###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work: Entering with SQHOS 2381120###
```

```
SCHEDULER dispatch_work:Scheduler Dispatching Event 20 for Conid 1

MACRO print_sq: user connection ID 1
MACRO print_sq: event
MACRO print_s: message request
###MESSAGE anasdmsg: Entering with sq_ptr->SQ_conid 1###
###UTILITIES bldpdu: Entering with sq_element->SQ_event_code 20###
###UTILITIES anaintpd: Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu: Exiting with status 0###
###NETWORK logical_write: Entering with con_p->ccb_id 1###
###NETWORK net_write: Entering with con_p->ccb_id 1###
###NETWORK Check_PCI: Entering with dir 111###
###NETWORK Check_PCI: Exiting with pdu_type 114###
###NETWORK net_write: Exiting with ret_code 0###
###NETWORK logical_write: Exiting with ret_code 25056###
###MESSAGE anasdmsg: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER dispatch_work: Dispatcher processing complete for event
20 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
NETWORK Poll_Streams: table count: 1 poll count: 0
###SCHEDULER gather_work: Entering with HOS 2347752###
###NETWORK net_write: Entering with con_p->ccb_id 1###
###NETWORK net_write: Exiting with ret_code 0###
SCHEDULER gather_work: Status Code 0 Returned by NETWORK WRITE
ROUTINE
###MESSAGE message_write_handler: Entering with 0 0###
###MESSAGE message_write_handler: Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work: Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2381120###

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###MESSAGE anasdmsg : Entering with sq_ptr->SQ_conid 1###
###MESSAGE anasdmsg : Exiting with PROCESS_COMPLETE 100###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work:Exiting with current_request_status
0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work: Entering with HOS 2347752###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
```

```
SCHEDULER scheduler: Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work: Entering with HOS 2347752###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
NETWORK Poll_Streams : table count: 1 poll count: 0
NETWORK Poll_Streams : table count: 1 poll count: 1
NETWORK naswait - poll : no event on any fd: error 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event: Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 201###
###NETWORK convert_pdu_to_event: Exiting with
SUCCESSFUL_COMPLETION 0##
SCHEDULER gather_work Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work: Entering with SQHOS 2381120###

SCHEDULER dispatch_work:Scheduler Dispatching Event 201 for Conid
1

MACRO print_sq: user connection ID 1
MACRO print_sq: event
MACRO print_sq: send file request pdu
###FILE_XFER ==>> anaftsrv: Entering with 0 0###
FILE_XFER anaftsrv: starting sendfile PDU
###FILE_XFER ==>> fts_init_request: Entering with 0 0###
###FILE_XFER ==>> validate_request: Entering with 0 0###
###FILE_XFER f_erase: Entering with 0 0###
###FILE_XFER f_erase: Exiting with fpb_ptr->file_return 0###
###FILE_XFER ==>> validate_request: Exiting with
SUCCESSFUL_COMPLETION 0###
###FILE_XFER ==>>anlz_fts_pdu: Exiting with status 0###
###FILE_XFER f_open: Entering with 0 0###
###FILE_XFER f_open: Exiting with fpb_ptr->file_return 0###
###FILE_XFER ==>> fts_init_request: Exiting with 0 0###
###FILE_XFER get_input: Entering with sq_p->SQ_event_code 201###
###FILE_XFER get_input: Exiting with out 1###
###FILE_XFER ==>> ftsfsms: Entering with 0 0###
###FILE_XFER ==>> act002: Entering with 0 0###
###FILE_XFER ==>> fts_bld_pdu: Entering with 0 0###
###UTILITIES anaintpd: Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd: Exiting with SUCCESSFUL_COMPLETION 0###
###FILE_XFER ==>>fts_bld_pdu: Exiting with SUCCESSFUL_COMPLETION
0###
###NETWORK logical_write: Entering with con_p->ccb_id 1###
###NETWORK net_write: Entering with con_p->ccb_id 1###
###NETWORK Check_PCI: Entering with dir 111###
###NETWORK Check_PCI: Exiting with pdu_type 202###
```

```
###NETWORK net_write: Exiting with ret_code 0###
###NETWORK logical_write: Exiting with ret_code 25056###
###FILE_XFER ==>> act002: Exiting with ret_code 0###
FILE_XFER Ftsfsm : transition from state 0 to state 3 for
condition 1
###FILE_XFER ==>> ftsfsms: Exiting with status 0###
###FILE_XFER anaftsrv: Exiting with service_code 0###
SCHEDULER dispatch_work: Dispatcher processing complete for event
202 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
NETWORK Poll_Streams : table count: 1 poll count: 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###NETWORK net_write : Entering with con_p->ccb_id 1###
###NETWORK net_write : Exiting with ret_code 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTINE
###FILE_XFER ==>> fts_handler : Entering with
fpb_p->fts_async_status 4###
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###FILE_XFER ==>> fts_handler : Exiting with 0 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
NETWORK Poll_Streams : table count: 1 poll count: 0
NETWORK Poll_Streams : table count: 1 poll count: 1
NETWORK naswait - poll : no event on any fd: error 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 205###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2381120###

SCHEDULER dispatch_work : Scheduler Dispatching Event 205 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : data
###FILE_XFER ==>> anaftsrv : Entering with 0 0###
###FILE_XFER get_input : Entering with sq_p->SQ_event_code 205###
###FILE_XFER get_input : Exiting with out 4###
###FILE_XFER ==>> ftsfsms : Entering with 0 0###
###FILE_XFER ==>> act005 : Entering with 0 0###
```

```
###FILE_XFER ==>> validate_request : Entering with 0 0###
###FILE_XFER ==>> validate_request : Exiting with
SUCCESSFUL_COMPLETION 0###
###FILE_XFER ==>>anlz_fts_pdu : Exiting with status 0###
###FILE_XFER f_write : Entering with 0 0###
###FILE_XFER f_write : Exiting with fpb_ptr->file_return 0###
###FILE_XFER ==>> act005 : Exiting with 0 0###
FILE_XFER Ftsfsms : transition from state 3 to state 3 for
condition 4
###FILE_XFER ==>> ftsfsms : Exiting with status 0###
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###FILE_XFER anaftsrv : Exiting with service_code 0###SCHEDULER
dispatch_work:Dispatcher processing complete for event 205 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
NETWORK Poll_Streams : table count: 1 poll count: 0
NETWORK Poll_Streams : table count: 1 poll count: 1
NETWORK naswait - poll : no event on any fd: error 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 206###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2381120###

SCHEDULER dispatch_work : Scheduler Dispatching Event 206 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : file end of data
###FILE_XFER ==>> anaftsrv : Entering with 0 0###
###FILE_XFER get_input : Entering with sq_p->SQ_event_code 206###
###FILE_XFER get_input : Exiting with out 5###
###FILE_XFER ==>> ftsfsms : Entering with 0 0###
###FILE_XFER ==>> act006 : Entering with 0 0###
###FILE_XFER ==>>anlz_fts_pdu : Exiting with status 0###
###FILE_XFER f_close : Entering with 0 0###
###FILE_XFER f_close : Exiting with fpb_ptr->file_return 0###
###FILE_XFER ==>> fts_bld_pdu : Entering with 0 0###
###UTILITIES anaintpd : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd : Exiting with SUCCESSFUL_COMPLETION 0###
###FILE_XFER ==>>fts_bld_pdu : Exiting with SUCCESSFUL_COMPLETION
0###
###NETWORK net_write : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 111###
```

```
###NETWORK Check_PCI : Exiting with pdu_type 207###
###NETWORK net_write : Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###FILE_XFER ==>> act006 : Exiting with 0 0###
FILE_XFER Ftsfsms : transition from state 3 to state 0 for
condition 5
###FILE_XFER ==>> ftsfsms: Exiting with status 0###
###FILE_XFER anaftsrv : Exiting with service_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
207 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
NETWORK Poll_Streams: table count: 1 poll count: 0
###SCHEDULER gather_work: Entering with HOS 2347752######NETWORK
net_write : Entering with con_p->ccb_id 1###
###NETWORK net_write : Exiting with ret_code 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTIN
###FILE_XFER ==>> fts_handler : Entering with
fpb_p->fts_async_status 6###
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###FILE_XFER ==>> fts_handler : Exiting with 0 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2381120###

SCHEDULER dispatch_work : Scheduler Dispatching Event 121 for
Conid 1

MACRO print_sq : user connection ID
MACRO print_sq : event
MACRO print_sq : unknown event 121
###FILE_XFER ==>> anaftsrv : Entering with 0 0###
###FILE_XFER get_input : Entering with sq_p->SQ_event_code 121###
###FILE_XFER get_input : Exiting with out 18###
###FILE_XFER ==>> ftsfsms : Entering with 0 0###
###FILE_XFER ==>> act015 : Entering with 0 0###
FILE_XFER Ftsfsms : transition from state 0 to state 0 for
condition 18
###FILE_XFER ==>> ftsfsms : Exiting with status 1###
FILE_XFER anaftsrv : FTS==>> S/R file operation ending for ....
          source file  =
          dest file    = /usr3/edm/EDMMENU.PROFILE
FILE_XFER anaftsrv : FTS==>> CONTROLS used during transfer.....
transfer options = efff5788 record format = 20 record length = 1
FILE_XFER anaftsrv : FTS==>> CONTROLS used during transfer.....
record option = 0 format option = 1 xlate option = 0
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK logical_read : Exiting with ret_code 0###
###FILE_XFER anaftsrv : Exiting with service_code 100###
```

```
SCHEDULER dispatch_work : Dispatcher processing complete for event
121 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
NETWORK Poll_Streams : table count: 1 poll count: 0
NETWORK Poll_Streams : table count: 1 poll count: 1
NETWORK naswait - poll : no event on any fd: error 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 114###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2381120###

SCHEDULER dispatch_work : Scheduler Dispatching Event 114 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : message request pdu
###MESSAGE anasdmsg : Entering with sq_ptr->SQ_conid 1###
###UTILITIES analzpdu : Entering with 0 0###
###UTILITIES analzpdu : Exiting with SUCCESSFUL_COMPLETION 0###
###MESSAGE anasdmsg : Exiting with PROCESS_COMPLETE 100###
SCHEDULER dispatch_work : Dispatcher processing complete for event
315 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
SCHEDULER scheduler : returning event 315
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 22 on conid 1
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2381120###

SCHEDULER dispatch_work : Scheduler Dispatching Event 22 for Conid
1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : consume event
###UTILITIES consume : Entering with sq_element->SQ_conid 1###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
```

```
###UTILITIES consume : Exiting with status 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
UTILITIES anabldpm : invalid CALLING_AE_TITLE for event 5
###UTILITIES anabldpm : Exiting with 25048 25048###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES anapmgmt : Entering with sq_ptr -> SQ_event_code 1###
!!!UTILITIES anapmgmt : Scheduler Queue Element
000000 00000000 00000000 00000000 00000000  *................*
000010 00000000 00000000 ef638ea8 efff88b4  *.........c......*
000020 00000001 00000000 00000000 00000000  *................*
000030 00000000 00000000 00000000 00000001  *................*
000040 00200000 efff8c70 efff8acc efff8d78  *. .....p.......x*
000050 00000000 efff8c70 00000000 00000000  *.......p........*
000060 00000000 00000000 00000000 00000000  *................*
###UTILITIES anapmgmt: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler: Scheduler entered with event 5 on conid
-283246592
###SCHEDULER gather_work: Entering with HOS 2347752###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work: Entering with SQHOS 2381120###

SCHEDULER dispatch_work: Scheduler Dispatching Event 5 for Conid 0

MACRO print_sq : user connection ID 0
MACRO print_sq : event
MACRO print_sq : initiate request
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 5###
###PROTOCOL setup_connect : Entering with con_p->ccb_id 2###
###UTILITIES bldpdu : Entering with sq_element->SQ_event_code 5###
###UTILITIES anaintpd : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu : Exiting with status 0###
###PROTOCOL verify_locals : Entering with 0 0###
###PROTOCOL verify_locals: Exiting with SUCCESSFUL_COMPLETION 0###
###NETWORK nsm_typename: Entering with net 7###
###NETWORK nsm_typename: Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###PROTOCOL parse_addr: Entering with netp->nas_type 7###
###PROTOCOL parse_addr: Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL do_connect: Exiting with ret_code 0###
###PROTOCOL setup_connect: Exiting with se_p->se_return 0###
###PROTOCOL protfsm: Exiting with ret_code 0###
SCHEDULER dispatch_work: Dispatcher processing complete for event
5 Conid 2
```

```
SCHEDULER dispatch_work: Scheduler Dispatching Event 122 for Conid
2

MACRO print_sq: user connection ID 2
MACRO print_sq: event
MACRO print_sq: positive event
###PROTOCOL protfsm: Entering with sq_p->SQ_event_code 122###
###PROTOCOL send_init_pdu: Entering with con_p->ccb_id 2###
###NETWORK logical_write: Entering with con_p->ccb_id 2###
###NETWORK net_write: Entering with con_p->ccb_id 2###
###NETWORK Check_PCI : Entering with dir 111###
###NETWORK Check_PCI : Exiting with pdu_type 101###
NETWORK Poll_Streams : table count: 2 poll count: 1
###NETWORK net_write : Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###PROTOCOL initiate_reads : Entering with con_p->ccb_id 2###
###NETWORK logical_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK logical_read : Exiting with ret_code 25056###
###PROTOCOL initiate_reads : Exiting with ret_code 0###
###PROTOCOL send_init_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work:Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 2347752###
SCHEDULER gather_work : CLEARING ECB 2 for CONID 2
SCHEDULER gather_work : Status Code 25056 Returned by SERVICE
SPECIFIC EXIT
###NETWORK net_write : Entering with con_p->ccb_id 2###
###NETWORK net_write : Exiting with ret_code 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTINE
###PROTOCOL write_handler : Entering with con_p->ccb_id 2###
###PROTOCOL write_handler : Exiting with SUCCESSFUL_COMPLETION
0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###NETWORK net_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 2###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 102###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2395864###
```

```
SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2

SCHEDULER dispatch_work : Scheduler Dispatching Event 102 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : initiate response pdu
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 102###
###PROTOCOL split_pdu : Entering with 0 0###
###UTILITIES analzpdu : Entering with 0 0###
###UTILITIES analzpdu : Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL split_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
301 Conid 2

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : even
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 100###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
SCHEDULER scheduler : returning 301
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 22 on conid 2
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2240152###

SCHEDULER dispatch_work:Scheduler Dispatching Event 22 for Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : consume event
###UTILITIES consume : Entering with sq_element->SQ_conid 2###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
```

```
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###NETWORK logical_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###UTILITIES consume : Exiting with status 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 2
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler: Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work: Entering with HOS 2347752###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 20 on conid 2
###SCHEDULER gather_work: Entering with HOS 2347752###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work: Entering with SQHOS 2240152###

SCHEDULER dispatch_work:Scheduler Dispatching Event 20 for Conid 2

MACRO print_sq: user connection ID 2
MACRO print_sq: event
MACRO print_sq: message request
###MESSAGE anasdmsg: Entering with sq_ptr->SQ_conid 2###
###UTILITIES bldpdu: Entering with sq_element->SQ_event_code 20###
###UTILITIES anaintp : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu: Exiting with status 0###
###NETWORK logical_write: Entering with con_p->ccb_id 2###
###NETWORK net_write: Entering with con_p->ccb_id 2###
###NETWORK Check_PCI: Entering with dir 111###
###NETWORK Check_PCI: Exiting with pdu_type 114###
###NETWORK net_write: Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###MESSAGE anasdmsg: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER dispatch_work: Dispatcher processing complete for event
20 Conid 2
###SCHEDULER dispatch_work:Exiting with current_request_status
0###
NETWORK Poll_Streams: table count: 2 poll count: 0
###SCHEDULER gather_work: Entering with HOS 2347752###
```

```
###NETWORK net_write: Entering with con_p->ccb_id 2###
###NETWORK net_write: Exiting with ret_code 0###
SCHEDULER gather_work: Status Code 0 Returned by NETWORK WRITE
ROUTINE
###MESSAGE message_write_handler: Entering with 0 0###
###MESSAGE message_write_handler : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work: Entering with SQHOS 2240152###

SCHEDULER dispatch_work:Scheduler Dispatching Event 122 for Conid
2

MACRO print_sq: user connection ID 2
MACRO print_sq: event
MACRO print_sq: positive event
###MESSAGE anasdmsg: Entering with sq_ptr->SQ_conid 2###
###MESSAGE anasdmsg: Exiting with PROCESS_COMPLETE 100###
SCHEDULER dispatch_work: Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 7 on conid 2
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2240152###

SCHEDULER dispatch_work: Scheduler Dispatching Event 7 for Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : terminate request
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 7###
###PROTOCOL make_pdu : Entering with sq_p->SQ_event_code 7###
###UTILITIES bldpdu : Entering with sq_element->SQ_event_code 7###
###UTILITIES anaintpd : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu : Exiting with status 0###
###NETWORK logical_write : Entering with con_p->ccb_id 2###
###NETWORK net_write : Entering with con_p->ccb_id 2###
###NETWORK Check_PCI : Entering with dir 111###
###NETWORK Check_PCI : Exiting with pdu_type 103###
###NETWORK net_write : Exiting with ret_code 0###
```

```
###NETWORK logical_write : Exiting with ret_code 25056###
###PROTOCOL make_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
7 Conid 2
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###NETWORK net_write : Entering with con_p->ccb_id 2###
###NETWORK net_write : Exiting with ret_code 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTINE
###PROTOCOL write_handler : Entering with con_p->ccb_id 2###
###PROTOCOL write_handler: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2240152###

SCHEDULER dispatch_work:Scheduler Dispatching Event 122 for Conid
2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
NETWORK Poll_Streams : table count: 2 poll count: 0
NETWORK Poll_Streams : table count: 2 poll count: 1
NETWORK naswait - poll : no event on any fd: error 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###NETWORK net_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 2###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 104###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2240152###
```

```
SCHEDULER dispatch_work : Scheduler Dispatching Event 104 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : terminate response pdu
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 104###
###PROTOCOL save_term_resp : Entering with con_p->ccb_id 2###
###PROTOCOL save_term_resp : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
104 Conid 2

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL net_close : Entering with con_p->ccb_id 2###
###PROTOCOL net_close : Exiting with ret_code 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
NETWORK Poll_Streams : table count: 2 poll count: 1
###SCHEDULER gather_work : Entering with HOS 2347752###
###PROTOCOL net_close_handler : Entering with con_p->ccb_id 2###
###PROTOCOL net_close_handler : Exiting with SUCCESSFUL_COMPLETION
0###
SCHEDULER gather_work : Status Code 0 Returned by SERVICE SPECIFIC
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2395864###

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL toss_connection : Entering with con_p->ccb_id 2###
###UTILITIES analzpdu : Entering with 0 0###
###UTILITIES analzpdu : Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL toss_connection : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
SCHEDULER scheduler : returning event 303
```

```
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 22 on conid 2
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2395864###

SCHEDULER dispatch_work : Scheduler Dispatching Event 22 for Conid
2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : consume event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 22###
###UTILITIES consume : Entering with sq_element->SQ_conid 2###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES consume : Exiting with status 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 4 on conid 0
###UTILITIES anapmgmt : Entering with sq_ptr -> SQ_event_code 4###
!!!UTILITIES anapmgmt : Scheduler Queue Element
000000 00000000 00000000 00000000 00000000  *................*
000010 00000000 00000000 ef638f88 efff8c5c  *.........c.....\*
000020 00000001 00000000 00000000 00000000  *................*
000030 00000000 00000000 00000000 00000004  *................*
000040 00000000 efff8f08 00000000 00000000  *................*
000050 00000000 00000000 00000000 00000000  *................*
000060 00000000 00000000 00000000 00000000  *................*
###UTILITIES anapmgmt : Exiting with utilstat 0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 7 on conid 1
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2417448###
```

```
SCHEDULER dispatch_work : Scheduler Dispatching Event 7 for Conid
1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : terminate request
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 7###
###PROTOCOL make_pdu : Entering with sq_p->SQ_event_code 7###
###UTILITIES bldpdu : Entering with sq_element->SQ_event_code 7###
###UTILITIES anaintpd : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu : Exiting with status 0###
###NETWORK logical_write : Entering with con_p->ccb_id 1###
###NETWORK net_write : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 111###
###NETWORK Check_PCI : Exiting with pdu_type 103###
###NETWORK net_write : Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###PROTOCOL make_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
7 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###NETWORK net_write : Entering with con_p->ccb_id 1###
###NETWORK net_write : Exiting with ret_code 0###

SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTINE
###PROTOCOL write_handler : Entering with con_p->ccb_id 1###
###PROTOCOL write_handler : Exiting with SUCCESSFUL_COMPLETION
0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2417448###

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
```

```
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
NETWORK Poll_Streams : table count: 1 poll count: 0
NETWORK Poll_Streams : table count: 1 poll count: 1
NETWORK naswait - poll : no event on any fd: error 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 104###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2417448###

SCHEDULER dispatch_work : Scheduler Dispatching Event 104 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : terminate response pdu
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 104###
###PROTOCOL save_term_resp : Entering with con_p->ccb_id 1###
###PROTOCOL save_term_resp : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
104 Conid 1

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL net_close : Entering with con_p->ccb_id 1###
###PROTOCOL net_close : Exiting with ret_code 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
NETWORK Poll_Streams : table count: 1 poll count: 0
NETWORK Poll_Streams : table count: 1 poll count: 1
NETWORK naswait - poll : no event on any fd: error 0
###SCHEDULER gather_work : Entering with HOS 2347752###
###PROTOCOL net_close_handler : Entering with con_p->ccb_id 1###
NETWORK gen_pollfd_table : Active_Desc_Cnt is zero
###PROTOCOL net_close_handler: Exiting with SUCCESSFUL_COMPLETION
0###
```

```
SCHEDULER gather_work: Status Code 0 Returned by SERVICE SPECIFIC
EXIT
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2240152###

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL toss_connection : Entering with con_p->ccb_id 1###
###UTILITIES analzpdu : Entering with 0 0###
###UTILITIES analzpdu : Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL toss_connection : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
SCHEDULER scheduler : returning event 303
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 22 on conid 1
###SCHEDULER gather_work : Entering with HOS 2347752###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 2240152###

SCHEDULER dispatch_work : Scheduler Dispatching Event 22 for Conid
1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : consume event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 22###
###UTILITIES consume : Entering with sq_element->SQ_conid 1###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES consume : Exiting with status 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 4 on conid 0
###UTILITIES anapmgmt : Entering with sq_ptr -> SQ_event_code 4###
```

```
!!!UTILITIES anapmgmt : Scheduler Queue Element
000000 00000000 00000000 00000000 00000000  *................*
000010 00000000 00000000 ef638f88 efff9d7c  *........c.....|*
000020 00000001 00000000 00000000 00000000  *................*
000030 00000000 00000000 00000000 00000004  *................*
000040 00000000 efffb11c 00000000 00000000  *................*
000050 00000000 00000000 00000000 00000000  *................*
000060 00000000 00000000 00000000 00000000  *................*
###NETWORK cleanup_drivers : Entering with 0 0###
###NETWORK nsm_typename : Entering with net 7###
###NETWORK nsm_typename : Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###NETWORK nsm_typename : Entering with net 1###
###NETWORK nsm_typename : Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###NETWORK cleanup_drivers : Exiting with 0 0###
###UTILITIES anapmgmt : Exiting with utilstat 0###

CDMSON016I Sign on to EDM server PAVAN completed successfully.  You
have 0 EDM message(s).
```

NT

```
CDMSON000I

###UTILITIES anapmgmt: Entering with sq_ptr -> SQ_event_code 1###
!!!UTILITIES anapmgmt : Scheduler Queue Element
000000 00000000 00000000 00000000 00000000  *................*
000010 00000000 00000000 70f16200 10a41200  *........pñb..¤..*
000020 01000000 00000000 00000000 00000000  *................*
000030 00000000 00000000 00000000 01000000  *................*
000040 00040000 6cbb1200 e0a41200 74bc1200  *....l»..à¤..t¿..*
000050 00000000 6cbb1200 00000000 00000000  *....l»..........*
000060 00000000 00000000 00000000 00000000  *................*
###NETWORK initialize_drivers : Entering with 0 0###
###NETWORK nsm_typename : Entering with net 7###
###NETWORK nsm_typename : Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###NETWORK nsm_typename : Entering with net 7###

###NETWORK nsm_typename : Exiting with
*NAS_Driver_Protocol_Names[0] 85###
NETWORK initialize_drivers: NSM does not know driver LOC net type
LOC
###NETWORK nsm_typename : Entering with net 1###
###NETWORK nsm_typename : Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###NETWORK initialize_drivers : Exiting with any_functional 1###
###UTILITIES anapmgmt : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 5 on conid 1
###SCHEDULER gather_work : Entering with HOS 0###
```

```
###SCHEDULER gather_work : Exiting with NO_EVENT_PENDING 25024###
###SCHEDULER dispatch_work : Entering with SQHOS 13059616###

SCHEDULER dispatch_work: Scheduler Dispatching Event 5 for Conid 0

MACRO print_sq : user connection ID 0
MACRO print_sq : event
MACRO print_sq : initiate request
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 5###
###PROTOCOL setup_connect : Entering with con_p->ccb_id 1###
###UTILITIES bldpdu : Entering with sq_element->SQ_event_code 5###
###UTILITIES anaintpd : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu : Exiting with status 0###
###PROTOCOL verify_locals : Entering with 0 0###
###PROTOCOL verify_locals : Exiting with SUCCESSFUL_COMPLETION
0###
###NETWORK nsm_typename : Entering with net 1###
###NETWORK nsm_typename : Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###PROTOCOL parse_addr : Entering with netp->nas_type 1###
###PROTOCOL parse_addr : Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL do_connect : Exiting with ret_code 0###
###PROTOCOL setup_connect : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
5 Conid 1
SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###

###PROTOCOL send_init_pdu : Entering with con_p->ccb_id 1###
###NETWORK logical_write : Entering with con_p->ccb_id 1###
###NETWORK net_write : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 111###
###NETWORK Check_PCI : Exiting with pdu_type 101###
###NETWORK net_write : Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###PROTOCOL initiate_reads : Entering with con_p->ccb_id 1###
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###PROTOCOL initiate_reads : Exiting with ret_code 0###
###PROTOCOL send_init_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
```

```
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_write : Entering with con_p->ccb_id 1###
###NETWORK net_write : Exiting with ret_code 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTINE
###PROTOCOL write_handler : Entering with con_p->ccb_id 1###
###PROTOCOL write_handler : Exiting with SUCCESSFUL_COMPLETION
0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069392###

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 102###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069392###
```

```
SCHEDULER dispatch_work : Scheduler Dispatching Event 102 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : initiate response pdu
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 102###
###PROTOCOL split_pdu : Entering with 0 0###
###UTILITIES analzpdu : Entering with 0 0###
###UTILITIES analzpdu : Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL split_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
301 Conid 1

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 100###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
SCHEDULER scheduler : returning event 301
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 22 on conid 1
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069504###

SCHEDULER dispatch_work : Scheduler Dispatching Event 22 for Conid
1

MACRO print_sq : user connection ID
MACRO print_sq : event
MACRO print_sq : consume event
###UTILITIES consume : Entering with sq_element->SQ_conid 1###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
```

```
###UTILITIES analzdcp : Exiting with return_status 0###
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###UTILITIES consume : Exiting with status 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 20 on conid 1
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069504###

SCHEDULER dispatch_work : Scheduler Dispatching Event 20 for Conid
1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : message request
###MESSAGE anasdmsg : Entering with sq_ptr->SQ_conid 1###
###UTILITIES bldpdu : Entering with sq_element->SQ_event_code
20###
###UTILITIES anaintpd: Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd: Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu: Exiting with status 0###
###NETWORK logical_write: Entering with con_p->ccb_id 1###
###NETWORK net_write: Entering with con_p->ccb_id 1###
###NETWORK Check_PCI: Entering with dir 111###
###NETWORK Check_PCI: Exiting with pdu_type 114###
###NETWORK net_write: Exiting with ret_code 0###
###NETWORK logical_write: Exiting with ret_code 25056###
###MESSAGE anasdmsg: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
20 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_write: Entering with con_p->ccb_id 1###
###NETWORK net_write: Exiting with ret_code 0###
SCHEDULER gather_work: Status Code 0 Returned by NETWORK WRITE
ROUTINE
###MESSAGE message_write_handler : Entering with 0 0###
###MESSAGE message_write_handler : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work:Status Code 0 Returned by NETWORK WRITE EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069504###
```

```
SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###MESSAGE anasdmsg : Entering with sq_ptr->SQ_conid 1###
###MESSAGE anasdmsg : Exiting with PROCESS_COMPLETE 100###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Exiting with pdu_type 114###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069504###

SCHEDULER dispatch_work : Scheduler Dispatching Event 114 for
Conid 1

MACRO print_sq : user connection ID
MACRO print_sq : even
MACRO print_sq : message request pdu
###MESSAGE anasdmsg : Entering with sq_ptr->SQ_conid 1###
###UTILITIES analzpdu : Entering with 0 0###
###UTILITIES analzpdu : Exiting with SUCCESSFUL_COMPLETION 0###
###MESSAGE anasdmsg : Exiting with PROCESS_COMPLETE 100###
SCHEDULER dispatch_work : Dispatcher processing complete for event
315 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
SCHEDULER scheduler : returning event 315
```

```
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 22 on conid 1
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069504###

SCHEDULER dispatch_work : Scheduler Dispatching Event 22 for Conid
1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : consume event
###UTILITIES consume : Entering with sq_element->SQ_conid 1###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###NETWORK logical_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###UTILITIES consume : Exiting with status 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES anabldpm : Exiting with 25048 25048###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES anapmgmt : Entering with sq_ptr -> SQ_event_code 1###
!!!UTILITIES anapmgmt : Scheduler Queue Element

000000 00000000 00000000 00000000 00000000  *................*
000010 00000000 00000000 70f16200 548f1200  *........pñb.T...*
000020 01000000 00000000 00000000 00000000  *................*
000030 00000000 00000000 00000000 01000000  *................*
000040 00040000 9c941200 24901200 a4951200  *......$...¤..*
000050 00000000 9c941200 00000000 00000000  *..............*
000060 00000000 00000000 00000000 00000000  *................*
###UTILITIES anapmgmt : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 5 on conid 1
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069504###

SCHEDULER dispatch_work: Scheduler Dispatching Event 5 for Conid 0

MACRO print_sq : user connection ID 0
MACRO print_sq : event
MACRO print_sq : initiate request
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 5###
###PROTOCOL setup_connect : Entering with con_p->ccb_id 2###
```

```
###UTILITIES bldpdu : Entering with sq_element->SQ_event_code 5###
###UTILITIES anaintpd : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu : Exiting with status 0###
###PROTOCOL verify_locals : Entering with 0 0###
###PROTOCOL verify_locals : Exiting with SUCCESSFUL_COMPLETION
0###
###NETWORK nsm_typename : Entering with net 1###
###NETWORK nsm_typename : Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###PROTOCOL parse_addr : Entering with netp->nas_type 1###
###PROTOCOL parse_addr : Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL do_connect : Exiting with ret_code 0###
###PROTOCOL setup_connect : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
5 Conid 2

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL send_init_pdu : Entering with con_p->ccb_id 2###
###NETWORK logical_write : Entering with con_p->ccb_id 2###
###NETWORK net_write : Entering with con_p->ccb_id 2###
###NETWORK Check_PCI : Entering with dir 111###
###NETWORK Check_PCI : Exiting with pdu_type 101###
###NETWORK net_write : Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###PROTOCOL initiate_reads : Entering with con_p->ccb_id 2###
###NETWORK logical_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###PROTOCOL initiate_reads : Exiting with ret_code 0###
###PROTOCOL send_init_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_write : Entering with con_p->ccb_id 2###
###NETWORK net_write : Exiting with ret_code 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTINE
###PROTOCOL write_handler : Entering with con_p->ccb_id 2###
```

```
###PROTOCOL write_handler : Exiting with SUCCESSFUL_COMPLETION
0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13059616###

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 2###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 102###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13059616###

SCHEDULER dispatch_work : Scheduler Dispatching Event 102 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : initiate response pdu
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 102###
###PROTOCOL split_pdu : Entering with 0 0###
###UTILITIES analzpdu : Entering with 0 0###
###UTILITIES analzpdu : Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL split_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
301 Conid 2
```

```
SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 100###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
SCHEDULER scheduler : returning event 301
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 22 on conid 2
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069392###

SCHEDULER dispatch_work:Scheduler Dispatching Event 22 for Conid 2

MACRO print_sq: user connection ID 2
MACRO print_sq : event
MACRO print_sq : consume event
###UTILITIES consume : Entering with sq_element->SQ_conid 2###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###NETWORK logical_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Exiting with ret_code 25056###
###NETWORK logical_read : Exiting with ret_code 25056###
###UTILITIES consume : Exiting with status 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 2
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler: Scheduler entered with event 20 on conid 2
###SCHEDULER gather_work: Entering with HOS 22526552###
###SCHEDULER gather_work: Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work: Entering with SQHOS 13069392###
```

```
SCHEDULER dispatch_work:Scheduler Dispatching Event 20 for Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : message request
###MESSAGE anasdmsg : Entering with sq_ptr->SQ_conid 2###
###UTILITIES bldpdu : Entering with sq_element->SQ_event_code
20###
###UTILITIES anaintpd : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu : Exiting with status 0###
###NETWORK logical_write : Entering with con_p->ccb_id 2###
###NETWORK net_write : Entering with con_p->ccb_id 2###
###NETWORK Check_PCI : Entering with dir 111###
###NETWORK Check_PCI : Exiting with pdu_type 114###
###NETWORK net_write : Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###MESSAGE anasdmsg : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
20 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_write : Entering with con_p->ccb_id 2###
###NETWORK net_write : Exiting with ret_code 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTINE
###MESSAGE message_write_handler : Entering with 0 0###
###MESSAGE message_write_handler : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work:Status Code 0 Returned by NETWORK WRITE EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069392###

SCHEDULER dispatch_work:Scheduler Dispatching Event 122 for Conid
2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###MESSAGE anasdmsg : Entering with sq_ptr->SQ_conid 2###
###MESSAGE anasdmsg : Exiting with PROCESS_COMPLETE 100###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 7 on conid 2
```

```
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069392###

SCHEDULER dispatch_work : Scheduler Dispatching Event 7 for Conid
2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : terminate request
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 7###
###PROTOCOL make_pdu : Entering with sq_p->SQ_event_code 7###
###UTILITIES bldpdu : Entering with sq_element->SQ_event_code 7###
###UTILITIES anaintpd : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu : Exiting with status 0###
###NETWORK logical_write : Entering with con_p->ccb_id 2###
###NETWORK net_write : Entering with con_p->ccb_id 2###
###NETWORK Check_PCI : Entering with dir 111###
###NETWORK Check_PCI : Exiting with pdu_type 103###
###NETWORK net_write : Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###PROTOCOL make_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
7 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_write : Entering with con_p->ccb_id 2###
###NETWORK net_write : Exiting with ret_code 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTINE
###PROTOCOL write_handler : Entering with con_p->ccb_id 2###
###PROTOCOL write_handler : Exiting with SUCCESSFUL_COMPLETION
0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069392###

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
```

```
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_read : Entering with con_p->ccb_id 2###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 2###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 104###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069392###

SCHEDULER dispatch_work : Scheduler Dispatching Event 104 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : terminate response pdu
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 104###
###PROTOCOL save_term_resp : Entering with con_p->ccb_id 2###
###PROTOCOL save_term_resp : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
104 Conid 2

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL net_close : Entering with con_p->ccb_id 2###
###PROTOCOL net_close : Exiting with ret_code 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###SCHEDULER gather_work : Entering with HOS 22526552###
###PROTOCOL net_close_handler : Entering with con_p->ccb_id 2###
###PROTOCOL net_close_handler : Exiting with SUCCESSFUL_COMPLETION
0###
SCHEDULER gather_work : Status Code 0 Returned by SERVICE SPECIFIC
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069504###
```

```
SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL toss_connection : Entering with con_p->ccb_id 2###
###UTILITIES analzpdu : Entering with 0 0###
###UTILITIES analzpdu : Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL toss_connection : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
SCHEDULER scheduler : returning event 303
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 22 on conid 2
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13069504###

SCHEDULER dispatch_work : Scheduler Dispatching Event 22 for Conid
2

MACRO print_sq : user connection ID 2
MACRO print_sq : event
MACRO print_sq : consume event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 22###
###UTILITIES consume : Entering with sq_element->SQ_conid 2###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES consume : Exiting with status 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 2
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 4 on conid 0
###UTILITIES anapmgmt : Entering with sq_ptr -> SQ_event_code 4###
!!!UTILITIES anapmgmt : Scheduler Queue Element
000000 00000000 00000000 00000000 00000000  *................*
000010 00000000 00000000 50f26200 08931200  *........Pòb....*
000020 01000000 00000000 00000000 00000000  *................*
000030 00000000 00000000 00000000 04000000  *................*
```

```
000040 00000000 10951200 00000000 00000000  *................*
000050 00000000 00000000 00000000 00000000  *................*
000060 00000000 00000000 00000000 00000000  *................*
###UTILITIES anapmgmt : Exiting with utilstat 0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 7 on conid 1
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13072128###

SCHEDULER dispatch_work : Scheduler Dispatching Event 7 for Conid
1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : terminate request
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 7###
###PROTOCOL make_pdu : Entering with sq_p->SQ_event_code 7###
###UTILITIES bldpdu : Entering with sq_element->SQ_event_code 7###
###UTILITIES anaintpd : Entering with SUCCESSFUL_COMPLETION 0###
###UTILITIES anaintpd : Exiting with SUCCESSFUL_COMPLETION 0###
###UTILITIES bldpdu : Exiting with status 0###
###NETWORK logical_write : Entering with con_p->ccb_id 1###
###NETWORK net_write : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 111###
###NETWORK Check_PCI : Exiting with pdu_type 103###
###NETWORK net_write : Exiting with ret_code 0###
###NETWORK logical_write : Exiting with ret_code 25056###
###PROTOCOL make_pdu : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
7 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_write : Entering with con_p->ccb_id 1###
###NETWORK net_write : Exiting with ret_code 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
ROUTINE
###PROTOCOL write_handler : Entering with con_p->ccb_id 1###
###PROTOCOL write_handler : Exiting with SUCCESSFUL_COMPLETION
0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK WRITE
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13072128###
```

```
SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL protfsm : Exiting with se_p->se_return 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 21 on conid 0
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER gather_work : Entering with HOS 22526552###
###NETWORK net_read : Entering with con_p->ccb_id 1###
###NETWORK net_read : Exiting with ret_code 0###
###NETWORK convert_pdu_to_event : Entering with con_p->ccb_id 1###
###NETWORK Check_PCI : Entering with dir 105###
###NETWORK Check_PCI : Exiting with pdu_type 104###
###NETWORK convert_pdu_to_event : Exiting with
SUCCESSFUL_COMPLETION 0###
SCHEDULER gather_work : Status Code 0 Returned by NETWORK READ
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13072128###

SCHEDULER dispatch_work : Scheduler Dispatching Event 104 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : terminate response pdu
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 104###
###PROTOCOL save_term_resp : Entering with con_p->ccb_id 1###
###PROTOCOL save_term_resp : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
104 Conid 1

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL net_close : Entering with con_p->ccb_id 1###
###PROTOCOL net_close : Exiting with ret_code 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
```

```
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work : Exiting with current_request_status
0###
###SCHEDULER gather_work : Entering with HOS 22526552###
###PROTOCOL net_close_handler : Entering with con_p->ccb_id 1###
###PROTOCOL net_close_handler : Exiting with SUCCESSFUL_COMPLETION
0###
SCHEDULER gather_work : Status Code 0 Returned by SERVICE SPECIFIC
EXIT
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work : Entering with SQHOS 13072272###

SCHEDULER dispatch_work : Scheduler Dispatching Event 122 for
Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : positive event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 122###
###PROTOCOL toss_connection : Entering with con_p->ccb_id 1###
###UTILITIES analzpdu : Entering with 0 0###
###UTILITIES analzpdu : Exiting with SUCCESSFUL_COMPLETION 0###
###PROTOCOL toss_connection : Exiting with se_p->se_return 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
122 Conid 1
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
SCHEDULER scheduler : returning event 303
###UTILITIES bldschdprm : Entering with 0 0###
###UTILITIES bldschdprm : Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler : Scheduler entered with event 22 on conid 1
###SCHEDULER gather_work : Entering with HOS 22526552###
###SCHEDULER gather_work : Exiting with SUCCESSFUL_COMPLETION 0###
###SCHEDULER dispatch_work: Entering with SQHOS 13072272###

SCHEDULER dispatch_work:Scheduler Dispatching Event 22 for Conid 1

MACRO print_sq : user connection ID 1
MACRO print_sq : event
MACRO print_sq : consume event
###PROTOCOL protfsm : Entering with sq_p->SQ_event_code 22###
###UTILITIES consume : Entering with sq_element->SQ_conid 1###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Entering with return_status 0###
###UTILITIES analzdcp : Exiting with return_status 0###
###UTILITIES consume : Exiting with status 0###
###PROTOCOL protfsm : Exiting with ret_code 0###
SCHEDULER dispatch_work : Dispatcher processing complete for event
22 Conid 1
```

```
###SCHEDULER dispatch_work: Exiting with current_request_status
0###
###UTILITIES bldschdprm: Entering with 0 0###
###UTILITIES bldschdprm: Exiting with SUCCESSFUL_COMPLETION 0###
SCHEDULER scheduler: Scheduler entered with event 4 on conid 0
###UTILITIES anapmgmt: Entering with sq_ptr -> SQ_event_code 4###
!!!UTILITIES anapmgmt: Scheduler Queue Element
000000 00000000 00000000 00000000 00000000  *................*
000010 00000000 00000000 50f26200 14a41200  *........Pòb..¤..*
000020 01000000 00000000 00000000 00000000  *................*
000030 00000000 00000000 00000000 04000000  *................*
000040 00000000 1ca61200 00000000 00000000  *.....|.........*
000050 00000000 00000000 00000000 00000000  *................*
000060 00000000 00000000 00000000 00000000  *................*
###NETWORK cleanup_drivers : Entering with 0 0###
###NETWORK nsm_typename : Entering with net 7###
###NETWORK nsm_typename : Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###NETWORK nsm_typename : Entering with net 1###
###NETWORK nsm_typename : Exiting with
*NAS_Driver_Protocol_Names[0] 85###
###NETWORK cleanup_drivers : Exiting with 0 0###
###UTILITIES anapmgmt : Exiting with utilstat 0###

CDMSON016I Sign on to EDM server DHUMKETU completed successfully.
You have 0 EDM message(s).
```

Tracing Vault Problems with CVTRACE

If you are experiencing a Vault problem and you cannot resolve the problem, run a trace on Vault.

To run a trace on Vault, create a file called `CVTRACE.PARAMS` (for VM, the file is called `CVTRACE PARAMS`).

You can run `ANSTLEVL` with `CVTRACE`. To do this, set the `ANSTLEVL` environment variable before you begin the `CVTRACE` utility. The output from `ANSTLEVL` can be captured in the same file as the output from `CVTRACE`.

Please note: If you are having a problem with a particular Vault command and the problem does not seem to involve the full-screen interface, set `CVTRACE` and then run the Vault command using the command line format. This can generate less output than running the command using the full-screen format.

CVTRACE.PARAMS File Keywords

To capture all tracing from Vault execution, enter only the word ON in the CVTRACE.PARAMS file. In general, this is how you use CVTRACE.

The Customer Service Center can ask you to limit the trace to certain Vault subroutines. The keywords and the values you can be asked to create in the CVTRACE.PARAMS file are listed here for your information.

CODES

Specify a codes list to determine which trace commands are to be executed. The codes list is used for the functions specified with the INCLUDE keyword. If a codes list is not specified, all trace commands are printed. Possible values for the CODES keyword are:

- E (trace_entry)
- C (trace_com)
- D (trace_number, trace_string)
- L (trace_lower)
- S (trace_sql)
- X (trace_exit)
- O (trace_start)
- H (trace_stop)

INCLUDE

This keyword indicates the start of an INCLUDE list. Following the keyword are the function names to be included in the trace. Function names can be supplied by the Customer Service Center and trace commands for the functions in the INCLUDE list can be printed out, based on the codes list specified with the CODES keyword. Function names are restricted to 32 characters.

ENDINCLUDE

This keyword indicates the end of the INCLUDE list. If more than one INCLUDE list is given in the CVTRACE.PARAMS file, the last one is used.

EXCLUDE

This keyword indicates the start of an EXCLUDE list. Following the keyword are the function names to be excluded from the trace and function names can be supplied by the Customer Service Center.

Trace commands for the functions in the EXCLUDE list can not be printed out. The EXCLUDE list has precedence over the INCLUDE list.

ENDEXCLUDE

This keyword indicates the end of the EXCLUDE list.

BUFFSIZE

Set this keyword if the output is to be buffered. The maximum value is 32756 and the default value is 1, which indicates that messages are to be printed out as they are created.

The following is an example of a CVTRACE.PARAMS file that the Customer Service Center might ask you to create. This file is not case-sensitive.

```
ON
CODES=(e,s,x)
INCLUDE
dm_reqrvw
submit_file
start_review
process_file
ENDINCLUDE
EXCLUDE
asmgr
ENDEXCLUDE
BUFFSIZE=1
```

Placing the CVTRACE.PARAMS File

You can place the CVTRACE.PARAMS file under the edm account (Solaris or SunOS), the NSM or PDM account (VMS), or the EDMMAINT account (VM). This way, the trace file can contain all Vault activity. You can also place the CVTRACE.PARAMS file under an individual user account on a Vault Client node.

If you have diagnosed a problem to a particular network process (DM, DN, DD, or LOG), you can run CVTRACE on that process. You can also run CVTRACE on any combination of the DM, DN, DD, or LOG. (Because no Vault code is involved in the Process Manager or the PCA, do not run CVTRACE on PMGR or PCA. Use ANSTLEVL and/or NASCNFIG to diagnose problems with the PMGR or PCA.)

Sample Output from the CVTRACE Utility

The following are two samples (UNIX, followed by NT) of the output you receive from the CVTRACE utility. If you run CVTRACE with ANSTLEVL and/or NASCNFIG and send the output to the same file, you can see output from each trace interspersed in the file.

UNIX

```
10:44:18.884 RECEIVE_MESSAGE : Return from edm_getmessages: with
rc of 0 and ec of 0
10:44:18.884 GET_NORM_MESSAGE_TYPE : Entered routine
GET_NORM_MESSAGE_TYPE
10:44:18.884 GET_NORM_MESSAGE_TYPE : Leaving routine
GET_NORM_MESSAGE_TYPE
10:44:18.884 RECEIVE_MESSAGE : received_message_type = 2
10:44:18.896 RECEIVE_MESSAGE : Leaving routine RECEIVE_MESSAGE
10:44:18.896 PROCESS_MESSAGES : Return from RECEIVE_MESSAGE: with
rc of 0 and ec of 0
10:44:18.896 PROCESS_MESSAGES :  Entering INVOKE_DM_FUNCTION
10:44:18.896 INVOKE_DM_FUNCTION : Entered routine
INVOKE_DM_FUNCTION
10:44:18.896 INVOKE_DM_FUNCTION : About to enter:  = COPY
10:44:18.896 DM_COPY : Entered routine dm_copy
10:44:18.897 DM_COPY :  Entering COPY_FILE
10:44:18.897 COPY_FILE : Entered routine COPY_FILE
10:44:18.897 MAX_REVS_FROM_TABLES : Entered routine
max_revs_from_tables
10:44:18.898 MAX_REVS_FROM_TABLES :  About to get max rev from
DM_FILE_DIRECTORY
10:44:18.902 MAX_REVS_FROM_TABLES : Return from SQL: Table
DM_FILE_DIRECTORY; Command SELECT
10:44:18.902  SQLCODE = 0; #Rows: 1 Comment: Max file rev
10:44:18.902 MAX_REVS_FROM_TABLES:About to get max rev from
DM_ARCHIVE
10:44:18.905 MAX_REVS_FROM_TABLES : Return from SQL: Table
DM_FILE_DIRECTORY; Command SELECT
10:44:18.905  SQLCODE = 100; #Rows: 0 Comment: Max file rev
10:44:18.905 MAX_REVS_FROM_TABLES:Leaving routine
max_revs_from_tables
10:44:18.905 COPY_FILE :  select a  row
10:44:18.909 COPY_FILE : Return from SQL: Table Select; Command
DM_FILE_DIRECTORY
10:44:18.909  SQLCODE = 0; #Rows: 1 Comment: dm_file_name
10:44:18.909 COPY_FILE :  Entering PROCESS_FILE
10:44:18.909 PROCESS_FILE : Entered routine PROCESS_FILE
10:44:18.909 PROCESS_FILE :  entering  CHECK_AUTHORITY
10:44:18.909 CHECK_AUTHORITY : Entered routine CHECK_AUTHORITY
10:44:18.910 CHECK_AUTHORITY :  entering  ASMGR
10:44:18.910 ASMGR : Entered routine asmgr
10:44:18.910 ASMGR :  About to select from AS_COMMAND_TEST
```

```
10:44:18.913 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:18.913  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:18.913 GET_SQL_TEXT : Entered routine get_sql_text
10:44:18.913 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:18.913 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:18.913  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:18.914 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:18.915 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:18.915  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:18.916 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:18.917 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.917  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:18.917 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
10:44:18.918 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.918  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:18.918 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:18.918 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:18.918  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.918 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:18.918 ASMGR :  Entering sql_placeholder
10:44:18.918 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:18.919 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_COMMAND_LIST,DM_USER_AUTH WHERE DM_AUTH_USER_ID = 'EDMADMIN'
AND DM_AUTH_PROJ_ID = ' ' AND DM_AUTH_CMD_LIST = DM_COMMAND_LIST
AND DM_COMMAND_NAME IN ('COPY','ALL')
10:44:18.919 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:18.919 ASMGR :  Out of sql_placeholder
10:44:18.919 ASMGR :  About to prepare stmt ROW_CNT
10:44:18.921 ASMGR : Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:18.921  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:18.921 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:18.921 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:18.921  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:18.921 ASMGR :  About to open AS_COUNT_CURSOR
10:44:18.922 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:18.922  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:18.922 ASMGR :  About to do fetch AS_COUNT_CURSOR into
rowcnt for test
10:44:18.924 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
```

```
10:44:18.924  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:18.924 ASMGR : Rowcount  = 1
10:44:18.924 ASMGR :  About to close AS_COUNT_CURSOR
10:44:18.924 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:18.924  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.924 ASMGR :  About to select from AS_COMMAND_TEST
10:44:18.926 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:18.926  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:18.926 GET_SQL_TEXT : Entered routine get_sql_text
10:44:18.926 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:18.926 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:18.926  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:18.926 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:18.927 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:18.927  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:18.927 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:18.928 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.928  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:18.928 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
10:44:18.929 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.929  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:18.929 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:18.929 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:18.929  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.929 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:18.929 ASMGR :  Entering sql_placeholder
10:44:18.929 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:18.930 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_AUTH_RD_AUTHG = ' '
10:44:18.930 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:18.930 ASMGR :  Out of sql_placeholder
10:44:18.930 ASMGR :  About to prepare stmt ROW_CNT
10:44:18.931 ASMGR : Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:18.931  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:18.931 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:18.931 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:18.931  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:18.932 ASMGR :  About to open AS_COUNT_CURSOR
```

```
10:44:18.932 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:18.932  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:18.932 ASMGR :  About to do fetch AS_COUNT_CURSOR into
rowcnt for test
10:44:18.933 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
10:44:18.933  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:18.934 ASMGR : Rowcount  = 1
10:44:18.934 ASMGR :  About to close AS_COUNT_CURSOR
10:44:18.934 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:18.934  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.934 ASMGR :  About to select from AS_COMMAND_TEST
10:44:18.935 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:18.935  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:18.935 GET_SQL_TEXT : Entered routine get_sql_text
10:44:18.935 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:18.935 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:18.935  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:18.936 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:18.936 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:18.936  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:18.937 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:18.937 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.937  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:18.938 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
10:44:18.939 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.939  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:18.939 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:18.939 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:18.939  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.940 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:18.940 ASMGR :  Entering sql_placeholder
10:44:18.940 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:18.940 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH,DM_STATUS_CODE WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_STAT_PROJ_ID = ' ' AND
DM_STAT_STATUS_CD = 'IW' AND DM_AUTH_RD_AUTH >= DM_STAT_AUTH
10:44:18.940 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:18.940 ASMGR :  Out of sql_placeholder
10:44:18.940 ASMGR :  About to prepare stmt ROW_CNT
10:44:18.942 ASMGR:Return from SQL: Table ROW_CNT; Command PREPARE
10:44:18.942  SQLCODE = 0; #Rows: 1 Comment: Preparing test
```

```
10:44:18.942 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:18.942 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:18.942  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:18.942 ASMGR :  About to open AS_COUNT_CURSOR
10:44:18.943 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:18.943  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:18.943 ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt
for test
10:44:18.945 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
10:44:18.945  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:18.945 ASMGR : Rowcount  = 1
10:44:18.945 ASMGR :  About to close AS_COUNT_CURSOR
10:44:18.945 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:18.945  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.945 ASMGR :  About to select from AS_COMMAND_TEST
10:44:18.946 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:18.946  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:18.946 GET_SQL_TEXT : Entered routine get_sql_text
10:44:18.946 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:18.947 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:18.947  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:18.947 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:18.948 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:18.948  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:18.948 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:18.949 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.949  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:18.949 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
10:44:18.950 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.950  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:18.950 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:18.950 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:18.950  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.950 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:18.950 ASMGR :  Entering sql_placeholder
10:44:18.950 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:18.950 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_AUTH_WR_AUTHG = ' '
```

```
10:44:18.950 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:18.951 ASMGR :  Out of sql_placeholder
10:44:18.951 ASMGR :  About to prepare stmt ROW_CNT
10:44:18.952 ASMGR : Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:18.952  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:18.952 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:18.952 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:18.952  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:18.952 ASMGR :  About to open AS_COUNT_CURSOR
10:44:18.953 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:18.953  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:18.953 ASMGR :  About to do fetch AS_COUNT_CURSOR into
rowcnt for test
10:44:18.954 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
10:44:18.954  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:18.954 ASMGR : Rowcount  = 1
10:44:18.954 ASMGR :  About to close AS_COUNT_CURSOR
10:44:18.955 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:18.955  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.955 ASMGR :  About to select from AS_COMMAND_TEST
10:44:18.956 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:18.956  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:18.956 GET_SQL_TEXT : Entered routine get_sql_text
10:44:18.956 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:18.956 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:18.956  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:18.956 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:18.957 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:18.957  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:18.957 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:18.958 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.958  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:18.958 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
10:44:18.959 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.959  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:18.959 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:18.959 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:18.959  SQLCODE = 0; #Rows: 1 Comment: Close cursor
```

```
10:44:18.960 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:18.960 ASMGR :  Entering sql_placeholder
10:44:18.960 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:18.960 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH,DM_STATUS_CODE WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_STAT_PROJ_ID = ' ' AND
DM_STAT_STATUS_CD = 'IW' AND DM_AUTH_WR_AUTH >= DM_STAT_AUTH
10:44:18.960 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:18.960 ASMGR :  Out of sql_placeholder
10:44:18.961 ASMGR :  About to prepare stmt ROW_CNT
10:44:18.962 ASMGR : Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:18.962  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:18.962 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:18.962 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:18.962  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:18.962 ASMGR :  About to open AS_COUNT_CURSOR
10:44:18.963 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:18.963  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:18.963 ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt
for test
10:44:18.964 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
10:44:18.964  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:18.964 ASMGR : Rowcount  = 1
10:44:18.965 ASMGR :  About to close AS_COUNT_CURSOR
10:44:18.965 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:18.965  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.965 ASMGR :  About to select from AS_COMMAND_TEST
10:44:18.966 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:18.966  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:18.966 GET_SQL_TEXT : Entered routine get_sql_text
10:44:18.966 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:18.966 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:18.966  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:18.966 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:18.967 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:18.967  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:18.968 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:18.969 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.969  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:18.969 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
```

```
10:44:18.970 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:18.970  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:18.970 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:18.970 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:18.970  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.970 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:18.970 ASMGR :  Entering sql_placeholder
10:44:18.970 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:18.970 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_AUTH_ADMIN_FLAG = 'Y'
10:44:18.971 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:18.971 ASMGR :  Out of sql_placeholder
10:44:18.971 ASMGR :  About to prepare stmt ROW_CNT
10:44:18.972 ASMGR : Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:18.972  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:18.972 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:18.972 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:18.972  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:18.972 ASMGR :  About to open AS_COUNT_CURSOR
10:44:18.973 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:18.973  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:18.973 ASMGR :  About to do fetch AS_COUNT_CURSOR into
rowcnt for test
10:44:18.974 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
10:44:18.974  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:18.975 ASMGR : Rowcount  = 1
10:44:18.975 ASMGR :  About to close AS_COUNT_CURSOR
10:44:18.975 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:18.975  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:18.975 ASMGR : Leaving routine asmgr
10:44:18.975 CHECK_AUTHORITY : Return from ASMGR: with rc of -1
and ec of 0
10:44:18.975 CHECK_AUTHORITY : Leaving routine CHECK_AUTHORITY
10:44:18.975 PROCESS_FILE : Return from CHECK_AUTHORITY: with rc
of 0 and ec of 0
10:44:18.975 PROCESS_FILE :  entering  CHECK_AVAILABILITY
10:44:18.975 CHECK_AVAILABILITY : Entered routine
CHECK_AVAILABILITY
10:44:18.976 CHECK_AVAILABILITY : Leaving routine
CHECK_AVAILABILITY
10:44:18.976 PROCESS_FILE : Return from CHECK_AVAILABILITY: with
rc of 0 and ec of 0
10:44:18.976 PROCESS_FILE :  entering  MAKE_NEW_NAME
10:44:18.976 MAKE_NEW_NAME : Entered routine MAKE_NEW_NAME
```

```
10:44:18.976 MAKE_NEW_NAME : Leaving routine MAKE_NEW_NAME
10:44:18.976 PROCESS_FILE : Return from MAKE_NEW_NAME: with rc of
0 and ec of 0
10:44:18.976 PROCESS_FILE : entering  GET_OLD_POOLINFO
10:44:18.976 GET_OLD_POOL_INFO : Entered routine GET_OLD_POOL_INFO
10:44:18.976 GET_OLD_POOL_INFO :  select for DM_POOL_INFO
10:44:18.978 GET_OLD_POOL_INFO: Return from SQL: Table SELECT;
Command DM_POOL_INFO
10:44:18.978  SQLCODE = 0; #Rows: 1 Comment: select text
10:44:18.979 GET_OLD_POOL_INFO : Leaving routine GET_OLD_POOL_INFO
10:44:18.979 PROCESS_FILE : Return from GET_OLD_POOLINFO: with rc
of 0 and ec of 0
10:44:18.979 PROCESS_FILE :  entering  SET_ATTRIBS
10:44:18.979 SET_ATTRIBS : Entered routine SET_ATTRIBS
10:44:18.979 SET_ATTRIBS : entering DM_STORE
10:44:18.979 DM_STORE : Entered routine dm_store
10:44:18.980 DM_STORE: select count of -1 versions
DM_FILE_DIRECTORY
10:44:18.982 DM_STORE : Return from SQL: Table SELECT; Command
DM_FILE_DIRECTORY
10:44:18.982  SQLCODE = 0; #Rows: 1 Comment: select count in
DM_FILE_DIRECTORY
10:44:18.982 DM_STORE :  Entering internal select_directory
10:44:18.982 SELECT_DIRECTORY : Entered routine select_directory
10:44:18.982 SELECT_DIRECTORY: About to select from
DM_FILE_DIRECTORY
10:44:18.987 SELECT_DIRECTORY : Return from SQL: Table
DM_FILE_DIRECTORY; Command SELECT
10:44:18.987  SQLCODE = 100; #Rows: 0 Comment: select row
10:44:18.987 SELECT_DIRECTORY : Leaving routine select_directory
10:44:18.987 DM_STORE : Return from select_directory: with rc of
97 and ec of 0
10:44:18.987 DM_STORE :  Entering GET_NUM_REV
10:44:18.987 GET_NUM_REV : Entered routine get_num_rev
10:44:18.988 GET_NUM_REV :  About to read project table
10:44:18.990 GET_NUM_REV : Return from SQL: Table DM_PROJECT;
Command SELECT
10:44:18.990  SQLCODE = 0; #Rows: 1 Comment: revision sequence name
10:44:18.990 GET_NUM_REV :  About to read revision code table
10:44:18.992 GET_NUM_REV : Return from SQL: Table
DM_REVISION_CODE; Command SELECT
10:44:18.992  SQLCODE = 0; #Rows: 1 Comment: num for given char
10:44:18.993 DM_STORE: Return from get_num_rev: with rc of 0 and ec
of 0
10:44:18.993 DM_STORE :  select count in DM_FILE_DIRECTORY
10:44:18.995 DM_STORE: Return from SQL: Table SELECT; Command
DM_FILE_DIRECTORY
10:44:18.995  SQLCODE = 0; #Rows: 1 Comment: select count in
DM_FILE_DIRECTORY
10:44:18.995 DM_STORE:  select count in DM_ARCHIVE
10:44:18.997 DM_STORE: Return from SQL: Table SELECT; Command
DM_ARCHIVE
10:44:18.997 SQLCODE = 0; #Rows:1 Comment:select count in
DM_ARCHIVE
```

```
10:44:18.997 SELECT_PREVIOUS_INFO : Entered routine
select_previous_info
10:44:18.997 SELECT_PREVIOUS_INFO :  Entering max_revs_from_tables
10:44:18.998 MAX_REVS_FROM_TABLES : Entered routine
max_revs_from_tables
10:44:18.998 MAX_REVS_FROM_TABLES :  About to get max rev from
DM_FILE_DIRECTORY
10:44:18.999 MAX_REVS_FROM_TABLES : Return from SQL: Table
DM_FILE_DIRECTORY; Command SELECT
10:44:18.999  SQLCODE = 100; #Rows: 0 Comment: Max file rev
10:44:18.999 MAX_REVS_FROM_TABLES :  About to get max rev from
DM_ARCHIVE
10:44:19.000 MAX_REVS_FROM_TABLES : Return from SQL: Table
DM_FILE_DIRECTORY; Command SELECT
10:44:19.000  SQLCODE = 100; #Rows: 0 Comment: Max file rev
10:44:19.000 MAX_REVS_FROM_TABLES : Leaving routine
max_revs_from_tables
10:44:19.000 SELECT_PREVIOUS_INFO : Return from
max_revs_from_tables: with rc of 97 and ec of 100
10:44:19.000 SELECT_PREVIOUS_INFO : Leaving routine
select_previous_info
10:44:19.001 DM_STORE :  Entering internal chk_store_file_auth
10:44:19.001 CHK_STORE_FILE_AUTH : Entered routine
chk_store_file_auth
10:44:19.001 CHK_STORE_FILE_AUTH :  Entering asmgr
10:44:19.001 ASMGR : Entered routine asmgr
10:44:19.001 ASMGR :  About to select from AS_COMMAND_TEST
10:44:19.002 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:19.002  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:19.002 GET_SQL_TEXT : Entered routine get_sql_text
10:44:19.002 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:19.002 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:19.002  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:19.003 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:19.003 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:19.003  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:19.004 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:19.004 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.004  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:19.005 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
10:44:19.005 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.005  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:19.006 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:19.006 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:19.006  SQLCODE = 0; #Rows: 1 Comment: Close cursor
```

```
10:44:19.006 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:19.006 ASMGR :  Entering sql_placeholder
10:44:19.006 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:19.007 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_COMMAND_LIST,DM_USER_AUTH WHERE DM_AUTH_USER_ID = 'EDMADMIN'
AND DM_AUTH_PROJ_ID = ' ' AND DM_AUTH_CMD_LIST = DM_COMMAND_LIST
AND DM_COMMAND_NAME IN ('STORE','ALL')
10:44:19.007 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:19.007 ASMGR :  Out of sql_placeholder
10:44:19.007 ASMGR :  About to prepare stmt ROW_CNT
10:44:19.009 ASMGR: Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:19.009  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:19.009 ASMGR:  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:19.009 ASMGR: Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:19.009  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:19.009 ASMG :  About to open AS_COUNT_CURSOR
10:44:19.010 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:19.010  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:19.010 ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt
for test
10:44:19.011 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
10:44:19.011  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:19.011 ASMGR : Rowcount  = 1
10:44:19.011 ASMGR :  About to close AS_COUNT_CURSOR
10:44:19.012 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:19.012  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:19.012 ASMGR :  About to select from AS_COMMAND_TEST
10:44:19.013 ASMGR: Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:19.013  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:19.013 GET_SQL_TEXT : Entered routine get_sql_text
10:44:19.013 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:19.013 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:19.013  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:19.013 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:19.014 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:19.014  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:19.014 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:19.015 GET_SQL_TEXT: Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.015  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:19.015 GET_SQL_TEXT:  About to do next fetch from
AS_TEST_CURSOR
```

```
10:44:19.016 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.016  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:19.016 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:19.016 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:19.016  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:19.017 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:19.017 ASMGR :  Entering sql_placeholder
10:44:19.017 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:19.017 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_STATUS_CODE WHERE DM_STAT_PROJ_ID = ' ' AND DM_STAT_STATUS_CD =
'IW'
10:44:19.017 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:19.017 ASMGR :  Out of sql_placeholder
10:44:19.017 ASMGR :  About to prepare stmt ROW_CNT
10:44:19.018 ASMGR : Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:19.018  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:19.018 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:19.019 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:19.019  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:19.019 ASMGR :  About to open AS_COUNT_CURSOR
10:44:19.019 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:19.019  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:19.020 ASMGR :  About to do fetch AS_COUNT_CURSOR into
rowcnt for test
10:44:19.020 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
10:44:19.020  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:19.021 ASMGR : Rowcount  = 1
10:44:19.021 ASMGR :  About to close AS_COUNT_CURSOR
10:44:19.021 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:19.021  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:19.021 ASMGR :  About to select from AS_COMMAND_TEST
10:44:19.022 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:19.022  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:19.022 GET_SQL_TEXT : Entered routine get_sql_text
10:44:19.022 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:19.022 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:19.022  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:19.023 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:19.023 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:19.023  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
```

```
10:44:19.024 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:19.024 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.024  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:19.025 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
10:44:19.025 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.025  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:19.026 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:19.026 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:19.026  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:19.026 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:19.026 ASMGR :  Entering sql_placeholder
10:44:19.026 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:19.026 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_AUTH_WR_AUTHG = ' '
10:44:19.027 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:19.027 ASMGR :  Out of sql_placeholder
10:44:19.027 ASMGR :  About to prepare stmt ROW_CNT
10:44:19.028 ASMGR : Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:19.028  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:19.028 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:19.028 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:19.028  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:19.028 ASMGR :  About to open AS_COUNT_CURSOR
10:44:19.029 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:19.029  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:19.029 ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt
for test
10:44:19.030 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
10:44:19.030  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:19.030 ASMGR : Rowcount  = 1
10:44:19.030 ASMGR :  About to close AS_COUNT_CURSOR
10:44:19.031 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:19.031  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:19.031 ASMGR :  About to select from AS_COMMAND_TEST
10:44:19.032 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:19.032  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:19.032 GET_SQL_TEXT : Entered routine get_sql_text
10:44:19.032 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
```

```
10:44:19.032 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:19.032  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:19.032 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:19.044 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:19.044  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:19.044 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:19.045 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.045  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:19.045 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
10:44:19.046 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.046  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:19.046 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:19.046 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:19.046  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:19.046 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:19.046 ASMGR :  Entering sql_placeholder
10:44:19.046 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:19.047 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH,DM_STATUS_CODE WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_STAT_PROJ_ID = ' ' AND
DM_STAT_STATUS_CD = 'IW' AND DM_AUTH_WR_AUTH >= DM_STAT_AUTH
10:44:19.047 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:19.047 ASMGR :  Out of sql_placeholder
10:44:19.047 ASMGR :  About to prepare stmt ROW_CNT
10:44:19.048 ASMGR : Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:19.048  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:19.049 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:19.049 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:19.049  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:19.049 ASMGR :  About to open AS_COUNT_CURSOR
10:44:19.050 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:19.050  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:19.050 ASMGR :  About to do fetch AS_COUNT_CURSOR into
rowcnt for test
10:44:19.051 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
10:44:19.051  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:19.051 ASMGR : Rowcount  = 1
10:44:19.051 ASMGR :  About to close AS_COUNT_CURSOR
10:44:19.051 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:19.051  SQLCODE = 0; #Rows: 1 Comment: Close cursor
```

```
10:44:19.051 ASMGR :  About to select from AS_COMMAND_TEST
10:44:19.052 ASMGR : Return from SQL: Table AS_COMMAND_TEST;
Command SELECT
10:44:19.052  SQLCODE = 0; #Rows: 1 Comment: Get next test for the
command
10:44:19.053 GET_SQL_TEXT : Entered routine get_sql_text
10:44:19.053 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
10:44:19.053 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
10:44:19.053  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
10:44:19.053 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
10:44:19.054 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command OPEN CURSOR
10:44:19.054  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
10:44:19.054 GET_SQL_TEXT :  About to do first fetch from
AS_TEST_CURSOR
10:44:19.055 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.055  SQLCODE = 0; #Rows: 1 Comment: Fetch first row
10:44:19.056 GET_SQL_TEXT :  About to do next fetch from
AS_TEST_CURSOR
10:44:19.056 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command FETCH
10:44:19.056  SQLCODE = 100; #Rows: 1 Comment: Fetch next row
10:44:19.056 GET_SQL_TEXT :  About to close AS_TEST_CURSOR
10:44:19.057 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command CLOSE CURSOR
10:44:19.057  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:19.057 GET_SQL_TEXT : Leaving routine get_sql_text
10:44:19.057 ASMGR :  Entering sql_placeholder
10:44:19.057 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:19.057 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_AUTH_ADMIN_FLAG = 'Y'
10:44:19.058 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:19.058 ASMGR :  Out of sql_placeholder
10:44:19.058 ASMGR :  About to prepare stmt ROW_CNT
10:44:19.059 ASMGR : Return from SQL: Table ROW_CNT; Command
PREPARE
10:44:19.059  SQLCODE = 0; #Rows: 1 Comment: Preparing test
10:44:19.059 ASMGR :  About to declare cursor AS_COUNT_CURSOR for
ROW_CNT
10:44:19.059 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command DCL CURSOR
10:44:19.059  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:19.059 ASMGR :  About to open AS_COUNT_CURSOR
10:44:19.060 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command OPEN CURSOR
10:44:19.060  SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
10:44:19.060 ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt
for test
10:44:19.061 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command FETCH
```

```
10:44:19.061  SQLCODE = 0; #Rows: 1 Comment: First fetch
10:44:19.061 ASMGR : Rowcount  = 1
10:44:19.061 ASMGR :  About to close AS_COUNT_CURSOR
10:44:19.061 ASMGR : Return from SQL: Table AS_COUNT_CURSOR;
Command CLOSE
10:44:19.061  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:19.062 ASMGR : Leaving routine asmgr
10:44:19.062 CHK_STORE_FILE_AUTH : Return from asmgr: with rc of
-1 and ec of 0
10:44:19.062 CHK_STORE_FILE_AUTH : Leaving routine
chk_store_file_auth
10:44:19.062 DM_STORE : Return from chk_store_file_auth: with rc
of 0 and ec of 100
10:44:19.062 DM_STORE :  Entering makerowid
10:44:19.062 MAKEROWID : Entered routine makerowid
10:44:19.063 MAKEROWID : rowid = FPFNDGPNZA
10:44:19.063 MAKEROWID : Leaving routine makerowid
10:44:19.063 DM_STORE : Return from makerowid: with rc of 0 and ec
of 0
10:44:19.063 DM_STORE :  About to insert into DM_FILE_DIRECTORY
10:44:19.072 DM_STORE : Return from SQL: Table DM_FILE_DIRECTORY;
Command INSERT
10:44:19.072  SQLCODE = 0; #Rows: 1 Comment: store file
10:44:19.072 DM_STORE :  entering insert_rev_hist
10:44:19.072 INSERT_REV_HIST : Entered routine INSERT_REV_HIST
10:44:19.072 CHECK_IF_RELEASED : Entered routine check_if_released
10:44:19.073 CHECK_IF_RELEASED :  Select the seqno for the
released status
10:44:19.075 CHECK_IF_RELEASED : Return from SQL: Table
DM_STATUS_CODE; Command SELECT
10:44:19.075  SQLCODE = 0; #Rows: 1 Comment: Select based on
initial_cd and proj_id
10:44:19.075 CHECK_IF_RELEASED :  Select count based on
proj_id,stat_cd,and seqno
10:44:19.077 CHECK_IF_RELEASED : Return from SQL: Table
DM_STATUS_CODE; Command SELECT COUNT
10:44:19.077  SQLCODE = 0; #Rows: 1 Comment: Select count to see
if stat_cd is released
10:44:19.078 CHECK_IF_RELEASED : Leaving routine check_if_released
10:44:19.078 INSERT_REV_HIST : Leaving routine INSERT_REV_HIST
10:44:19.078 DM_STORE : Return from insert_rev_hist: with rc of 0
and ec of 0
10:44:19.078 FMT_MSG_STORE : Entered routine fmt_msg_store
10:44:19.080 FMT_MSG_STORE : message  = CDMCPY172I The file has
been stored.
10:44:19.081 FMT_MSG_STORE : Leaving routine fmt_msg_store
10:44:19.081 DM_STORE : Leaving routine dm_store
10:44:19.081 SET_ATTRIBS : Return from DM_STORE: with rc of 0 and
ec of 0
10:44:19.081 SET_ATTRIBS : Leaving routine SET_ATTRIBS
10:44:19.081 PROCESS_FILE : Return from SET_ATTRIBS: with rc of 0
and ec of 0
10:44:19.081 PROCESS_FILE :  Entering format_and_send_message
```

```
10:44:19.081 FORMAT_AND_SEND_MESSAGE : Entered routine
FORMAT_AND_SEND_MESSAGE
10:44:19.081 FORMAT_AND_SEND_MESSAGE :  Entering SEND_MESSAGE
10:44:19.081 SEND_MESSAGE : Entered routine SEND_MESSAGE
10:44:19.081 GET_MESSAGE_TYPE : Entered routine get_message_type
10:44:19.082 GET_MESSAGE_TYPE : Leaving routine get_message_type
10:44:19.082 SEND_MESSAGE : send_message_type = 2
10:44:19.082 SEND_MESSAGE : Return from get_message_type: with rc
of 0 and ec of 0
10:44:19.082 SEND_MESSAGE :  Entering D_MESSAGE
10:44:19.082 CALL_LM_TIMER : Entered routine call_lm_timer
10:44:19.082 CALL_LM_TIMER :  license Y/N   = 1
10:44:19.083 CALL_LM_TIMER : timestamp   = 14960472
10:44:19.083 CALL_LM_TIMER : timenow  = 14960474
10:44:19.083 CALL_LM_TIMER : Leaving routine call_lm_timer
10:44:19.084 SEND_MESSAGE : Return from D_MESSAGE: with rc of 0 and
ec of 0
10:44:19.084 EDM_READMESSAGES : Entered routine edm_readmessages
10:44:19.084 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
10:44:19.084 EDM_READMESSAGES : Return from D_CONSUME_NEXT: with
rc of 25024 and ec of 0
10:44:19.085 EDM_READMESSAGES : Leaving routine edm_readmessages
10:44:19.085 SEND_MESSAGE : Leaving routine SEND_MESSAGE
10:44:19.085 FORMAT_AND_SEND_MESSAGE : Return from SEND_MESSAGE:
with rc of 0 and ec of 0
10:44:19.085 FORMAT_AND_SEND_MESSAGE :  send message to log
10:44:19.085 FORMAT_AND_SEND_MESSAGE : Leaving routine
FORMAT_AND_SEND_MESSAGE
10:44:19.085 PROCESS_FILE : Return from FORMAT_AND_SEND_MESSAGE:
with rc of 0 and ec of 0
10:44:19.085 PROCESS_FILE : Leaving routine PROCESS_FILE
10:44:19.085 COPY_FILE : Return from PROCESS_FILE: with rc of 0 and
ec of 0
10:44:19.085 COPY_FILE :  Entering UPDATE_SOURCE_FILE
10:44:19.085 UPDATE_SOURCE_FILE : Entered routine
update_source_file
10:44:19.086 UPDATE_SOURCE_FILE :  Updating DM_FILE_DIRECTORY
date/time
10:44:19.095 UPDATE_SOURCE_FILE : Return from SQL: Table
DM_FILE_DIRECTORY; Command UPDATE
10:44:19.095  SQLCODE = 0; #Rows: 1 Comment: date/time
10:44:19.095 UPDATE_SOURCE_FILE : Leaving routine
update_source_file
10:44:19.095 COPY_FILE : Return from UPDATE_SOURCE_FILE: with rc
of 0 and ec of 0
10:44:19.095 COPY_FILE : Leaving routine COPY_FILE
10:44:19.095 DM_COPY : Return from COPY_FILE: with rc of 0 and ec
of 0
10:44:19.095 DM_COPY :  Entering commit_only
10:44:19.095 SQL_COMMIT_ONLY : Entered routine sql_commit_only
10:44:19.095 SQL_COMMIT_ONLY :  About to SQL COMMIT WORK
10:44:19.117 SQL_COMMIT_ONLY : Return from SQL: Table  ; Command
COMMIT
10:44:19.117  SQLCODE = 0; #Rows: 1 Comment: After commit
```

```
10:44:19.117 SQL_COMMIT_ONLY : Leaving routine sql_commit_only
10:44:19.117 DM_COPY : Return from commit_only: with rc of 0 and
ec of 0
10:44:19.117 DM_COPY :  Entering SEND_MESSAGE
10:44:19.117 SEND_MESSAGE : Entered routine SEND_MESSAGE
10:44:19.117 GET_MESSAGE_TYPE : Entered routine get_message_type
10:44:19.117 GET_MESSAGE_TYPE : Leaving routine get_message_type
10:44:19.117 SEND_MESSAGE : send_message_type = 2
10:44:19.117 SEND_MESSAGE : Return from get_message_type: with rc
of 0 and ec of 0
10:44:19.118 SEND_MESSAGE :  Entering D_MESSAGE
10:44:19.130 SEND_MESSAGE : Return from D_MESSAGE: with rc of 0
and ec of 0
10:44:19.131 EDM_READMESSAGES : Entered routine edm_readmessages
10:44:19.131 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
10:44:19.131 EDM_READMESSAGES : Return from D_CONSUME_NEXT: with
rc of 25024 and ec of 0
10:44:19.131 EDM_READMESSAGES : Leaving routine edm_readmessages
10:44:19.131 SEND_MESSAGE : Leaving routine SEND_MESSAGE
10:44:19.131 DM_COPY : Return from SEND_MESSAGE: with rc of 0 and
ec of 0
10:44:19.131 DM_COPY : Leaving routine dm_copy
10:44:19.131 INVOKE_DM_FUNCTION : Return from COPY    : with rc of
0 and ec of 0
10:44:19.131 INVOKE_DM_FUNCTION :  Entering SQL_COMMIT_ONLY
10:44:19.131 SQL_COMMIT_ONLY : Entered routine sql_commit_only
10:44:19.132 SQL_COMMIT_ONLY :  About to SQL COMMIT WORK
10:44:19.132 SQL_COMMIT_ONLY : Return from SQL: Table  ; Command
COMMIT
10:44:19.132  SQLCODE = 0; #Rows: 1 Comment: After commit
10:44:19.133 SQL_COMMIT_ONLY : Leaving routine sql_commit_only
10:44:19.133 INVOKE_DM_FUNCTION : Return from SQL_COMMIT_ONLY:
with rc of 0 and ec of 0
10:44:19.133 INVOKE_DM_FUNCTION : Leaving routine
INVOKE_DM_FUNCTION
10:44:19.133 PROCESS_MESSAGES : Return from INVOKE_DM_FUNCTION:
with rc of 0 and ec of 0
10:44:19.133 PROCESS_MESSAGES : Leaving routine PROCESS_MESSAGES
10:44:19.133 PROCESS_MESSAGES : Entered routine PROCESS_MESSAGES
10:44:19.133 PROCESS_MESSAGES :  Entering RECEIVE_MESSAGE
10:44:19.133 RECEIVE_MESSAGE : Entered routine RECEIVE_MESSAGE
10:44:19.133 RECEIVE_MESSAGE :  edm_getmessages
10:44:19.134 EDM_GETMESSAGES : Entered routine edm_getmessages
10:44:19.134 EDM_GETMESSAGES :  edm_readmessages
10:44:19.134 EDM_READMESSAGES : Entered routine edm_readmessages
10:44:19.134 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
10:44:19.134 EDM_READMESSAGES : Return from D_CONSUME_NEXT: with
rc of 25024 and ec of 0
10:44:19.134 EDM_READMESSAGES : Leaving routine edm_readmessages
10:44:19.134 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
10:44:19.134 EDM_GETMESSAGES : rec->conlist[0] = -1
10:44:19.134 EDM_GETMESSAGES :  edm_readmessages
10:44:19.135 EDM_READMESSAGES : Entered routine edm_readmessages
```

```
10:44:19.135 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
10:44:19.172 EDM_READMESSAGES : Return from D_CONSUME_NEXT: with
rc of 0 and ec of 315
10:44:19.172 EDM_READMESSAGES :  MESSAGE_INDICATE
10:44:19.172 EDM_READMESSAGES : conid = 5
10:44:19.172 EDM_READMESSAGES : Leaving routine edm_readmessages
10:44:19.172 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
10:44:19.173 EDM_GETMESSAGES : rec->conlist[0] = -1
10:44:19.173 EDM_GETMESSAGES :  Any connection. Got a message
10:44:19.173 EDM_GETMESSAGES : conid = 5
10:44:19.173 EDM_GETMESSAGES : Leaving routine edm_getmessages
10:44:19.173 RECEIVE_MESSAGE : Return from edm_getmessages: with
rc of 0 and ec of 0
10:44:19.173 GET_NORM_MESSAGE_TYPE : Entered routine
GET_NORM_MESSAGE_TYPE
10:44:19.173 GET_NORM_MESSAGE_TYPE : Leaving routine
GET_NORM_MESSAGE_TYPE
10:44:19.173 RECEIVE_MESSAGE : received_message_type = 2
10:44:19.173 RECEIVE_MESSAGE : Leaving routine RECEIVE_MESSAGE
10:44:19.174 PROCESS_MESSAGES : Return from RECEIVE_MESSAGE: with
rc of 0 and ec of 0
10:44:19.174 PROCESS_MESSAGES :  Entering INVOKE_DM_FUNCTION
10:44:19.174 INVOKE_DM_FUNCTION : Entered routine
INVOKE_DM_FUNCTION
10:44:19.174 INVOKE_DM_FUNCTION : About to enter:  = STORE
10:44:19.174 DM_STORE : Entered routine dm_store
10:44:19.174 DM_STORE :  Entering internal select_directory
10:44:19.174 SELECT_DIRECTORY : Entered routine select_directory
10:44:19.174 SELECT_DIRECTORY :  About to select from
DM_FILE_DIRECTORY
10:44:19.178 SELECT_DIRECTORY : Return from SQL: Table
DM_FILE_DIRECTORY; Command SELECT
10:44:19.178  SQLCODE = 0; #Rows: 1 Comment: select row
10:44:19.179 SELECT_DIRECTORY :  About to select from DM_PART_FILE
10:44:19.181 SELECT_DIRECTORY : Return from SQL: Table
DM_PART_FILE; Command SELECT
10:44:19.181  SQLCODE = 100; #Rows: 0 Comment: part_rowid
10:44:19.181 SELECT_DIRECTORY : Leaving routine select_directory
10:44:19.181 DM_STORE : Return from select_directory: with rc of 0
and ec of 0
10:44:19.181 FINDPOOL : Entered routine findpool
10:44:19.181 FINDPOOL :  Entering find_spool
10:44:19.181 FIND_SPOOL : Entered routine find_spool
10:44:19.181 FIND_SPOOL :  Entering get_pool_query
10:44:19.182 GET_POOL_QUERY : Entered routine get_pool_query
10:44:19.182 GET_POOL_QUERY : file rowid = FPFNDGPNZA
10:44:19.182 GET_POOL_QUERY :  SELECT 1st sequence number from
DM_POOL_QUEST
10:44:19.184 GET_POOL_QUERY : Return from SQL: Table
DM_POOL_QUEST; Command SELECT
10:44:19.184  SQLCODE = 0; #Rows: 1 Comment: Get 1st seqno
10:44:19.184 GET_POOL_QUERY :  Retrieving DM_POOL_QUEST row
```

```
10:44:19.186 GET_POOL_QUERY : Return from SQL: Table
DM_POOL_QUEST; Command SELECT
10:44:19.186  SQLCODE = 0; #Rows: 1 Comment: Get quest step info
10:44:19.186 GET_POOL_QUERY :  Entering fetch_query_text
10:44:19.187 FETCH_QUERY_TEXT : Entered routine fetch_query_text
10:44:19.187 FETCH_QUERY_TEXT :  Declaring cursor SRCH_QUERY_TEXT
10:44:19.187 FETCH_QUERY_TEXT : Return from SQL: Table
DM_SRCH_QUERY; Command dcl cursor
10:44:19.187  SQLCODE = 0; #Rows: 1 Comment: SRCH_QUERY_TEXT
10:44:19.187 FETCH_QUERY_TEXT :  Opening cursor SRCH_QUERY_TEXT
10:44:19.189 FETCH_QUERY_TEXT : Return from SQL: Table
DM_SRCH_QUERY; Command open cursor
10:44:19.189  SQLCODE = 0; #Rows: 0 Comment: SRCH_QUERY_TEXT
10:44:19.189 FETCH_QUERY_TEXT :  Fetching query text
10:44:19.190 FETCH_QUERY_TEXT : Return from SQL: Table
DM_SRCH_QUERY; Command fetch cursor
10:44:19.190  SQLCODE = 0; #Rows: 1 Comment: SRCH_QUERY_TEXT
10:44:19.190 FETCH_QUERY_TEXT :  Fetching query text
10:44:19.191 FETCH_QUERY_TEXT : Return from SQL: Table
DM_SRCH_QUERY; Command fetch cursor
10:44:19.191  SQLCODE = 100; #Rows: 1 Comment: SRCH_QUERY_TEXT
10:44:19.191 FETCH_QUERY_TEXT :  Closing cursor SRCH_QUERY_TEXT
10:44:19.191 FETCH_QUERY_TEXT : Return from SQL: Table
DM_SRCH_QUERY; Command close cursor
10:44:19.191  SQLCODE = 0; #Rows: 1 Comment: SRCH_QUERY_TEXT
10:44:19.191 FETCH_QUERY_TEXT : Leaving routine fetch_query_text
10:44:19.191 GET_POOL_QUERY : Return from fetch_query_text: with
rc of 0 and ec of 0
10:44:19.192 GET_POOL_QUERY :  Entering sql_placeholder
10:44:19.192 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:19.192 SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_STORAGE_POOL WHERE DM_POOL_PCT_USED < 50 AND DM_POOL_STATUS =
'0' AND DM_POOL_NAME <> (SELECT DM_FILE_POOL_NAME FROM
DM_FILE_DIRECTORY WHERE DM_FILE_ROWID = 'FPFNDGPNZA')
10:44:19.192 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:19.192 GET_POOL_QUERY : Return from sql_placeholder: with rc
of 0 and ec of 0
10:44:19.192 GET_POOL_QUERY :  Entering get_rowcount
10:44:19.192 GET_ROWCOUNT : Entered routine get_rowcount
10:44:19.192 GET_ROWCOUNT :  About to prepare stmt ROW_CNT
10:44:19.200 GET_ROWCOUNT : Return from SQL: Table ROW_CNT;
Command PREPARE
10:44:19.200  SQLCODE = 0; #Rows: 1 Comment: Preparing SELECT
COUNT(*) query
10:44:19.200 GET_ROWCOUNT :  Declaring cursor POOL_SRCH_CURSOR for
ROW_CNT
10:44:19.200 GET_ROWCOUNT : Return from SQL: Table
POOL_SRCH_CURSOR; Command dcl cursor
10:44:19.200  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
ROW_CNT
10:44:19.201 GET_ROWCOUNT :  Opening cursor POOL_SRCH_CURSOR
10:44:19.201 GET_ROWCOUNT : Return from SQL: Table
POOL_SRCH_CURSOR; Command OPEN CURSOR
10:44:19.201  SQLCODE = 0; #Rows: 0 Comment: Opening cursor
```

```
10:44:19.202 GET_ROWCOUNT :  Fetching POOL_SRCH_CURSOR into rowcnt
for query
10:44:19.203 GET_ROWCOUNT : Return from SQL: Table
POOL_SRCH_CURSOR; Command FETCH
10:44:19.203  SQLCODE = 0; #Rows: 1 Comment: First and only fetch
10:44:19.203 GET_ROWCOUNT : Rowcount  = 0
10:44:19.204 GET_ROWCOUNT :  Closing cursor POOL_SRCH_CURSOR
10:44:19.204 GET_ROWCOUNT : Return from SQL: Table
POOL_SRCH_CURSOR; Command CLOSE
10:44:19.204  SQLCODE = 0; #Rows: 1 Comment: Close cursor
10:44:19.204 GET_ROWCOUNT : Leaving routine get_rowcount
10:44:19.204 GET_POOL_QUERY : Return from get_rowcount: with rc of
0 and ec of 0
10:44:19.204 GET_POOL_QUERY : Leaving routine get_pool_query
10:44:19.204 FIND_SPOOL : Return from get_pool_query: with rc of 0
and ec of 0
10:44:19.204 FIND_SPOOL :  Entering fetch_where_text
10:44:19.204 FETCH_WHERE_TEXT : Entered routine fetch_where_text
10:44:19.204 FETCH_WHERE_TEXT :  Declaring cursor POOL_QUERY_TEXT
10:44:19.205 FETCH_WHERE_TEXT : Return from SQL: Table
DM_POOL_QUERY; Command dcl cursor
10:44:19.205  SQLCODE = 0; #Rows: 1 Comment: POOL_QUERY_TEXT
10:44:19.205 FETCH_WHERE_TEXT :  Opening cursor POOL_QUERY_TEXT
10:44:19.207 FETCH_WHERE_TEXT : Return from SQL: Table
DM_POOL_QUERY; Command open cursor
10:44:19.207  SQLCODE = 0; #Rows: 0 Comment: POOL_QUERY_TEXT
10:44:19.207 FETCH_WHERE_TEXT :  Fetching query text
10:44:19.208 FETCH_WHERE_TEXT : Return from SQL: Table
DM_POOL_QUERY; Command fetch cursor
10:44:19.208  SQLCODE = 0; #Rows: 1 Comment: POOL_QUERY_TEXT
10:44:19.208 FETCH_WHERE_TEXT :  Fetching query text
10:44:19.209 FETCH_WHERE_TEXT : Return from SQL: Table
DM_POOL_QUERY; Command fetch cursor
10:44:19.209  SQLCODE = 100; #Rows: 1 Comment: POOL_QUERY_TEXT
10:44:19.209 FETCH_WHERE_TEXT :  Closing cursor POOL_QUERY_TEXT
10:44:19.209 FETCH_WHERE_TEXT : Return from SQL: Table
DM_POOL_QUERY; Command close cursor
10:44:19.209  SQLCODE = 0; #Rows: 1 Comment: POOL_QUERY_TEXT
10:44:19.209 FETCH_WHERE_TEXT : Leaving routine fetch_where_text
10:44:19.209 FIND_SPOOL : Return from fetch_where_text: with rc of
0 and ec of 0
10:44:19.210 FIND_SPOOL :  Entering sql_placeholder
10:44:19.210 SQL_PLACEHOLDER : Entered routine sql_placeholder
10:44:19.210 SQL_PLACEHOLDER : parsed text = WHERE
DM_POOL_PCT_USED < 50 AND DM_POOL_NAME NOT IN (SELECT
DM_FILE_POOL_NAME FROM DM_FILE_DIRECTORY WHERE DM_FILE_ROWID =
'FPFNDGPNZA') ORDER BY DM_POOL_PCT_USED
10:44:19.210 SQL_PLACEHOLDER : Leaving routine sql_placeholder
10:44:19.210 FIND_SPOOL : Return from sql_placeholder: with rc of
0 and ec of 0
10:44:19.210 FIND_SPOOL : pool select query = SELECT

DM_POOL_NAME,DM_POOL_NO_FILES,DM_POOL_SIZE,DM_POOL_SIZE_USED,DM_P
OOL_SIZE_FREE,DM_POOL_PCT_USED,DM_POOL_PCT_FREE,DM_POOL_NEXT_FILE
```

```
,DM_POOL_STATUS,DM_POOL_USER_ID,DM_POOL_DATE_WR,DM_POOL_TIME_WR
FROM DM_STORAGE_POOL WHERE DM_POOL_PCT_USED < 50 AND DM_POOL_NAME
NOT IN (SELECT DM_FILE_POOL_NAME FROM DM_FILE_DIRECTORY WHERE
DM_FILE_ROWID = 'FPFNDGPNZA') ORDER BY DM_POOL_PCT_USED

10:44:19.210 FIND_SPOOL :  About to prepare stmt POOL_QUERY
10:44:19.218 FIND_SPOOL : Return from SQL: Table POOL_QUERY;
Command PREPARE
10:44:19.218  SQLCODE = 0; #Rows: 1 Comment: Preparing pool
selection query
10:44:19.218 FIND_SPOOL :  Declaring cursor FIND_POOL_CURSOR for
POOL_QUERY
10:44:19.218 FIND_SPOOL : Return from SQL: Table FIND_POOL_CURSOR;
Command dcl cursor
10:44:19.218  SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt
POOL_QUERY
10:44:19.218 FIND_SPOOL :  Opening cursor FIND_POOL_CURSOR
10:44:19.219 FIND_SPOOL : Return from SQL: Table DM_STORAGE_POOL;
Command OPEN CURSOR
10:44:19.219  SQLCODE = 0; #Rows: 0 Comment: FIND_POOL_CURSOR
10:44:19.219 FIND_SPOOL :  Fetching pool row
10:44:19.221 FIND_SPOOL : Return from SQL: Table DM_STORAGE_POOL;
Command fetch cursor
10:44:19.221  SQLCODE = 0; #Rows: 1 Comment: FIND_POOL_CURSOR
10:44:19.221 FIND_SPOOL :  Closing cursor FIND_POOL_CURSOR
10:44:19.221 FIND_SPOOL : Return from SQL: Table DM_STORAGE_POOL;
Command CLOSE CURSOR
10:44:19.221  SQLCODE = 0; #Rows: 1 Comment: FIND_POOL_CURSOR
10:44:19.221 FIND_SPOOL :  Retrieving pool info
10:44:19.223 FIND_SPOOL : Return from SQL: Table DM_POOL_INFO;
Command SELECT
10:44:19.223  SQLCODE = 0; #Rows: 1 Comment: pool info
10:44:19.224 FIND_SPOOL :  About to update DM_STORAGE_POOL
10:44:19.227 FIND_SPOOL : Return from SQL: Table DM_STORAGE_POOL;
Command UPDATE
10:44:19.227  SQLCODE = 0; #Rows: 1 Comment: Update storage pool
10:44:19.227 FIND_SPOOL : Leaving routine find_spool
10:44:19.227 FINDPOOL : Return from find_spool: with rc of 0 and
ec of 0
10:44:19.227 FINDPOOL :  Entering make_sp_filename
10:44:19.227 FINDPOOL : Return from make_sp_filename: with rc of 0
and ec of 0
10:44:19.227 FINDPOOL : Leaving routine findpool
10:44:19.228 DM_STORE :  About to update DM_FILE_DIRECTORY
10:44:19.232 DM_STORE : Return from SQL: Table DM_FILE_DIRECTORY;
Command UPDATE
10:44:19.232  SQLCODE = 0; #Rows: 1 Comment: -1 version
10:44:19.232 FMT_MSG_STORE : Entered routine fmt_msg_store
10:44:19.235 FMT_MSG_STORE :  Entering send_message
10:44:19.235 SEND_MESSAGE : Entered routine SEND_MESSAGE
10:44:19.235 GET_MESSAGE_TYPE : Entered routine get_message_type
10:44:19.236 GET_MESSAGE_TYPE : Leaving routine get_message_type
10:44:19.236 SEND_MESSAGE : send_message_type = 2
```

```
10:44:19.236 SEND_MESSAGE : Return from get_message_type: with rc
of 0 and ec of 0
10:44:19.236 SEND_MESSAGE :  Entering D_MESSAGE
10:44:19.244 SEND_MESSAGE : Return from D_MESSAGE: with rc of 0 and
ec of 0
10:44:19.244 EDM_READMESSAGES : Entered routine edm_readmessages
10:44:19.244 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
10:44:19.244 EDM_READMESSAGES : Return from D_CONSUME_NEXT: with
rc of 25024 and ec of 0
10:44:19.244 EDM_READMESSAGES : Leaving routine edm_readmessages
10:44:19.244 SEND_MESSAGE : Leaving routine SEND_MESSAGE
10:44:19.245 FMT_MSG_STORE : Return from send_message: with rc of
0 and ec of 0
10:44:19.245 FMT_MSG_STORE : message  = CDMCPY172I The file has
been stored.
10:44:19.245 FMT_MSG_STORE : Leaving routine fmt_msg_store
10:44:19.245 DM_STORE : Leaving routine dm_store
10:44:19.245 INVOKE_DM_FUNCTION : Return from STORE   : with rc of
0 and ec of 0
10:44:19.245 INVOKE_DM_FUNCTION :  Entering SQL_COMMIT_ONLY
10:44:19.246 SQL_COMMIT_ONLY : Entered routine sql_commit_only
10:44:19.246 SQL_COMMIT_ONLY :  About to SQL COMMIT WORK
10:44:19.274 SQL_COMMIT_ONLY:Return from SQL: Table;Command COMMIT
10:44:19.274  SQLCODE = 0; #Rows: 1 Comment: After commit
10:44:19.275 SQL_COMMIT_ONLY : Leaving routine sql_commit_only
10:44:19.275 INVOKE_DM_FUNCTION : Return from SQL_COMMIT_ONLY:
with rc of 0 and ec of 0
10:44:19.275 INVOKE_DM_FUNCTION : Leaving routine
INVOKE_DM_FUNCTION
10:44:19.275 PROCESS_MESSAGES : Return from INVOKE_DM_FUNCTION:
with rc of 0 and ec of 0
10:44:19.275 PROCESS_MESSAGES : Leaving routine PROCESS_MESSAGES
10:44:19.275 PROCESS_MESSAGES : Entered routine PROCESS_MESSAGES
10:44:19.275 PROCESS_MESSAGES :  Entering RECEIVE_MESSAGE
10:44:19.275 RECEIVE_MESSAGE : Entered routine RECEIVE_MESSAGE
10:44:19.275 RECEIVE_MESSAGE :  edm_getmessages
10:44:19.276 EDM_GETMESSAGES : Entered routine edm_getmessages
10:44:19.276 EDM_GETMESSAGES :  edm_readmessages
10:44:19.276 EDM_READMESSAGES : Entered routine edm_readmessages
10:44:19.276 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
10:44:19.276 EDM_READMESSAGES : Return from D_CONSUME_NEXT: with
rc of 25024 and ec of 0
10:44:19.276 EDM_READMESSAGES : Leaving routine edm_readmessages
10:44:19.276 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
10:44:19.276 EDM_GETMESSAGES : rec->conlist[0] = -1
10:44:19.276 EDM_GETMESSAGES :  edm_readmessages
10:44:19.277 EDM_READMESSAGES : Entered routine edm_readmessages
10:44:19.278 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
10:44:19.548 EDM_READMESSAGES : Return from D_CONSUME_NEXT: with
rc of 0 and ec of 315
10:44:19.548 EDM_READMESSAGES :  MESSAGE_INDICATE
10:44:19.548 EDM_READMESSAGES : conid = 5
10:44:19.549 EDM_READMESSAGES : Leaving routine edm_readmessages
```

```
10:44:19.549 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
10:44:19.549 EDM_GETMESSAGES : rec->conlist[0] = -1
10:44:19.549 EDM_GETMESSAGES :  Any connection. Got a message
10:44:19.549 EDM_GETMESSAGES : conid = 5
10:44:19.549 EDM_GETMESSAGES : Leaving routine edm_getmessages
10:44:19.549 RECEIVE_MESSAGE : Return from edm_getmessages: with
rc of 0 and ec of 0
10:44:19.549 GET_NORM_MESSAGE_TYPE : Entered routine
GET_NORM_MESSAGE_TYPE
10:44:19.549 GET_NORM_MESSAGE_TYPE : Leaving routine
GET_NORM_MESSAGE_TYPE
10:44:19.549 RECEIVE_MESSAGE : received_message_type = 2
10:44:19.550 RECEIVE_MESSAGE : Leaving routine RECEIVE_MESSAGE
10:44:19.550 PROCESS_MESSAGES : Return from RECEIVE_MESSAGE: with
rc of 0 and ec of 0
10:44:19.550 PROCESS_MESSAGES :  Entering INVOKE_DM_FUNCTION
10:44:19.550 INVOKE_DM_FUNCTION : Entered routine
INVOKE_DM_FUNCTION
10:44:19.550 INVOKE_DM_FUNCTION : About to enter:  = REVOKE
10:44:19.550 DM_REVOKE : Entered routine dm_revoke
10:44:19.550 DM_REVOKE : Command to revoke:  = STORE
10:44:19.551 DM_REVOKE : Selection scope:    = F
10:44:19.551 DM_REVOKE : This item type:     = F
10:44:19.551 DM_REVOKE : This file name:     = vlt_stup2
10:44:19.551 DM_REVOKE : EOT switch (ON=Y):  = N
10:44:19.551 DM_REVOKE : Revoke switch (GOOD=Y):  = Y
10:44:19.551 FINISH_STORE : Entered routine finish_store
10:44:19.551 FINISH_STORE :  about to select from
DM_FILE_DIRECTORY
10:44:19.556 FINISH_STORE : Return from SQL: Table
DM_FILE_DIRECTORY; Command SELECT
10:44:19.556  SQLCODE = 100; #Rows: 0 Comment: select from
dm_file_directory
10:44:19.556 FINISH_STORE :  about to update DM_FILE_DIRECTORY
10:44:19.787 FINISH_STORE : Return from SQL: Table
DM_FILE_DIRECTORY; Command UPDATE
10:44:19.787  SQLCODE = 0; #Rows: 1 Comment: update
dm_file_directory
10:44:19.787 FINISH_STORE :  entering insert_backup
10:44:19.788 INSERT_BACKUP : Entered routine insert_backup
10:44:19.788 INSERT_BACKUP :  About to select from
DM_FILE_DIRECTORY
10:44:19.792 INSERT_BACKUP : Return from SQL: Table
DM_FILE_DIRECTORY; Command SELECT
10:44:19.792  SQLCODE = 0; #Rows: 1 Comment: file_directory table
10:44:19.792 INSERT_BACKUP :  About to insert in DM_FILE_BACKUP
10:44:19.805 INSERT_BACKUP : Return from SQL: Table
DM_FILE_BACKUP; Command INSERT
10:44:19.805  SQLCODE = 0; #Rows: 1 Comment: insert into
dm_file_backup table
10:44:19.805 INSERT_BACKUP : Leaving routine insert_backup
10:44:19.805 FINISH_STORE : Return from insert_backup: with rc of
0 and ec of 0
```

```
10:44:19.805 FINISH_STORE :  entering update_storage_pool
10:44:19.805 UPDATE_STORAGE_POOL : Entered routine
update_storage_pool
10:44:19.805 UPDATE_STORAGE_POOL :  entering pool_count
10:44:19.806 UPDATE_STORAGE_POOL : Return from pool_count: with rc
of 0 and ec of 0
10:44:19.806 UPDATE_STORAGE_POOL :  entering updt_pl_tbl
10:44:19.806 UPDT_PL_TBL : Entered routine updt_pl_tbl
10:44:19.806 UPDT_PL_TBL :  Update the row contents in table -
DM_STORAGE_POOL
10:44:19.809 UPDT_PL_TBL : Return from SQL: Table DM_STORAGE_POOL;
Command UPDATE
10:44:19.809  SQLCODE = 0; #Rows: 1 Comment: Update the row
contents
10:44:19.810 UPDT_PL_TBL : Leaving routine updt_pl_tbl
10:44:19.810 UPDATE_STORAGE_POOL : Return from updt_pl_tbl: with
rc of 0 and ec of 0
10:44:19.810 UPDATE_STORAGE_POOL : Leaving routine
update_storage_pool
10:44:19.810 FINISH_STORE : Return from update_storage_pool: with
rc of 0 and ec of 0
10:44:19.810 FINISH_STORE : Leaving routine finish_store
10:44:19.810 DM_REVOKE :  send_back_reply
10:44:19.810 SEND_BACK_REPLY : Entered routine send_back_reply
10:44:19.810 SEND_BACK_REPLY :  Entering send_message
10:44:19.810 SEND_MESSAGE : Entered routine SEND_MESSAGE
10:44:19.811 GET_MESSAGE_TYPE : Entered routine get_message_type
10:44:19.811 GET_MESSAGE_TYPE : Leaving routine get_message_type
10:44:19.811 SEND_MESSAGE : send_message_type = 2
```

NT

```
12:37:44.564 DM_CNTL : Entered routine DM_CNTL
DM_CNTL :  init_global_var
INIT_GLOBAL_VAR : Entered routine init_global_var
12:37:44.564 INIT_GLOBAL_VAR :  Entering get_evar
INIT_GLOBAL_VAR: Return from get_evar: with rc of 31020 and ec of 0
INIT_GLOBAL_VAR :  About to enter pdm_xtract
PDM_XTRACT : Entered routine pdm_xtract
PDM_XTRACT :  About to bind with process manager
PDM_XTRACT:Return from Returned from bind: with rc of 0 and ec of 0
PDM_XTRACT :  About to extract information from the configuration
file.
PDM_XTRACT : Return from Extract results: with rc of 0 and ec of 0
PDM_XTRACT :  Entering parse routine exoutprt
12:37:44.654 EXOUTPRT : Entered routine exoutprt
EXOUTPRT : Leaving routine exoutprt
PDM_XTRACT : Return from Exit parse routine exoutprt: with rc of 0
and ec of 0
PDM_XTRACT :  Entering unbind to process manager
12:37:44.684 PDM_XTRACT : Return from exit unbind: with rc of 0 and
ec of 0
```

```
PDM_XTRACT : Leaving routine pdm_xtract
INIT_GLOBAL_VAR : Return from pdm_xtract: with rc of 0 and ec of 0
INIT_GLOBAL_VAR : Leaving routine init_global_var
DM_CNTL : Return from init_global_var: with rc of 0 and ec of 0
DM_CNTL :  getEdmLicense
GETEDMLICENSE : Entered routine getEdmLicense
ENSURE_LMS_INIT : Entered routine ensure_lms_init
ENSURE_LMS_INIT : Leaving routine ensure_lms_init
GETVAULTLICENSE : Entered routine getVaultLicense
ENSURE_LMS_INIT : Entered routine ensure_lms_init
ENSURE_LMS_INIT : Leaving routine ensure_lms_init
12:37:44.905 DETERMINELICENSENEEDED : Entered routine
determineLicenseNeeded
GETEDMLICENSE : Leaving routine getEdmLicense
DM_CNTL : Return from getEdmLicense: with rc of 0 and ec of 0
DM_CNTL :  Entering sql_connect
DBS_PSCONNECT : Entered routine dbs_psconnect
DBS_PSCONNECT : userid = PDMDM
DBS_PSCONNECT : userpw = PDMDM
DBS_PSCONNECT : Leaving routine dbs_psconnect
DM_CNTL : Return from sql_connect: with rc of 0 and ec of 0
DM_CNTL :  Entering ACTIVATE_SERVER
ACTIVATE_SERVER : Entered routine ACTIVATE_SERVER
ACTIVATE_SERVER :  Entering ACTIVATE
12:37:45.105 ACTIVATE_SERVER : Return from ACTIVATE: with rc of 0
and ec of 0
12:37:45.105 ACTIVATE_SERVER : Leaving routine ACTIVATE_SERVER
DM_CNTL : Return from ACTIVATE_SERVER: with rc of 0 and ec of 0
LOG_START_STOP_MSG : Entered routine log_start_stop_msg
LOG_START_STOP_MSG :  Entering get_os_userid
LOG_START_STOP_MSG : Return from get_os_userid: with rc of 0 and
ec of 0
CI_DISPLAY_MSG : Entered routine ci_display_msg
CI_DISPLAY_MSG :   Entering display_write
EDM data management process dhumketu:DHUMKETU:PDMDM:0 started.
CI_DISPLAY_MSG:Return from display_write: with rc of 0 and ec of 0
CI_DISPLAY_MSG : Leaving routine ci_display_msg
SENDMSG_TO_LOG : Entered routine SENDMSG_TO_LOG
SENDMSG_TO_LOG :  Entering CONNECT_TO_LOG
12:37:45.145 CONNECT_TO_LOG : Entered routine CONNECT_TO_LOG
CONNECT_TO_LOG :  REQUEST_CONNECT
12:37:45.145 REQUEST_CONNECT : Entered routine REQUEST_CONNECT
12:37:45.145 REQUEST_CONNECT :  Entering D_INIT_REQ
REQUEST_CONNECT : Return from D_INIT_REQ: with rc of 0 and ec of 0
REQUEST_CONNECT :  D_WAIT_EVENT
12:37:45.165 REQUEST_CONNECT : Return from D_WAIT_EVENT: with rc
of 0 and ec of 0
12:37:45.165 REQUEST_CONNECT :  Entering D_COMSUME_EVENT
12:37:45.165 REQUEST_CONNECT : Return from D_COMSUME_EVENT: with
rc of 0 and ec of 0
12:37:45.165 REQUEST_CONNECT : Leaving routine REQUEST_CONNECT
CONNECT_TO_LOG:Return from REQUEST_CONNECT:with rc of 0 and ec of
0
CONNECT_TO_LOG : Leaving routine CONNECT_TO_LOG
```

```
SENDMSG_TO_LOG:Return from CONNECT_TO_LOG: with rc of 0 and ec of 0
SENDMSG_TO_LOG :  Entering SEND_MESSAGE
12:37:45.165 SEND_MESSAGE : Entered routine SEND_MESSAGE
GET_MESSAGE_TYPE : Entered routine get_message_type
GET_MESSAGE_TYPE : Leaving routine get_message_type
12:37:45.165 SEND_MESSAGE : send_message_type = 2
12:37:45.165 SEND_MESSAGE : Return from get_message_type: with rc
of 0 and ec of 0
TRANSLATE_MESSAGE : Entered routine TRANSLATE_MESSAGE
TRANSLATE_MESSAGE :  Entering hexchar
TRANSLATE_MESSAGE : Return from hexchar: with rc of 0 and ec of 0
TRANSLATE_MESSAGE :  Entering NETWORK_CHARS
TRANSLATE_MESSAGE : Return from NETWORK_CHARS: with rc of 0 and ec
of 0
TRANSLATE_MESSAGE : Leaving routine TRANSLATE_MESSAGE
SEND_MESSAGE : Return from translate_to_network: with rc of 0 and
ec of 0
SEND_MESSAGE :  Entering D_MESSAGE
12:37:45.165 SEND_MESSAGE : Return from D_MESSAGE: with rc of 0 and
ec of 0
12:37:45.165 EDM_READMESSAGES : Entered routine edm_readmessages
EDM_READMESSAGES :  Q initialized
12:37:45.175 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
12:37:45.175 SEND_MESSAGE : Leaving routine SEND_MESSAGE
SENDMSG_TO_LOG : Return from SEND_MESSAGE: with rc of 0 and ec of 0
SENDMSG_TO_LOG : Leaving routine SENDMSG_TO_LOG
LOG_START_STOP_MSG : Leaving routine log_start_stop_msg
DM_CNTL :  Entering ANNOUNCE_SERVER
ANNOUNCE_SERVER : Entered routine ANNOUNCE_SERVER
ANNOUNCE_SERVER :  Entering ANNOUNCE
12:37:45.215 ANNOUNCE_SERVER : Return from ANNOUNCE: with rc of 0
and ec of 0
ANNOUNCE_SERVER : Leaving routine ANNOUNCE_SERVER
DM_CNTL : Return from ANNOUNCE_SERVER: with rc of 0 and ec of 0
12:37:45.215 PROCESS_MESSAGES : Entered routine PROCESS_MESSAGES
12:37:45.215 PROCESS_MESSAGES :  Entering RECEIVE_MESSAGE
12:37:45.215 RECEIVE_MESSAGE : Entered routine RECEIVE_MESSAGE
12:37:45.215 RECEIVE_MESSAGE :  edm_getmessages
12:37:45.215 EDM_GETMESSAGES : Entered routine edm_getmessages
12:37:45.215 EDM_GETMESSAGES :  edm_readmessages
12:37:45.215 EDM_READMESSAGES : Entered routine edm_readmessages
12:37:45.215 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
12:37:45.215 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
12:37:45.215 EDM_GETMESSAGES : rec->conlist[0] = -1
12:37:45.215 EDM_GETMESSAGES :  edm_readmessages
EDM_READMESSAGES : Entered routine edm_readmessages
12:37:45.215 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
```

```
12:38:00.217 CALL_LM_TIMER : Entered routine call_lm_timer
12:38:00.227 CALL_LM_TIMER :  license Y/N   = 1
12:38:00.227 CALL_LM_TIMER : timestamp   = 0
12:38:00.227 CALL_LM_TIMER : timenow  = 14959148
12:38:00.227 CALL_LM_TIMER :  calling lm_timer =
             CALL_LM_TIMER : Leaving routine call_lm_timer
12:38:15.228 CALL_LM_TIMER : Entered routine call_lm_timer
12:38:15.228 CALL_LM_TIMER :  license Y/N   = 1
12:38:15.228 CALL_LM_TIMER : timestamp   = 14959148
12:38:15.228 CALL_LM_TIMER : timenow  = 14959148
12:38:15.228 CALL_LM_TIMER : Leaving routine call_lm_timer
12:38:30.230 CALL_LM_TIMER : Entered routine call_lm_timer
12:38:30.230 CALL_LM_TIMER :  license Y/N   = 1
12:38:30.230 CALL_LM_TIMER : timestamp   = 14959148
12:38:30.230 CALL_LM_TIMER : timenow  = 14959148
12:38:30.230 CALL_LM_TIMER : Leaving routine call_lm_timer
12:38:45.241 CALL_LM_TIMER : Entered routine call_lm_timer
12:38:45.241 CALL_LM_TIMER :  license Y/N   = 1
12:38:45.241 CALL_LM_TIMER : timestamp   = 14959148
12:38:45.241 CALL_LM_TIMER : timenow  = 14959148
12:38:45.241 CALL_LM_TIMER : Leaving routine call_lm_timer
12:39:00.243 CALL_LM_TIMER : Entered routine call_lm_timer
12:39:00.243 CALL_LM_TIMER :  license Y/N   = 1
12:39:00.243 CALL_LM_TIMER : timestamp   = 14959148
12:39:00.243 CALL_LM_TIMER : timenow  = 14959149
12:39:00.243 CALL_LM_TIMER : Leaving routine call_lm_timer
12:39:15.254 CALL_LM_TIMER : Entered routine call_lm_timer
12:39:15.254 CALL_LM_TIMER :  license Y/N   = 1
12:39:15.254 CALL_LM_TIMER : timestamp   = 14959148
12:39:15.254 CALL_LM_TIMER : timenow  = 14959149
12:39:15.254 CALL_LM_TIMER : Leaving routine call_lm_timer
12:39:30.276 CALL_LM_TIMER : Entered routine call_lm_timer
12:39:30.276 CALL_LM_TIMER :  license Y/N   = 1
12:39:30.276 CALL_LM_TIMER : timestamp   = 14959148
12:39:30.276 CALL_LM_TIMER : timenow  = 14959149
12:39:30.276 CALL_LM_TIMER : Leaving routine call_lm_timer
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 0 and ec
of 300
EDM_READMESSAGES :  INITIATE_REQUEST_INDICATE
EDM_READMESSAGES : conid = 4
EDM_READMESSAGES :  Entering D_INIT_RESP
12:39:32.369 EDM_READMESSAGES : Return from D_INIT_RESP: with rc
of 0 and ec of 300
12:39:32.369 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 0 and ec
of 315
EDM_READMESSAGES :  MESSAGE_INDICATE
EDM_READMESSAGES : conid = 4
EDM_READMESSAGES : Leaving routine edm_readmessages
12:39:34.783 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
12:39:34.783 EDM_GETMESSAGES : rec->conlist[0] = -1
12:39:34.783 EDM_GETMESSAGES :  Any connection. Got a message
12:39:34.783 EDM_GETMESSAGES : conid = 4
```

```
EDM_GETMESSAGES : Leaving routine edm_getmessages
RECEIVE_MESSAGE: Return from edm_getmessages: with rc of 0 and ec
of 0
GET_NORM_MESSAGE_TYPE : Entered routine GET_NORM_MESSAGE_TYPE
GET_NORM_MESSAGE_TYPE : Leaving routine GET_NORM_MESSAGE_TYPE
RECEIVE_MESSAGE : received_message_type = 2
RECEIVE_MESSAGE : Leaving routine RECEIVE_MESSAGE
12:39:34.783 PROCESS_MESSAGES : Return from RECEIVE_MESSAGE: with
rc of 0 and ec of 0
12:39:34.783 PROCESS_MESSAGES :  Entering INVOKE_DM_FUNCTION
INVOKE_DM_FUNCTION : Entered routine INVOKE_DM_FUNCTION
INVOKE_DM_FUNCTION : About to enter:  = COPY
DM_COPY : Entered routine dm_copy
DM_COPY :  Entering COPY_FILE
COPY_FILE : Entered routine COPY_FILE
12:39:34.823 MAX_REVS_FROM_TABLES : Entered routine
max_revs_from_tables
MAX_REVS_FROM_TABLES :  About to get max rev from
DM_FILE_DIRECTORY
MAX_REVS_FROM_TABLES : Return from SQL: Table DM_FILE_DIRECTORY;
Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Max file rev
MAX_REVS_FROM_TABLES :  About to get max rev from DM_ARCHIVE
MAX_REVS_FROM_TABLES : Return from SQL: Table DM_FILE_DIRECTORY;
Command SELECT
SQLCODE = 100; #Rows: 0 Comment: Max file rev
MAX_REVS_FROM_TABLES : Leaving routine max_revs_from_tables
COPY_FILE :  select a  row
COPY_FILE : Return from SQL: Table Select; Command
DM_FILE_DIRECTORY
 SQLCODE = 0; #Rows: 1 Comment: dm_file_name
COPY_FILE :  Entering PROCESS_FILE
PROCESS_FILE : Entered routine PROCESS_FILE
PROCESS_FILE :  entering  CHECK_AUTHORITY
CHECK_AUTHORITY : Entered routine CHECK_AUTHORITY
CHECK_AUTHORITY :  entering  ASMGR
ASMGR : Entered routine asmgr
12:39:34.953 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
12:39:35.063 GET_SQL_TEXT : Entered routine get_sql_text
12:39:35.063 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:35.063 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:35.063  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:35.063 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
```

```
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.123 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:35.123 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_COMMAND_LIST,DM_USER_AUTH WHERE DM_AUTH_USER_ID = 'EDMADMIN'
AND DM_AUTH_PROJ_ID = ' ' AND DM_AUTH_CMD_LIST = DM_COMMAND_LIST
AND DM_COMMAND_NAME IN ('COPY','ALL')
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.193 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
GET_SQL_TEXT : Entered routine get_sql_text
12:39:35.213 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:35.213 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:35.213  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:35.213 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
```

```
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.233 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:35.233 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM DM_USER_AUTH
WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND DM_AUTH_PROJ_ID = ' ' AND
DM_AUTH_RD_AUTHG = ' '
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.273 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
GET_SQL_TEXT : Entered routine get_sql_text
12:39:35.273 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:35.273 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:35.273  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:35.273 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.313 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:35.313 ASMGR :  Entering sql_placeholder
```

```
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH,DM_STATUS_CODE WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_STAT_PROJ_ID = ' ' AND
DM_STAT_STATUS_CD = 'IW' AND DM_AUTH_RD_AUTH >= DM_STAT_AUTH
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.393 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
GET_SQL_TEXT : Entered routine get_sql_text
12:39:35.393 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:35.393 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:35.393  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:35.393 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.434 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:35.434 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM DM_USER_AUTH
WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND DM_AUTH_PROJ_ID = ' ' AND
DM_AUTH_WR_AUTHG = ' '
```

```
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.474 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
GET_SQL_TEXT : Entered routine get_sql_text
12:39:35.474 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:35.474 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:35.474  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:35.474 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.514 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:35.514 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH,DM_STATUS_CODE WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_STAT_PROJ_ID = ' ' AND
DM_STAT_STATUS_CD = 'IW' AND DM_AUTH_WR_AUTH >= DM_STAT_AUTH
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
```

```
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.554 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
GET_SQL_TEXT : Entered routine get_sql_text
12:39:35.554 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:35.554 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:35.554  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:35.554 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.594 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:35.594 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM DM_USER_AUTH
WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND DM_AUTH_PROJ_ID = ' ' AND
DM_AUTH_ADMIN_FLAG = 'Y'
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
```

```
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
ASMGR : Leaving routine asmgr
CHECK_AUTHORITY : Return from ASMGR: with rc of -1 and ec of 0
CHECK_AUTHORITY : Leaving routine CHECK_AUTHORITY
PROCESS_FILE: Return from CHECK_AUTHORITY: with rc of 0 and ec of 0
PROCESS_FILE :  entering  CHECK_AVAILABILITY
CHECK_AVAILABILITY : Entered routine CHECK_AVAILABILITY
CHECK_AVAILABILITY : Leaving routine CHECK_AVAILABILITY
PROCESS_FILE: Return from CHECK_AVAILABILITY: with rc of 0 and ec
of 0
PROCESS_FILE :  entering  MAKE_NEW_NAME
MAKE_NEW_NAME : Entered routine MAKE_NEW_NAME
MAKE_NEW_NAME : Leaving routine MAKE_NEW_NAME
PROCESS_FILE : Return from MAKE_NEW_NAME: with rc of 0 and ec of 0
PROCESS_FILE :  entering  GET_OLD_POOLINFO
GET_OLD_POOL_INFO : Entered routine GET_OLD_POOL_INFO
12:39:35.614 GET_OLD_POOL_INFO :  select for DM_POOL_INFO
GET_OLD_POOL_INFO : Return from SQL: Table SELECT; Command
DM_POOL_INFO
SQLCODE = 0; #Rows: 1 Comment: select text
GET_OLD_POOL_INFO : Leaving routine GET_OLD_POOL_INFO
PROCESS_FILE:Return from GET_OLD_POOLINFO: with rc of 0 and ec of 0
PROCESS_FILE :  entering  SET_ATTRIBS
12:39:35.634 SET_ATTRIBS : Entered routine SET_ATTRIBS
SET_ATTRIBS :  entering DM_STORE
DM_STORE : Entered routine dm_store
DM_STORE :  select count of -1 versions DM_FILE_DIRECTORY
DM_STORE : Return from SQL: Table SELECT; Command
DM_FILE_DIRECTORY
SQLCODE = 0; #Rows: 1 Comment: select count in DM_FILE_DIRECTORY
DM_STORE :  Entering internal select_directory
SELECT_DIRECTORY : Entered routine select_directory
12:39:35.674 SELECT_DIRECTORY :  About to select from
DM_FILE_DIRECTORY
SELECT_DIRECTORY : Return from SQL: Table DM_FILE_DIRECTORY;
Command SELECT
SQLCODE = 100; #Rows: 0 Comment: select row
SELECT_DIRECTORY : Leaving routine select_directory
DM_STORE : Return from select_directory: with rc of 97 and ec of 0
DM_STORE :  Entering GET_NUM_REV
GET_NUM_REV : Entered routine get_num_rev
GET_NUM_REV :  About to read project table
GET_NUM_REV : Return from SQL: Table DM_PROJECT; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: revision sequence name
GET_NUM_REV :  About to read revision code table
```

```
GET_NUM_REV : Return from SQL: Table DM_REVISION_CODE; Command
SELECT
SQLCODE = 0; #Rows: 1 Comment: num for given char
12:39:35.794 DM_STORE : Return from get_num_rev: with rc of 0 and
ec of 0
DM_STORE :  select count in DM_FILE_DIRECTORY
DM_STORE : Return from SQL: Table SELECT; Command
DM_FILE_DIRECTORY
SQLCODE = 0; #Rows: 1 Comment: select count in DM_FILE_DIRECTORY
DM_STORE :  select count in DM_ARCHIVE
DM_STORE : Return from SQL: Table SELECT; Command DM_ARCHIVE
SQLCODE = 0; #Rows: 1 Comment: select count in DM_ARCHIVE
SELECT_PREVIOUS_INFO : Entered routine select_previous_info
SELECT_PREVIOUS_INFO :  Entering max_revs_from_tables
12:39:35.854 MAX_REVS_FROM_TABLES : Entered routine
max_revs_from_tables
MAX_REVS_FROM_TABLES :  About to get max rev from
DM_FILE_DIRECTORY
MAX_REVS_FROM_TABLES : Return from SQL: Table DM_FILE_DIRECTORY;
Command SELECT
SQLCODE = 100; #Rows: 0 Comment: Max file rev
MAX_REVS_FROM_TABLES :  About to get max rev from DM_ARCHIVE
MAX_REVS_FROM_TABLES : Return from SQL: Table DM_FILE_DIRECTORY;
Command SELECT
SQLCODE = 100; #Rows: 0 Comment: Max file rev
MAX_REVS_FROM_TABLES : Leaving routine max_revs_from_tables
SELECT_PREVIOUS_INFO : Return from max_revs_from_tables: with rc
of 97 and ec of 100
SELECT_PREVIOUS_INFO : Leaving routine select_previous_info
DM_STORE :  Entering internal chk_store_file_auth
CHK_STORE_FILE_AUTH : Entered routine chk_store_file_auth
CHK_STORE_FILE_AUTH :  Entering asmgr
12:39:35.874 ASMGR : Entered routine asmgr
12:39:35.874 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
12:39:35.874 GET_SQL_TEXT : Entered routine get_sql_text
12:39:35.874 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:35.874 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:35.874  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:35.874 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
```

```
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.914 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:35.914 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_COMMAND_LIST,DM_USER_AUTH WHERE DM_AUTH_USER_ID = 'EDMADMIN'
AND DM_AUTH_PROJ_ID = ' ' AND DM_AUTH_CMD_LIST = DM_COMMAND_LIST
AND DM_COMMAND_NAME IN ('STORE','ALL')
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:35.964 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
GET_SQL_TEXT : Entered routine get_sql_text
12:39:35.984 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:35.984 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:35.984  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:35.984 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:36.004 GET_SQL_TEXT : Leaving routine get_sql_text
```

```
12:39:36.004 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_STATUS_CODE WHERE DM_STAT_PROJ_ID = ' ' AND DM_STAT_STATUS_CD =
'IW'
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:36.044 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
GET_SQL_TEXT : Entered routine get_sql_text
12:39:36.044 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:36.044 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:36.044  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:36.044 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:36.084 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:36.084 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM DM_USER_AUTH
WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND DM_AUTH_PROJ_ID = ' ' AND
DM_AUTH_WR_AUTHG = ' '
```

```
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:36.124 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
GET_SQL_TEXT : Entered routine get_sql_text
12:39:36.124 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:36.124 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:36.124  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:36.124 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:36.165 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:36.165 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_USER_AUTH,DM_STATUS_CODE WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND
DM_AUTH_PROJ_ID = ' ' AND DM_STAT_PROJ_ID = ' ' AND
DM_STAT_STATUS_CD = 'IW' AND DM_AUTH_WR_AUTH >= DM_STAT_AUTH
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
```

```
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR: Return from SQL: Table AS_COUNT_CURSOR; Command OPEN CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:36.185 ASMGR :  About to select from AS_COMMAND_TEST
ASMGR : Return from SQL: Table AS_COMMAND_TEST; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: Get next test for the command
GET_SQL_TEXT : Entered routine get_sql_text
12:39:36.205 GET_SQL_TEXT :  About to declare AS_TEST_CURSOR
12:39:36.205 GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR;
Command DCL CURSOR
12:39:36.205  SQLCODE = 0; #Rows: 1 Comment: For table AS_TEST
12:39:36.205 GET_SQL_TEXT :  About to open AS_TEST_CURSOR
GET_SQL_TEXT: Return from SQL: Table AS_TEST_CURSOR; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_SQL_TEXT :  About to do first fetch from AS_TEST_CURSOR
GET_SQL_TEXT: Return from SQL: Table AS_TEST_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: Fetch first row
GET_SQL_TEXT :  About to do next fetch from AS_TEST_CURSOR
GET_SQL_TEXT: Return from SQL: Table AS_TEST_CURSOR; Command FETCH
SQLCODE = 100; #Rows: 1 Comment: Fetch next row
GET_SQL_TEXT :  About to close AS_TEST_CURSOR
GET_SQL_TEXT : Return from SQL: Table AS_TEST_CURSOR; Command
CLOSE CURSOR
SQLCODE = 0; #Rows: 1 Comment: Close cursor
12:39:36.245 GET_SQL_TEXT : Leaving routine get_sql_text
12:39:36.245 ASMGR :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM DM_USER_AUTH
WHERE DM_AUTH_USER_ID = 'EDMADMIN' AND DM_AUTH_PROJ_ID = ' ' AND
DM_AUTH_ADMIN_FLAG = 'Y'
SQL_PLACEHOLDER : Leaving routine sql_placeholder
ASMGR :  Out of sql_placeholder
ASMGR :  About to prepare stmt ROW_CNT
ASMGR : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing test
ASMGR :  About to declare cursor AS_COUNT_CURSOR for ROW_CNT
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command DCL CURSOR
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
ASMGR :  About to open AS_COUNT_CURSOR
ASMGR: Return from SQL: Table AS_COUNT_CURSOR; Command OPEN CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening  cursor
ASMGR :  About to do fetch AS_COUNT_CURSOR into rowcnt for test
```

```
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command FETCH
SQLCODE = 0; #Rows: 1 Comment: First fetch
ASMGR : Rowcount  = 1
ASMGR :  About to close AS_COUNT_CURSOR
ASMGR : Return from SQL: Table AS_COUNT_CURSOR; Command CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
ASMGR : Leaving routine asmgr
CHK_STORE_FILE_AUTH : Return from asmgr: with rc of -1 and ec of 0
CHK_STORE_FILE_AUTH : Leaving routine chk_store_file_auth
12:39:36.265 DM_STORE : Return from chk_store_file_auth: with rc
of 0 and ec of 100
DM_STORE :  Entering makerowid
MAKEROWID : Entered routine makerowid
MAKEROWID : rowid = FPFMDVTCNH
MAKEROWID : Leaving routine makerowid
DM_STORE : Return from makerowid: with rc of 0 and ec of 0
DM_STORE :  About to insert into DM_FILE_DIRECTORY
DM_STORE : Return from SQL: Table DM_FILE_DIRECTORY; Command
INSERT
SQLCODE = 0; #Rows: 1 Comment: store file
DM_STORE :  entering insert_rev_hist
INSERT_REV_HIST : Entered routine INSERT_REV_HIST
12:39:36.295 CHECK_IF_RELEASED : Entered routine check_if_released
CHECK_IF_RELEASED :  Select the seqno for the released status
CHECK_IF_RELEASED : Return from SQL: Table DM_STATUS_CODE; Command
SELECT
SQLCODE = 0; #Rows: 1 Comment: Select based on initial_cd and
proj_id
CHECK_IF_RELEASED :  Select count based on proj_id,stat_cd,and
seqno
CHECK_IF_RELEASED : Return from SQL: Table DM_STATUS_CODE; Command
SELECT COUNT
SQLCODE = 0; #Rows: 1 Comment: Select count to see if stat_cd is
released
CHECK_IF_RELEASED : Leaving routine check_if_released
INSERT_REV_HIST : Leaving routine INSERT_REV_HIST
12:39:36.355 DM_STORE : Return from insert_rev_hist: with rc of 0
and ec of 0
FMT_MSG_STORE : Entered routine fmt_msg_store
FMT_MSG_STORE : message  = CDMCPY172I The file has been stored.
FMT_MSG_STORE : Leaving routine fmt_msg_store
12:39:36.355 DM_STORE : Leaving routine dm_store
SET_ATTRIBS : Return from DM_STORE: with rc of 0 and ec of 0
SET_ATTRIBS : Leaving routine SET_ATTRIBS
PROCESS_FILE : Return from SET_ATTRIBS: with rc of 0 and ec of 0
PROCESS_FILE :  Entering format_and_send_message
12:39:36.395 FORMAT_AND_SEND_MESSAGE : Entered routine
FORMAT_AND_SEND_MESSAGE
FORMAT_AND_SEND_MESSAGE :  Entering SEND_MESSAGE
SEND_MESSAGE : Entered routine SEND_MESSAGE
GET_MESSAGE_TYPE : Entered routine get_message_type
GET_MESSAGE_TYPE : Leaving routine get_message_type
12:39:36.395 SEND_MESSAGE : send_message_type = 2
```

```
12:39:36.395 SEND_MESSAGE : Return from get_message_type: with rc
of 0 and ec of 0
12:39:36.395 SEND_MESSAGE :  Entering D_MESSAGE
12:39:36.395 SEND_MESSAGE : Return from D_MESSAGE: with rc of 0
and ec of 0
12:39:36.405 EDM_READMESSAGES : Entered routine edm_readmessages
12:39:36.405 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
12:39:36.405 SEND_MESSAGE : Leaving routine SEND_MESSAGE
12:39:36.405 FORMAT_AND_SEND_MESSAGE : Return from SEND_MESSAGE:
with rc of 0 and ec of 0
12:39:36.405 FORMAT_AND_SEND_MESSAGE :  send message to log
12:39:36.405 FORMAT_AND_SEND_MESSAGE : Leaving routine
FORMAT_AND_SEND_MESSAGE
PROCESS_FILE : Return from FORMAT_AND_SEND_MESSAGE: with rc of 0
and ec of 0
PROCESS_FILE : Leaving routine PROCESS_FILE
COPY_FILE : Return from PROCESS_FILE: with rc of 0 and ec of 0
COPY_FILE :  Entering UPDATE_SOURCE_FILE
12:39:36.405 UPDATE_SOURCE_FILE : Entered routine
update_source_file
12:39:36.405 UPDATE_SOURCE_FILE :  Updating DM_FILE_DIRECTORY
date/time
UPDATE_SOURCE_FILE : Return from SQL: Table DM_FILE_DIRECTORY;
Command UPDATE
SQLCODE = 0; #Rows: 1 Comment: date/time
UPDATE_SOURCE_FILE : Leaving routine update_source_file
COPY_FILE:Return from UPDATE_SOURCE_FILE: with rc of 0 and ec of 0
COPY_FILE : Leaving routine COPY_FILE
DM_COPY : Return from COPY_FILE: with rc of 0 and ec of 0
DM_COPY :  Entering commit_only
SQL_COMMIT_ONLY : Entered routine sql_commit_only
SQL_COMMIT_ONLY :  About to SQL COMMIT WORK
SQL_COMMIT_ONLY : Return from SQL: Table  ; Command COMMIT
 SQLCODE = 0; #Rows: 1 Comment: After commit
SQL_COMMIT_ONLY : Leaving routine sql_commit_only
DM_COPY : Return from commit_only: with rc of 0 and ec of 0
DM_COPY :  Entering SEND_MESSAGE
12:39:36.916 SEND_MESSAGE : Entered routine SEND_MESSAGE
12:39:36.916 GET_MESSAGE_TYPE : Entered routine get_message_type
12:39:36.916 GET_MESSAGE_TYPE : Leaving routine get_message_type
12:39:36.916 SEND_MESSAGE : send_message_type = 2
12:39:36.916 SEND_MESSAGE : Return from get_message_type: with rc
of 0 and ec of 0
12:39:36.916 SEND_MESSAGE :  Entering D_MESSAGE
12:39:36.936 SEND_MESSAGE : Return from D_MESSAGE: with rc of 0
and ec of 0
EDM_READMESSAGES : Entered routine edm_readmessages
12:39:36.936 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
```

```
12:39:36.936 SEND_MESSAGE : Leaving routine SEND_MESSAGE
12:39:36.936 DM_COPY : Return from SEND_MESSAGE: with rc of 0 and
ec of 0
12:39:36.936 DM_COPY : Leaving routine dm_copy
INVOKE_DM_FUNCTION : Return from COPY    : with rc of 0 and ec of 0
INVOKE_DM_FUNCTION :  Entering SQL_COMMIT_ONLY
SQL_COMMIT_ONLY : Entered routine sql_commit_only
SQL_COMMIT_ONLY :  About to SQL COMMIT WORK
SQL_COMMIT_ONLY : Return from SQL: Table  ; Command COMMIT
SQLCODE = 0; #Rows: 1 Comment: After commit
SQL_COMMIT_ONLY : Leaving routine sql_commit_only
INVOKE_DM_FUNCTION : Return from SQL_COMMIT_ONLY: with rc of 0 and
ec of 0
INVOKE_DM_FUNCTION : Leaving routine INVOKE_DM_FUNCTION
12:39:36.966 PROCESS_MESSAGES : Return from INVOKE_DM_FUNCTION:
with rc of 0 and ec of 0
12:39:36.966 PROCESS_MESSAGES : Leaving routine PROCESS_MESSAGES
12:39:36.966 PROCESS_MESSAGES : Entered routine PROCESS_MESSAGES
12:39:36.966 PROCESS_MESSAGES :  Entering RECEIVE_MESSAGE
12:39:36.966 RECEIVE_MESSAGE : Entered routine RECEIVE_MESSAGE
12:39:36.966 RECEIVE_MESSAGE :  edm_getmessages
12:39:36.966 EDM_GETMESSAGES : Entered routine edm_getmessages
12:39:36.966 EDM_GETMESSAGES :  edm_readmessages
12:39:36.966 EDM_READMESSAGES : Entered routine edm_readmessages
12:39:36.966 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
12:39:36.966 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
12:39:36.966 EDM_GETMESSAGES : rec->conlist[0] = -1
12:39:36.966 EDM_GETMESSAGES :  edm_readmessages
EDM_READMESSAGES : Entered routine edm_readmessages
12:39:36.966 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 0 and ec
of 315
EDM_READMESSAGES :  MESSAGE_INDICATE
EDM_READMESSAGES : conid = 4
EDM_READMESSAGES : Leaving routine edm_readmessages
12:39:36.976 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
12:39:36.976 EDM_GETMESSAGES : rec->conlist[0] = -1
12:39:36.976 EDM_GETMESSAGES :  Any connection. Got a message
12:39:36.976 EDM_GETMESSAGES : conid = 4
EDM_GETMESSAGES : Leaving routine edm_getmessages
RECEIVE_MESSAGE : Return from edm_getmessages: with rc of 0 and ec
of 0
GET_NORM_MESSAGE_TYPE : Entered routine GET_NORM_MESSAGE_TYPE
GET_NORM_MESSAGE_TYPE : Leaving routine GET_NORM_MESSAGE_TYPE
RECEIVE_MESSAGE : received_message_type = 2
RECEIVE_MESSAGE : Leaving routine RECEIVE_MESSAGE
12:39:36.976 PROCESS_MESSAGES : Return from RECEIVE_MESSAGE: with
rc of 0 and ec of 0
12:39:36.976 PROCESS_MESSAGES :  Entering INVOKE_DM_FUNCTION
```

```
INVOKE_DM_FUNCTION : Entered routine INVOKE_DM_FUNCTION
INVOKE_DM_FUNCTION : About to enter:  = STORE
DM_STORE : Entered routine dm_store
DM_STORE :  Entering internal select_directory
SELECT_DIRECTORY : Entered routine select_directory
12:39:36.976 SELECT_DIRECTORY :  About to select from
DM_FILE_DIRECTORY
SELECT_DIRECTORY : Return from SQL: Table DM_FILE_DIRECTORY;
Command SELECT
SQLCODE = 0; #Rows: 1 Comment: select row
SELECT_DIRECTORY :  About to select from DM_PART_FILE
SELECT_DIRECTORY : Return from SQL: Table DM_PART_FILE; Command
SELECT
SQLCODE = 100; #Rows: 0 Comment: part_rowid
SELECT_DIRECTORY : Leaving routine select_directory
DM_STORE : Return from select_directory: with rc of 0 and ec of 0
FINDPOOL : Entered routine findpool
FINDPOOL :  Entering find_spool
FIND_SPOOL : Entered routine find_spool
12:39:37.496 FIND_SPOOL :  Entering get_pool_query
GET_POOL_QUERY : Entered routine get_pool_query
12:39:37.496 GET_POOL_QUERY : file rowid = FPFMDVTCNH
12:39:37.496 GET_POOL_QUERY :  SELECT 1st sequence number from
DM_POOL_QUEST
GET_POOL_QUERY : Return from SQL: Table DM_POOL_QUEST; Command
SELECT
SQLCODE = 0; #Rows: 1 Comment: Get 1st seqno
GET_POOL_QUERY :  Retrieving DM_POOL_QUEST row
GET_POOL_QUERY : Return from SQL: Table DM_POOL_QUEST; Command
SELECT
SQLCODE = 0; #Rows: 1 Comment: Get quest step info
GET_POOL_QUERY :  Entering fetch_query_text
FETCH_QUERY_TEXT : Entered routine fetch_query_text
12:39:37.547 FETCH_QUERY_TEXT :  Declaring cursor SRCH_QUERY_TEXT
12:39:37.547 FETCH_QUERY_TEXT : Return from SQL: Table
DM_SRCH_QUERY; Command dcl cursor
12:39:37.547  SQLCODE = 0; #Rows: 1 Comment: SRCH_QUERY_TEXT
12:39:37.547 FETCH_QUERY_TEXT :  Opening cursor SRCH_QUERY_TEXT
FETCH_QUERY_TEXT : Return from SQL: Table DM_SRCH_QUERY; Command
open cursor
SQLCODE = 0; #Rows: 0 Comment: SRCH_QUERY_TEXT
FETCH_QUERY_TEXT :  Fetching query text
FETCH_QUERY_TEXT : Return from SQL: Table DM_SRCH_QUERY; Command
fetch cursor
SQLCODE = 0; #Rows: 1 Comment: SRCH_QUERY_TEXT
FETCH_QUERY_TEXT :  Fetching query text
FETCH_QUERY_TEXT : Return from SQL: Table DM_SRCH_QUERY; Command
fetch cursor
SQLCODE = 100; #Rows: 1 Comment: SRCH_QUERY_TEXT
FETCH_QUERY_TEXT :  Closing cursor SRCH_QUERY_TEXT
FETCH_QUERY_TEXT : Return from SQL: Table DM_SRCH_QUERY; Command
close cursor
SQLCODE = 0; #Rows: 1 Comment: SRCH_QUERY_TEXT
FETCH_QUERY_TEXT : Leaving routine fetch_query_text
```

```
GET_POOL_QUERY : Return from fetch_query_text: with rc of 0 and ec
of 0
GET_POOL_QUERY :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = SELECT COUNT(*) FROM
DM_STORAGE_POOL WHERE DM_POOL_PCT_USED < 50 AND DM_POOL_STATUS =
'0' AND DM_POOL_NAME <> (SELECT DM_FILE_POOL_NAME FROM
DM_FILE_DIRECTORY WHERE DM_FILE_ROWID = 'FPFMDVTCNH')
SQL_PLACEHOLDER : Leaving routine sql_placeholder
GET_POOL_QUERY : Return from sql_placeholder: with rc of 0 and ec
of 0
GET_POOL_QUERY :  Entering get_rowcount
12:39:37.607 GET_ROWCOUNT : Entered routine get_rowcount
12:39:37.607 GET_ROWCOUNT :  About to prepare stmt ROW_CNT
GET_ROWCOUNT : Return from SQL: Table ROW_CNT; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing SELECT COUNT(*) query
GET_ROWCOUNT :  Declaring cursor POOL_SRCH_CURSOR for ROW_CNT
GET_ROWCOUNT : Return from SQL: Table POOL_SRCH_CURSOR; Command
dcl cursor
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt ROW_CNT
GET_ROWCOUNT :  Opening cursor POOL_SRCH_CURSOR
GET_ROWCOUNT : Return from SQL: Table POOL_SRCH_CURSOR; Command
OPEN CURSOR
SQLCODE = 0; #Rows: 0 Comment: Opening cursor
GET_ROWCOUNT :  Fetching POOL_SRCH_CURSOR into rowcnt for query
GET_ROWCOUNT : Return from SQL: Table POOL_SRCH_CURSOR; Command
FETCH
SQLCODE = 0; #Rows: 1 Comment: First and only fetch
GET_ROWCOUNT : Rowcount  = 2
GET_ROWCOUNT :  Closing cursor POOL_SRCH_CURSOR
GET_ROWCOUNT : Return from SQL: Table POOL_SRCH_CURSOR; Command
CLOSE
SQLCODE = 0; #Rows: 1 Comment: Close cursor
GET_ROWCOUNT : Leaving routine get_rowcount
GET_POOL_QUERY : Return from get_rowcount: with rc of 0 and ec of 0
GET_POOL_QUERY : Leaving routine get_pool_query
12:39:37.737 FIND_SPOOL : Return from get_pool_query: with rc of 0
and ec of 0
FIND_SPOOL :  Entering fetch_where_text
FETCH_WHERE_TEXT : Entered routine fetch_where_text
12:39:37.737 FETCH_WHERE_TEXT :  Declaring cursor POOL_QUERY_TEXT
12:39:37.737 FETCH_WHERE_TEXT : Return from SQL: Table
DM_POOL_QUERY; Command dcl cursor
12:39:37.737  SQLCODE = 0; #Rows: 1 Comment: POOL_QUERY_TEXT
12:39:37.737 FETCH_WHERE_TEXT :  Opening cursor POOL_QUERY_TEXT
FETCH_WHERE_TEXT : Return from SQL: Table DM_POOL_QUERY; Command
open cursor
SQLCODE = 0; #Rows: 0 Comment: POOL_QUERY_TEXT
FETCH_WHERE_TEXT :  Fetching query text
FETCH_WHERE_TEXT : Return from SQL: Table DM_POOL_QUERY; Command
fetch cursor
SQLCODE = 0; #Rows: 1 Comment: POOL_QUERY_TEXT
FETCH_WHERE_TEXT :  Fetching query text
```

```
FETCH_WHERE_TEXT : Return from SQL: Table DM_POOL_QUERY; Command
fetch cursor
SQLCODE = 100; #Rows: 1 Comment: POOL_QUERY_TEXT
FETCH_WHERE_TEXT :  Closing cursor POOL_QUERY_TEXT
FETCH_WHERE_TEXT : Return from SQL: Table DM_POOL_QUERY; Command
close cursor
SQLCODE = 0; #Rows: 1 Comment: POOL_QUERY_TEXT
FETCH_WHERE_TEXT : Leaving routine fetch_where_text
FIND_SPOOL: Return from fetch_where_text: with rc of 0 and ec of 0
FIND_SPOOL :  Entering sql_placeholder
SQL_PLACEHOLDER : Entered routine sql_placeholder
SQL_PLACEHOLDER : parsed text = WHERE DM_POOL_PCT_USED < 50 AND
DM_POOL_NAME NOT IN (SELECT DM_FILE_POOL_NAME FROM
DM_FILE_DIRECTORY WHERE DM_FILE_ROWID = 'FPFMDVTCNH') ORDER BY
DM_POOL_PCT_USED
SQL_PLACEHOLDER : Leaving routine sql_placeholder
FIND_SPOOL : Return from sql_placeholder: with rc of 0 and ec of 0
FIND_SPOOL : pool select query = SELECT
DM_POOL_NAME,DM_POOL_NO_FILES,DM_POOL_SIZE,DM_POOL_SIZE_USED,DM_P
OOL_SIZE_FREE,DM_POOL_PCT_USED,DM_POOL_PCT_FREE,DM_POOL_NEXT_FILE
,DM_POOL_STATUS,DM_POOL_USER_ID,DM_POOL_DATE_WR,DM_POOL_TIME_WR
FROM DM_STORAGE_POOL WHERE DM_POOL_PCT_USED < 50 AND DM_POOL_NAME
NOT IN (SELECT DM_FILE_POOL_NAME FROM DM_FILE_DIRECTORY WHERE
DM_FILE_ROWID = 'FPFMDVTCNH') ORDER BY DM_POOL_PCT_USED
FIND_SPOOL :  About to prepare stmt POOL_QUERY
FIND_SPOOL : Return from SQL: Table POOL_QUERY; Command PREPARE
SQLCODE = 0; #Rows: 1 Comment: Preparing pool selection query
FIND_SPOOL :  Declaring cursor FIND_POOL_CURSOR for POOL_QUERY
FIND_SPOOL : Return from SQL: Table FIND_POOL_CURSOR; Command dcl
cursor
SQLCODE = 0; #Rows: 1 Comment: Cursor for stmt POOL_QUERY
FIND_SPOOL :  Opening cursor FIND_POOL_CURSOR
FIND_SPOOL : Return from SQL: Table DM_STORAGE_POOL; Command OPEN
CURSOR
SQLCODE = 0; #Rows: 0 Comment: FIND_POOL_CURSOR
FIND_SPOOL :  Fetching pool row
FIND_SPOOL : Return from SQL: Table DM_STORAGE_POOL; Command fetch
cursor
SQLCODE = 0; #Rows: 1 Comment: FIND_POOL_CURSOR
FIND_SPOOL :  Closing cursor FIND_POOL_CURSOR
FIND_SPOOL : Return from SQL: Table DM_STORAGE_POOL; Command CLOSE
CURSOR
SQLCODE = 0; #Rows: 1 Comment: FIND_POOL_CURSOR
FIND_SPOOL :  Retrieving pool info
FIND_SPOOL : Return from SQL: Table DM_POOL_INFO; Command SELECT
SQLCODE = 0; #Rows: 1 Comment: pool info
FIND_SPOOL :  About to update DM_STORAGE_POOL
FIND_SPOOL: Return from SQL: Table DM_STORAGE_POOL; Command UPDATE
SQLCODE = 0; #Rows: 1 Comment: Update storage pool
FIND_SPOOL : Leaving routine find_spool
12:39:37.887 FINDPOOL : Return from find_spool: with rc of 0 and
ec of 0
12:39:37.887 FINDPOOL :  Entering make_sp_filename
FINDPOOL : Return from make_sp_filename: with rc of 0 and ec of 0
```

```
FINDPOOL : Leaving routine findpool
DM_STORE :  About to update DM_FILE_DIRECTORY
DM_STORE: Return from SQL: Table DM_FILE_DIRECTORY; Command UPDATE
SQLCODE = 0; #Rows: 1 Comment: -1 version
FMT_MSG_STORE : Entered routine fmt_msg_store
FMT_MSG_STORE :  Entering send_message
SEND_MESSAGE : Entered routine SEND_MESSAGE
GET_MESSAGE_TYPE : Entered routine get_message_type
GET_MESSAGE_TYPE : Leaving routine get_message_type
12:39:37.907 SEND_MESSAGE : send_message_type = 2
12:39:37.907 SEND_MESSAGE : Return from get_message_type: with rc
of 0 and ec of 0
12:39:37.907 SEND_MESSAGE :  Entering D_MESSAGE
12:39:37.917 SEND_MESSAGE : Return from D_MESSAGE: with rc of 0 and
ec of 0
EDM_READMESSAGES : Entered routine edm_readmessages
12:39:37.917 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
12:39:37.917 SEND_MESSAGE : Leaving routine SEND_MESSAGE
12:39:37.917 FMT_MSG_STORE : Return from send_message: with rc of
0 and ec of 0
12:39:37.917 FMT_MSG_STORE : message  = CDMCPY172I The file has
been stored.
12:39:37.917 FMT_MSG_STORE : Leaving routine fmt_msg_store
12:39:37.917 DM_STORE : Leaving routine dm_store
12:39:37.917 INVOKE_DM_FUNCTION : Return from STORE   : with rc of
0 and ec of 0
12:39:37.917 INVOKE_DM_FUNCTION :  Entering SQL_COMMIT_ONLY
SQL_COMMIT_ONLY : Entered routine sql_commit_only
SQL_COMMIT_ONLY :  About to SQL COMMIT WORK
SQL_COMMIT_ONLY : Return from SQL: Table  ; Command COMMIT
SQLCODE = 0; #Rows: 1 Comment: After commit
SQL_COMMIT_ONLY : Leaving routine sql_commit_only
12:39:38.258 INVOKE_DM_FUNCTION : Return from SQL_COMMIT_ONLY:
with rc of 0 and ec of 0
12:39:38.258 INVOKE_DM_FUNCTION : Leaving routine
INVOKE_DM_FUNCTION
12:39:38.258 PROCESS_MESSAGES : Return from INVOKE_DM_FUNCTION:
with rc of 0 and ec of 0
12:39:38.258 PROCESS_MESSAGES : Leaving routine PROCESS_MESSAGES
12:39:38.258 PROCESS_MESSAGES : Entered routine PROCESS_MESSAGES
12:39:38.258 PROCESS_MESSAGES :  Entering RECEIVE_MESSAGE
RECEIVE_MESSAGE : Entered routine RECEIVE_MESSAGE
RECEIVE_MESSAGE :  edm_getmessages
EDM_GETMESSAGES : Entered routine edm_getmessages
EDM_GETMESSAGES :  edm_readmessages
12:39:38.258 EDM_READMESSAGES : Entered routine edm_readmessages
12:39:38.258 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
```

```
EDM_GETMESSAGES : Return from edm_readmessages: with rc of 0 and
ec of 0
EDM_GETMESSAGES : rec->conlist[0] = -1
EDM_GETMESSAGES :  edm_readmessages
EDM_READMESSAGES : Entered routine edm_readmessages
12:39:38.268 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 0 and ec
of 315
EDM_READMESSAGES :  MESSAGE_INDICATE
EDM_READMESSAGES : conid = 4
EDM_READMESSAGES : Leaving routine edm_readmessages
EDM_GETMESSAGES : Return from edm_readmessages: with rc of 0 and
ec of 0
EDM_GETMESSAGES : rec->conlist[0] = -1
EDM_GETMESSAGES :  Any connection. Got a message
EDM_GETMESSAGES : conid = 4
EDM_GETMESSAGES : Leaving routine edm_getmessages
RECEIVE_MESSAGE: Return from edm_getmessages: with rc of 0 and ec
of 0
GET_NORM_MESSAGE_TYPE : Entered routine GET_NORM_MESSAGE_TYPE
GET_NORM_MESSAGE_TYPE : Leaving routine GET_NORM_MESSAGE_TYPE
RECEIVE_MESSAGE : received_message_type = 2
RECEIVE_MESSAGE : Leaving routine RECEIVE_MESSAGE
12:39:38.608 PROCESS_MESSAGES : Return from RECEIVE_MESSAGE: with
rc of 0 and ec of 0
12:39:38.608 PROCESS_MESSAGES :  Entering INVOKE_DM_FUNCTION
12:39:38.608 INVOKE_DM_FUNCTION : Entered routine
INVOKE_DM_FUNCTION
INVOKE_DM_FUNCTION : About to enter:  = REVOKE
DM_REVOKE : Entered routine dm_revoke
DM_REVOKE : Command to revoke:  = STORE
DM_REVOKE : Selection scope:    = F
DM_REVOKE : This item type:     = F
DM_REVOKE : This file name:     = vlt_n1
DM_REVOKE : EOT switch (ON=Y):  = N
DM_REVOKE : Revoke switch (GOOD=Y):  = Y
FINISH_STORE : Entered routine finish_store
FINISH_STORE :  about to select from DM_FILE_DIRECTORY
FINISH_STORE : Return from SQL: Table DM_FILE_DIRECTORY; Command
SELECT
SQLCODE = 100; #Rows: 0 Comment: select from dm_file_directory
FINISH_STORE :  about to update DM_FILE_DIRECTORY
FINISH_STORE : Return from SQL: Table DM_FILE_DIRECTORY; Command
UPDATE
SQLCODE = 0; #Rows: 1 Comment: update dm_file_directory
FINISH_STORE :  entering insert_backup
INSERT_BACKUP : Entered routine insert_backup
INSERT_BACKUP :  About to select from DM_FILE_DIRECTORY
INSERT_BACKUP : Return from SQL: Table DM_FILE_DIRECTORY; Command
SELECT
SQLCODE = 0; #Rows: 1 Comment: file_directory table
INSERT_BACKUP :  About to insert in DM_FILE_BACKUP
INSERT_BACKUP : Return from SQL: Table DM_FILE_BACKUP; Command
INSERT
```

```
SQLCODE = 0; #Rows: 1 Comment: insert into dm_file_backup table
INSERT_BACKUP : Leaving routine insert_backup
12:39:38.728 FINISH_STORE : Return from insert_backup: with rc of
0 and ec of 0
12:39:38.728 FINISH_STORE :  entering update_storage_pool
UPDATE_STORAGE_POOL : Entered routine update_storage_pool
UPDATE_STORAGE_POOL :  entering pool_count
UPDATE_STORAGE_POOL : Return from pool_count: with rc of 0 and ec
of 0
UPDATE_STORAGE_POOL :  entering updt_pl_tbl
UPDT_PL_TBL : Entered routine updt_pl_tbl
UPDT_PL_TBL :  Update the row contents in table - DM_STORAGE_POOL
UPDT_PL_TBL : Return from SQL: Table DM_STORAGE_POOL; Command
UPDATE
SQLCODE = 0; #Rows: 1 Comment: Update the row contents
UPDT_PL_TBL : Leaving routine updt_pl_tbl
UPDATE_STORAGE_POOL : Return from updt_pl_tbl: with rc of 0 and ec
of 0
UPDATE_STORAGE_POOL : Leaving routine update_storage_pool
FINISH_STORE : Return from update_storage_pool: with rc of 0 and ec
of 0
FINISH_STORE : Leaving routine finish_store
DM_REVOKE :  send_back_reply
SEND_BACK_REPLY : Entered routine send_back_reply
SEND_BACK_REPLY :  Entering send_message
SEND_MESSAGE : Entered routine SEND_MESSAGE
GET_MESSAGE_TYPE : Entered routine get_message_type
GET_MESSAGE_TYPE : Leaving routine get_message_type
12:39:38.778 SEND_MESSAGE : send_message_type = 2
12:39:38.778 SEND_MESSAGE : Return from get_message_type: with rc
of 0 and ec of 0
12:39:38.778 SEND_MESSAGE :  Entering D_MESSAGE
12:39:38.798 SEND_MESSAGE : Return from D_MESSAGE: with rc of 0 and
ec of 0
12:39:38.798 EDM_READMESSAGES : Entered routine edm_readmessages
12:39:38.798 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
12:39:38.798 SEND_MESSAGE : Leaving routine SEND_MESSAGE
12:39:38.798 SEND_BACK_REPLY : Return from send_message: with rc
of 0 and ec of 0
12:39:38.798 SEND_BACK_REPLY : Leaving routine send_back_reply
12:39:38.798 DM_REVOKE : Return from send_back_reply: with rc of 0
and ec of 0
12:39:38.798 DM_REVOKE : Leaving routine dm_revoke
INVOKE_DM_FUNCTION : Return from REVOKE  : with rc of 0 and ec of 0
INVOKE_DM_FUNCTION :  Entering SQL_COMMIT_ONLY
SQL_COMMIT_ONLY : Entered routine sql_commit_only
SQL_COMMIT_ONLY :  About to SQL COMMIT WORK
SQL_COMMIT_ONLY : Return from SQL: Table  ; Command COMMIT
SQLCODE = 0; #Rows: 1 Comment: After commit
SQL_COMMIT_ONLY : Leaving routine sql_commit_only
```

```
INVOKE_DM_FUNCTION : Return from SQL_COMMIT_ONLY: with rc of 0 and
ec of 0
INVOKE_DM_FUNCTION : Leaving routine INVOKE_DM_FUNCTION
12:39:38.999 PROCESS_MESSAGES : Return from INVOKE_DM_FUNCTION:
with rc of 0 and ec of 0
12:39:38.999 PROCESS_MESSAGES : Leaving routine PROCESS_MESSAGES
12:39:38.999 PROCESS_MESSAGES : Entered routine PROCESS_MESSAGES
12:39:38.999 PROCESS_MESSAGES :  Entering RECEIVE_MESSAGE
12:39:38.999 RECEIVE_MESSAGE : Entered routine RECEIVE_MESSAGE
12:39:38.999 RECEIVE_MESSAGE :  edm_getmessages
12:39:38.999 EDM_GETMESSAGES : Entered routine edm_getmessages
12:39:38.999 EDM_GETMESSAGES :  edm_readmessages
EDM_READMESSAGES : Entered routine edm_readmessages
12:39:38.999 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
12:39:38.999 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
12:39:38.999 EDM_GETMESSAGES : rec->conlist[0] = -1
12:39:38.999 EDM_GETMESSAGES :  edm_readmessages
EDM_READMESSAGES : Entered routine edm_readmessages
12:39:38.999 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 0 and ec
of 315
EDM_READMESSAGES :  MESSAGE_INDICATE
EDM_READMESSAGES : conid = 4
EDM_READMESSAGES : Leaving routine edm_readmessages
12:39:39.009 EDM_GETMESSAGES : Return from edm_readmessages: with
rc of 0 and ec of 0
12:39:39.009 EDM_GETMESSAGES : rec->conlist[0] = -1
12:39:39.009 EDM_GETMESSAGES :  Any connection. Got a message
12:39:39.009 EDM_GETMESSAGES : conid = 4
EDM_GETMESSAGES : Leaving routine edm_getmessages
RECEIVE_MESSAGE : Return from edm_getmessages: with rc of 0 and ec
of 0
GET_NORM_MESSAGE_TYPE : Entered routine GET_NORM_MESSAGE_TYPE
GET_NORM_MESSAGE_TYPE : Leaving routine GET_NORM_MESSAGE_TYPE
RECEIVE_MESSAGE : received_message_type = 2
RECEIVE_MESSAGE : Leaving routine RECEIVE_MESSAGE
12:39:39.009 PROCESS_MESSAGES : Return from RECEIVE_MESSAGE: with
rc of 0 and ec of 0
12:39:39.009 PROCESS_MESSAGES :  Entering INVOKE_DM_FUNCTION
INVOKE_DM_FUNCTION : Entered routine INVOKE_DM_FUNCTION
INVOKE_DM_FUNCTION : About to enter:  = REVOKE
DM_REVOKE : Entered routine dm_revoke
DM_REVOKE : Command to revoke:  = STORE
DM_REVOKE : Selection scope:    = F
DM_REVOKE : This item type:     = F
DM_REVOKE : This file name:     = vlt_n1
DM_REVOKE : EOT switch (ON=Y):  = Y
DM_REVOKE : Revoke switch (GOOD=Y):  = Y
FINISH_STORE : Entered routine finish_store
FINISH_STORE : Leaving routine finish_store
```

```
DM_REVOKE :  send_back_reply
SEND_BACK_REPLY : Entered routine send_back_reply
SEND_BACK_REPLY :  Entering send_message
12:39:39.009 SEND_MESSAGE : Entered routine SEND_MESSAGE
GET_MESSAGE_TYPE : Entered routine get_message_type
GET_MESSAGE_TYPE : Leaving routine get_message_type
12:39:39.009 SEND_MESSAGE : send_message_type = 2
12:39:39.009 SEND_MESSAGE : Return from get_message_type: with rc
of 0 and ec of 0
12:39:39.009 SEND_MESSAGE :  Entering D_MESSAGE
12:39:39.069 SEND_MESSAGE : Return from D_MESSAGE: with rc of 0 and
ec of 0
EDM_READMESSAGES : Entered routine edm_readmessages
12:39:39.069 EDM_READMESSAGES :  Entering D_CONSUME_NEXT
EDM_READMESSAGES : Return from D_CONSUME_NEXT: with rc of 25024
and ec of 0
EDM_READMESSAGES : Leaving routine edm_readmessages
12:39:39.069 SEND_MESSAGE : Leaving routine SEND_MESSAGE
12:39:39.069 SEND_BACK_REPLY : Return from send_message: with rc
of 0 and ec of 0
12:39:39.069 SEND_BACK_REPLY : Leaving routine send_back_reply
12:39:39.0
```

Chapter 12

Troubleshooting Common Problems

This chapter describes common Vault-related problems and details of possible solutions.

This chapter is for the system administrators who are familiar with the tools and utilities provided by the operating systems where Optegra is running. If you cannot find help for any specific problem, refer Chapter 11, “Problem Solving.”

- Startup
- File Transfer
- Attributes
- SQL Errors
- Vault Startup
- Storing Parts in Vault
- Vault Administration
- Vault Runtime Errors
- Optegra Client Applications
- Distributed Vault
- Reporting Problems

Startup

This section describes possible startup problems and the corresponding corrective actions.

Vault Connection Problems

Any of the following can be the cause of Vault connection problems:

Problem	Solution
The `pm.config` file is incorrect.	Check the `pm.config` file on the Vault server and edit your local `pm.config` file to exactly match the one on the server.
The `pm.config` file is not world-readable.	Change the permissions on the `pm.config` file so that it is world-readable.
Your system is not defined as a Vault client node in the `nsm.config` file.	Create an entry in the `nsm.config` file for your system.
ORACLE is not up and running.	Start ORACLE.
An incorrect ANSPATH environment variable/logical exists.	Correct the ANSPATH environment variable/logical or redefine it.

Vault Network Problems

Either of the following can be the cause of Vault network problems:

- **Symptom**: Cannot start Vault network processes.

Explanation	Solution
If you use the PCA, one or more GRPCTL statements in the `nsm.config` file or the START statement (located at the end of the `nsm.config file)` is incorrect.	Edit and correct the `nsm.config file`. Shut down and restart the network.
If you do not use the PCA, network processes must be started manually.	Start the network process(es) manually. Enter the `nsmstart` command to start each instance of the DD or DN.

File Transfer

The following are possible file transfer problems and corresponding solutions:

- Symptom: While attempting to replace files in the Vault, you receive a message that no storage pools are available.

Explanation	Solution
If DDs are 75% concurrently active, you need at least one more storage pool than you have DDs. One of your storage pools can be full. Vault uses storage pool space such that the first two or three storage pools can always fill more than subsequent storage pools.	Rerun the IBKUP command, followed by the DELOV command to reclaim database space. A long-term solution is to add one or more storage pools to the database.

- **Symptom**: You are unable to replace a CADDS part.

Explanation	Solution
Your part has new files in it (for example, figures, solid images) that are not defined in the Vault CADDS part definition file, or the Vault system administrator has deleted file types from the CADDS part definition file.	Solution: Verify which CADDS file types are included in the Vault CADDS part definition file, and determine if the file has recently been edited. The CADDS part definition in force when you replace a part in the Vault must contain all the file types included in the CADDS part you obtained with the Vault GET command.

- Symptom: While attempting to GET a Vault file, part, file set, or catalog that is at a released status code (thus creating an entry for the file(s) at the next revision code), you receive a message that the file is not found at the requested revision.

Explanation	Solution
The file, part, file set, or catalog is already at the last revision code in the revision sequence, so you cannot advance it to the next revision.	Solution: Add more revision codes to the revision code sequence.

Attributes

This section describes possible problems related to attribute and provides solutions.

- Symptom: The following error message displays:

```
CDMSTR555E - Processing not done - The supplied attributes did not
satisfy all the required attributes.
```

Explanation	Solution
This message can signal that the attribute table “ATTR_DATA” does not have enough space to store all of the attribute values for the operation. The command generates a diagnostic file, `ATTRFI.OUT` within your current working directory. Check this file for more information. A message in this file similar to the following indicates that insufficient space exists: `$$MGSTEXT=CAMEDM998E Processing not done - Unexpected return code from SQL. Sqlcode=-1,653 Sqltext=ORA-01653:Unable to extend table EDMATTR.ATTR_DATA by 500 in tablespa`. For EPD.Connect, the current working directory is `/tmp/nav`.	Extend the database

SQL Errors

This section provides background information on SQL errors and how to troubleshoot these problems. This information applies to SQL errors with a specific SQL code value.

Consider the following examples:

```
xxxyyy998E=CAMEDM998E Processing not done - Unexpected
return code from SQL. Sqlcode=-1,653 Sqltext=ORA-01653:
unable to extend table EDMATTR.ATTR_DATA by 500 in
tablespace.
```

```
xxxyyy898F - Unexpected SQL error. "ORA-01653: unable to
extend table EDMATTR.ATTR_DATA by 500 in tablespace.
```

Solution: Use the Oracle error utility "`oerr`" to obtain an explanation for this error message. For example,

```
% oerr ora 1653
01653, 00000, "unable to extend table %s.%s by %s in tablespace %s"
Cause:  Failed to allocate an extent for table segment in
tablespace.
Action: Use ALTER TABLESPACE ADD DATAFILE statement to add one or
more files to the tablespace indicated.
```

You can locate the Oracle error utility in the following location:

```
$ORACLE_HOME/bin/oerr
```

The format for the utility is:

```
% oerr ora <message number>
```

When you enter the `oerr` command without any parameters, the following information is provided:

```
% oerr
Usage: $ORACLE_HOME/bin/oerr facility error
Facility is identified by the three-letter prefix in the error
string. For example, for ORA-7300, "ora" is the facility and "7300"
is the error. So you should type "oerr ora 7300".
```

For example, when you see `LCD-111`, type "`oerr lcd 111`".

Vault Startup

If any of the following symptoms occur, the Vault has failed to complete its startup sequence:

- The Vault user account (for example, `edm`) alias `psedm` lists only the processes `ansprcoa` and `anspmgr`. Terminate these processes before trying to restart the Vault.
- The system log file/messages file indicates that not all of the following core processes have started:

  ```
  LOG.STARTUP
  ATTR.STARTUP
  DM.STARTUP
  DD.STARTUP
  ADMN.STARTUP
  ```

- Sign on to the Vault (for example, `cisignon`) gives either of these error messages:

  ```
  CDMSON622F Network error 20,014 - NSM Error.
  ERROR STARTING PMMS
  CDMSON622F Network error 20,029 - NSM Error.
  CANNOT START THIS AE
  ```

The most common reason for the Vault to fail to start up is that the system configuration has been changed in some way, and is now invalid. Use the checklists that follow to confirm and fix the configuration.

Licensing Checklist

- Check that the license file is up-to-date. On UNIX, check the license dates in the file `/usr/CVswlm/epd/epd.licenses`, and on Windows NT, check the file `C:\CVswlmepdepd.lic`.
- Check that the license server can be reached over the network with the `ping` command.
- Check that the license manager process is running with the `lmstat` command (see *Using the License Manager* for more information).
- Check that the license file on the Vault is the same as the license file on the license server.

Please note: Make sure that you update EVERY system when you receive a new license file.

Oracle Checklist

If the Oracle instance is not available to the Vault user account, the system log or messages file can show the following error message from the Vault script:

```
LOG.STARTUP: LOG.STARTUP: SQL connect failed with SQL return code
= -1,034.
```

Check the following causes:

- The Oracle instance is not running (on UNIX, check for processes containing `ora_`; on Windows NT, check the status of the Oracle processes in the Services tool).
- The ORACLE_HOME and/or ORACLE_SID environment variables are not set correctly in the Vault user account. Check that these correspond to the values defined for the Oracle user account (on Windows NT, check the User Profile).

Configuration File Checklist

- Check that the Vault user account home directory is set correctly in the `nsm.config` file, which can be found in the data directory for the Vault user account, `$EPD_HOME/data` or `/usr/apl/edm/data`. The `nsm.config` file defines the startup scripts for all the Vault processes, by specifying their absolute pathname. This should point to the correct location for these scripts on your system.
- Check that the Vault hostname is set correctly in the `pm.config` file, usually found in the same directory as `nsm.config`. This file has the following format:

  ```
  RESOURCE(MYVAULT:::myVault:process_manager_domain:process
  _manager_AE:0,loc,PMGR)
  ```

- The first parameter in the brackets should match the system host name in the opposite case (this is the Vault *domain* name), and the next completed parameter is the host name in the correct case.
- Check also that the Vault host and domain name are correct in the `nsm.config` file.
- Check that there are no extraneous `pm.config` and `nsm.config` files in the current directory (the directory from which you started the Vault). These take precedence over the files in the normal location (the data directory) and can cause the Vault not to start up.

General Checklist

- The Vault requires that the system's Internet Protocol (IP) address is defined in the system's host file (`/etc/hosts` on UNIX). Confirm that this is present and correct.
- The Vault can appear to have failed startup, when, in fact, the Vault's attribute server is performing a `warmstart`, which can take 30 or more minutes. The Vault startup can eventually complete successfully.
- You can eliminate this startup delay by switching off the attribute servers. Do this by adding the following line in the AE definition for the EDMATTR process in the nsm.config file.

  ```
  USER(WARMSTART=NO)
  ```

Storing Parts in Vault

When storing a part in the Vault on NFS-mounted pools on the Solaris with NFS server 2.0, it shows an error. You may not be able to store a part even though there is adequate space on the remote pools and in the `DM_STORAGE_POOL` table.

A solution to this problem is available in the `$EPD_HOME/install` directory. Do the following:

1. Copy the RPC server daemon `$EPD_HOME/install/statvfs_svc` to `/usr/sbin/` of the remote pool machine.
2. Create a new script file `statvfs_svc` in the `/etc/init.d` of the remote pool machine. Doing this will either start or stop the daemon.

   A sample script is as follows:

```
#!/bin/sh
# Statvfs Daemon
killproc() { # kill the named process(es)
  pid=`/usr/bin/ps -e | \
       /usr/bin/grep statvfs_ | \
       /usr/bin/sed -e 's/^  *//' -e 's/ .*//'`
  [ "$pid" != "" ] && kill $pid
}
# Start/stop statvfs_svc
case "$1" in
'start')
  /usr/sbin/statvfs_svc /dev/console 2>&1
# start daemon
  ;;
'stop')
  killproc  # kill daemon
  ;;
*)
  echo "Usage: /etc/init.d/statvfs_svc
  { start | stop }"
  ;;
esac
```

3. Create another script file `S99statvfs_svc` in the `/etc/rc2.d/` directory of the remote pool machine as follows:

```
#!/bin/sh
/etc/init.d/statvfs_svc start &
```

4. Execute the following command:

   ```
   chmod 744 /etc/rc2.d/S99statvfs_svc
   ```

   The daemon automatically starts every time the remote pool machine is booted.

   - To manually start the daemon, enter the following command at the prompt:

     ```
     % /etc/init.d/statvfs_svc start
     ```

   - To manually stop the daemon, enter the following command at the prompt:

     ```
     % /etc/init.d/statvfs_svc stop
     ```

5. Start `statvfs_svc` or the remote pool machine.
6. Restart Vault.

Vault Administration

This section describes possible Vault administration problems and provides solutions.

- Symptom: Attempting to add a storage pool fails (the ADDSP command gives errors).

Explanation	Solution
The local directory specified does not meet the rules for becoming a storage pool.	Ensure that the required local directory satisfies the following requirements: - The local path entered is an empty disk partition, not a directory on a disk that contains other files and directories. - The local directory should appear in the system's mount table (for example, `fstab` or equivalent, or on Windows NT, a disk drive letter). - The disk partition specified by the local path must not contain any files. - The disk partition and mount point must be owned by the Vault user account. - On UNIX, there must be a lost+found directory in the disk partition. This is created by the `newfs` command.

- Symptom: `Unable to extend DM_FILE_DIRECTORY...` reported in system log or messages file and/or Oracle alert log.

Explanation	Solution
The Oracle file space available for the database table that lists the files in the Vault is full. While it is expected that the database can grow as more files are put into the Vault, there are some routine Vault administration tasks that can ensure that this database table is as compact as possible.	Make sure you are running regular incremental backups (`IBKUP`) and delete old versions (`DELOV`).

- Symptom: `Unable to extend DM_FILE_DIRECTORY...` reported in system log or messages file and/or Oracle alert log.

Explanation	Solution
The Oracle file space available for the database table that lists the files on incremental backup tapes is full. This table grows as files are updated on the system, and records the backup copies made. If the backup tapes are recycled, data from this table should be deleted for each tape as it is re-used.	Use the CLEAR and CYCLES parameters to the universal backup command, `ciubkup, to clear` out redundant entries in the incremental backup table as tapes are re-used. See the *Vault Command Reference* for more details.

- Symptom: `Unable to extend DM_FILE_DIRECTORY...` reported in system log or messages file and/or Oracle alert log.

Explanation	Solution
The Oracle file space available for the database table that records Vault activity is full.	Ensure that you delete old logs using the command DELLOG as part of your regular system administration.

- Symptom: Universal backup fails after writing tape with error `cannot open /dev/tty`.

Explanation	Solution
The universal backup command `ciubkup` requires write access to the input/output device on which it displays its messages (`tty`). If you run the command in the background (for example, from the `cron`) or after using `switch user` (`su`), your `tty` does not have write permission.	Log in from scratch to the Vault user account, or start a new window as the Vault user. Also be aware that if the command does fail with this message, it can leave a large number of temporary files in the Vault user account's home directory. These files end with the process ID of the universal backup command, and should be manually deleted.

Vault Runtime Errors

This section describes possible Vault runtime errors and provides solutions.

- Symptom**:** Vault processes terminate unexpectedly.

Explanation	Solution
All files in the /tmp directory have been deleted while the Vault was running.	Stop and restart the Vault.
The `/tmp` directory is full.	Delete some files from `/tmp`, for example, look for core files and delete them.

- Symptom: Users receive `poster` or `waiter` errors.

Explanation	Solution
The system has custom Vault triggers that are not working correctly.	Check if all the triggers that were started with the Vault are still running; if not, check for core dumps in the triggers' working directory. Review the trigger coding for possible errors, particularly in the passing of valid parameters.

- Symptom: Users receive the error `cannot get a license for OptegraOracle` or `OptegraVault`.

Explanation	Solution
The license manager has been restarted while the Vault was running.	Stop and restart the Vault.

- Symptom: Users receive `CADDSRPC` errors from Optegra commands.

Explanation	Solution
There is a firewall between the user's system and the Vault system, which is limiting the network traffic to packets destined for specified TCP port numbers. This is used to restrict access to servers via FTP, TELNET etc.	Vault processes use RPC for client-server communications, which makes use of arbitrary TCP port numbers (assigned by the system's `portmapper`). For this reason, large-scale filtering of TCP port numbers using a firewall is not recommended.

- Symptom: Vault Process Manager (`anspmgr`) does not shut down.

Explanation	Solution
The Process Manager is waiting for processes it controls to terminate, but they have already terminated abnormally without notifying the Process Manager.	Shut down the Process Manager manually (on UNIX, terminate the `anspmgr` process). In the future, before shutting down the Vault, check for "dubious" processes with `nsmquery` and clear them using `nsmflush`, then shut down.

Optegra Client Applications

This section describes possible client application problems and provides solutions.

- Symptom: Users performing file transfers with the Vault receive the error `no executable for ??? found`, where ??? is a rulebase name (for example, CADDS, LOCAL).

Explanation	Solution
The client requires certain local files to be created after the Optegra client software has been installed. For example, `EDM.DEFAULTS`, which defines the rulebase executables.	Ensure that the script `edmcinstall`, located in the install directory for the Optegra client software is run on each client.

- Symptom: `Edmgui` gives segmentation fault on start up, and terminates.

Explanation	Solution
There is an invalid `.Edmgui` file in your home directory.	Delete the `.Edmgui` file.

- Symptom: Commands to transfer files from the Vault receive the error `File/Part is in-use by the system`.

Explanation	Solution
A previous transaction has failed to complete.	Reset the file/part by using `sqlplus` to set the database columns DM_FILE_IN_USE_CD and DM_PART_IN_USE_CD in tables DM_FILE_DIRECTORY and DM_PART_DIRECTORY, respectively, to a single space. Column DM_FILE_IN_USE_CD in table DM_FILE_DIRECTORY is set to a T when a file has been updated/replaced/stored, but the user-defined attributes have failed to update into the ATTR_DATA table (for example, because the table is full). The column can also be set to the first letter of the last command to be executed on the file, for example, G for `get`.

- Symptom: Client commands receive the error `Unable to access message text`.

Explanation	Solution
Explanation: This environment variables LANG and NLSPATH are not set correctly.	Resolution: These variables should be set in the user's account (for example, from the `.cshrc` files). See the template Vault user account files `.login-template` and `.cshrc-template` in the `install` directory of the Vault client software.

Distributed Vault

This section describes possible Distributed Vault problems and provides solutions.

- Symptom: Remote Vaults are offline, as shown by the status column in the table DM_VAULT_CONFIG.

Explanation	Solution
Remote Vaults are not defined correctly in the `nsm.config` file for `PDMUSER, OAXIS_CLIENT, EDMClient.`	Errors usually arise when Vault configuration is changed (for example, host names). To be sure that the Vaults refer to each other correctly, use the commands `ciremVault` and `ciaddVault` to update the distributed Vault configuration when system changes occur. See the *Vault Command Reference* for more information.
Remote Vaults are not defined correctly in the `pm.config` file.	
Remote Vaults not are declared correctly in DM_VAULT_CONFIG. They must be type Vault or DOD, and be referred to by their Vault domain name (usually the hostname in uppercase).	

- Symptom: Importing an object fails.

Explanation	Solution
Vault project definitions are not synchronized between the Vaults.	When using Distributed Vault in combination with Vault projects, write a script to automate the creation of projects so that the projects have the same parameters on every Vault.
Vault `USERIDs` for remote Vaults are not users on the projects.	
VAULTID parameter in the command was not given in uppercase (corresponding to the exporting Vault's domain name).	
There is insufficient disk space in `/tmp` or the `oaxis` staging area (Vault user account home directory).	Provide sufficient space both in /tmp and in the Vault user account home directory for the number of files you anticipate being queued for import.
Vault user account home directory set incorrectly in the account's `.cshrc`.	Amend the setting of EPD_HOME in the `.cshrc`.

- Symptom: All user-defined attributes are not transferred with an object.

Explanation	Solution
The attribute rules are not synchronized between the Vaults.	Create Vault attributes, sets, and rules using a script that you can run on every Vault.

Reporting Problems

1. Review the events that led to your concern. Check the related documentation to ensure the correct use of software commands and system features. Try the commands again in a simple test case. If the error recurs, report it as a problem.
2. Have the following information available before you report the problem:
   - System number
   - Contact name and telephone number
   - Software product being used (for example, Vault)
   - Release of software (for example, Vault 2.0)
   - System being used
   - A brief, but full problem description, including the command or keyword with which there is a concern.
   - Trace the operation (see “Using the Trace Utilities” on page 11-6 for information on generating traces).

Refer to the Preface for information on how to contact Customer Service.

Appendix A

Work Example on Customizing Storage Pool Selection

This chapter provides step-by-step examples of how to customize storage pool selection logic.

- Prerequisite SQL Knowledge
- Example 1: Selection without Typed Storage Pools
- Example 2: Selection with Typed Storage Pools
- Example 3: Generalizing Selection with Typed Storage Pools
- Example 4: Selection Using Multiple Storage Pool Types

Prerequisite SQL Knowledge

The selection queries that examine the Vault database and the pool filters that select a storage pool are based upon the SQL language. All selection queries are `SQL SELECT` commands. The pool filters are where clauses that Vault appends to `SELECT` commands at run time.

The control logic does not use SQL statements. It controls the use of the selection queries and pool filters.

The SQL syntax used is sophisticated but is limited to queries (the `SELECT` command). There is no requirement to understand SQL that is embedded in a program knowledge of interactive SQL queries as entered through query utilities such as SQL*Plus (Oracle) or ISQL (SQL/DS) is sufficient. Particular SQL features employed include:

- Testing for existence—the SQL SELECT COUNT (*)
- Selecting specific rows—the WHERE clause
- Expressions and compound expressions
- Matching a value in a list—IN lists
- Controlling output order—the ORDER BY clause
- Table joins
- Subqueries and multiple subqueries

Depending upon your needs, you might find other SQL features useful. For example:

- Group functions (`MAX`, `MIN`, `AVG`, `SUM`)
- Date functions

The rest of this chapter presents detailed descriptions of how to design, code, and install custom storage pool logic. The examples include comments about how to improve performance and how to generalize the selection logic so that it needs minimal maintenance.

Each example selects storage pools by file classification, either PUB, PRO, or PRI. The final example selects by classification and owner, where owner is either a project or a user ID, depending upon classification. Each example uses SQL in increasingly more sophisticated ways.

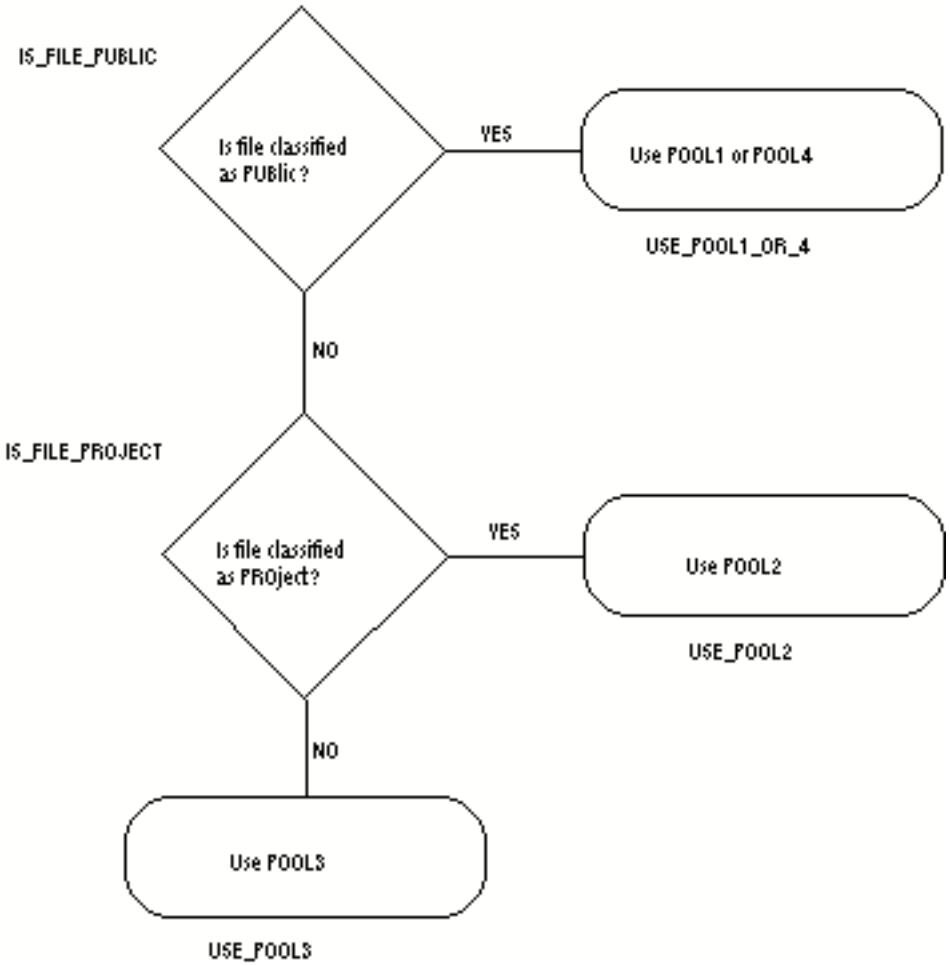
Example 1: Selection without Typed Storage Pools

This example assumes that there are four storage pools assigned to Vault, and that two are used for public files, one for project files, and one for private files.

Steps 1 and 2: Plan the Selection Process

The selection flow is shown below.

Figure A-1 Example 1: Selection Logic



Logic needs two tests to distinguish among the three possible file classifications. It takes a default after the second test. If most of your files are public, the sequence of steps shown is the most efficient because only one test needs to execute to choose a pool filter. If some other file classification is more common, you can change the sequence of steps so that the first test selects the most common classification.

This example does not use storage pool types, so skip step 2. You will see later why storage pool types can make your selection logic more flexible.

Step 3: Code the Control Logic

Use the Control Logic File information in Chapter 3, “Changing Storage Pool Selection”, code the statements needed to implement the diagramed selection flow. Copy the labels for selection queries and pool filters from the Selection Flow diagram. Here is what the control logic file looks like for Example 1:

```
SEQ    SELECTION QUERY  FOUND NOT_FOUND  FOUND_FILTER
NOT_FOUND_FILTER
00000  IS_FILE_PUBLIC     -1      00010  USE_POOL1_OR_4
00010  IS_FILE_PROJECT    -1         -1  USE_POOL2       USE_POOL3
```

Step 4: Code the Selection Queries

This example has two selection queries. Each query is a complete SQL statement that tests one file classification.

Normally when you write SQL queries, you ask SQL to return the value of one or more columns for one or more rows in the database. All of the selection queries for storage pool selection return only a single number, usually the count of the number of rows satisfying the criteria in the query. You can do this by using the COUNT function. For this example, the count function is used simply to test whether a condition is true or false.

```
IS_FILE_PUBLIC
SELECT COUNT (*)
      FROM DM_FILE_DIRECTORY
      WHERE DM_FILE_ROWID = ’&001’ AND
            DM_FILE_CLASS = ’PUB’;

IS FILE PROJECT
SELECT COUNT (*)
      FROM DM_FILE_DIRECTORY
      WHERE DM_FILE_ROWID = ’&001’ AND
            DM_FILE_CLASS = ’PRO’;
```

The substitution placeholder `&001` for the ROWID of the file. The ROWID is a unique key assigned by Vault to the file you are storing. You can use this substitution value anywhere in your SQL statement to constrain the test to consider only the file you are storing now. You do not have to know the precise value of this key, Vault substitutes the ROWID value into the SQL query before executing it.

Step 5: Code the Pool Filters

This example has three pool filters. Each filter is a fragment of a SQL query. The queries here all include a WHERE clause, to test for a particular pool type. Before Vault can use the pool filter, it must construct a complete SQL query that returns all the information Vault needs about the applicable storage pools. Since some pool filters can find more than one storage pool that satisfies the criteria, Vault generates a list using the query

```
SELECT DM_POOL_NAME, DM_POOL_SIZE_FREE ...
  FROM DM_STORAGE_POOL WHERE ...;
```

The pool filters needed for this example are as follows:

```
USE_POOL1_OR_4
WHERE DM_POOL_NAME IN ('POOL1','POOL4')
ORDER BY DM_POOL_PCT_USED;
USE_POOL2
WHERE DM_POOL_NAME = 'POOL2';
USE_POOL3
WHERE DM_POOL_NAME = 'POOL3';
```

Please note: The pool filter named USE_POOL1_OR_4 can select either of two pools. This is accomplished by using the SQL IN list. It is functionally equivalent to stating

```
WHERE DM_POOL_NAME = 'POOL1' OR DM_POOL_NAME = 'POOL4' ...
```

To exercise some control over which pool is selected, pool filter USE_POOL1_OR_4 includes an ORDER BY clause. This ensures that whichever pool has the most free space is used for the next file stored. Without the ORDER BY clause SQL can select either pool at random.

Note that these pool filters fail to select a storage pool if the named pool is not available for any reason. If Vault cannot find a storage pool for your file using the selected pool filter, your file is not stored and Vault displays an error message. For example, if Vault fails to store a project file, it shows the message:

```
CDMSTR441E Processing not done - the storage pools
selected by the USE_POOL2 query are unavailable.
Please notify your Vault administrator.
```

Step 6: Edit Script File ldedmspl

Now that all three files are coded, you can prepare the script file **ldedmspl** found in the Vault installation directory. Add a line for each file you have prepared. FD1 is the name of the control file, FD2 contains the selection queries, and FD3

contains the pool filters. Here, all file names are prefixed with `EXAMPLE1.` You can name the files anything you like.

```
FD1=EXAMPLE1.POOLQUST
FD2=EXAMPLE1.POOLSRCH
FD3=EXAMPLE1.POOLFLTR
```

Be sure you comment out any other values in the script for files FD1=, FD2=, and FD3=.

Step 7: Load the Template

Load the selected template into temporary EDM SQL tables by running the script file `ldedmspl` that you have just edited.

If the load fails, check for coding errors such as character data entered into numeric-only fields. Correct the errors and rerun.

Step 8: Make the New Code the Production Version

If you want to make the code just loaded the production version, run script file `edmispl.` This script checks the temporary tables for logic errors. If none are found, it swaps the current production logic with the logic in the temporary tables.

If the install fails, return to step 3, 4, or 5 to correct the error. If you want to check the new code for logic errors but not install it, run the script `edmespl.`

Step 9: Test the New Production Logic

Test the new production logic. Try storing files that fit every category of storage pool to exercise all pool filters.

If you find errors, you can back out the selection logic most recently installed. You can do so by running the script `edmbspl` which restores the system to its status prior to step 8 above.

How Good Is This Design?

This example uses a simple SQL logic. The logic is easy to read. But there are disadvantages in designing storage pool selection this way. Each time you add or change storage pools, you must recode your selection logic and pool filters and install and test them. The next example illustrates how using typed storage pools can eliminate the need to change your installed logic when you add or change storage pools.

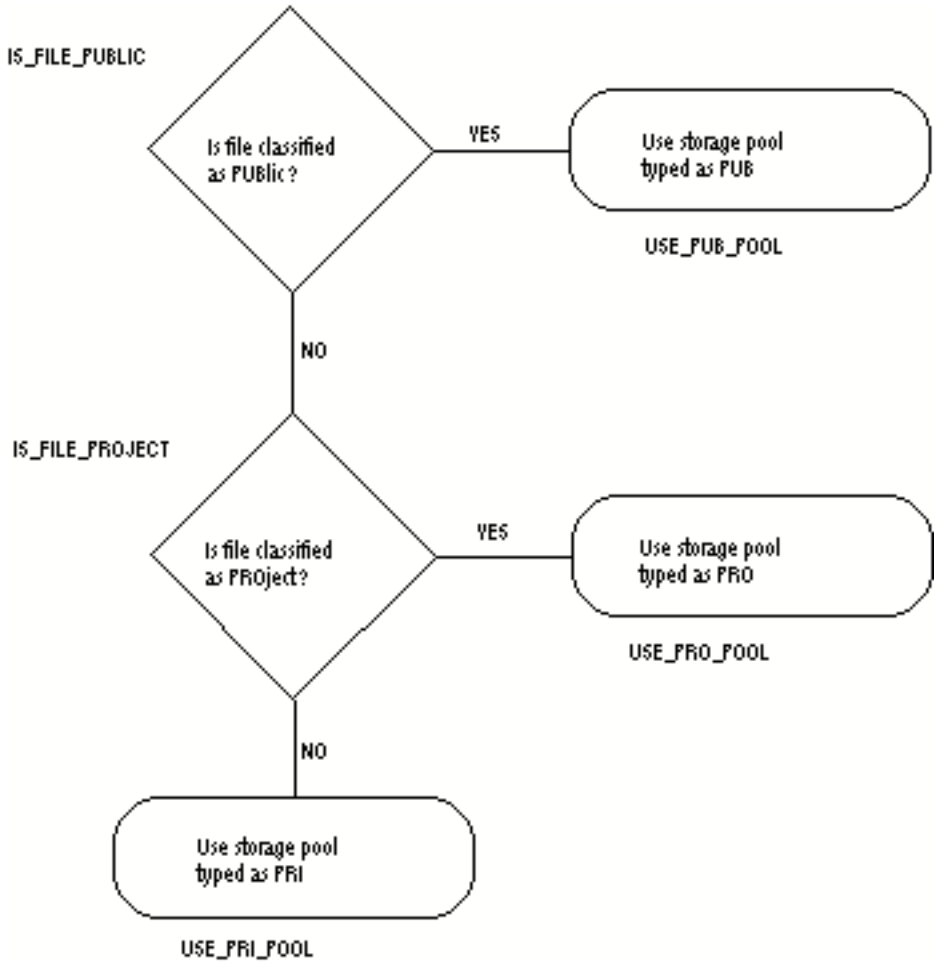
Example 2: Selection with Typed Storage Pools

In this example, the same four storage pools are used as before. This time, type them with the labels PUB, PRO, and PRI to correspond to the file classifications. After you type the storage pools, you can change the pool filters to search for pools by type instead of name. When you add storage pools in the future, you can use them as soon as they are typed. You no longer need to change the selection logic to use new pools.

Step 1: Design the Selection Flow

The selection flow changes little from the previous example. Only the pool filter names and descriptions change. The logic is illustrated below.

Figure A-2 Example 2: Selection Logic



Step 2: Assign Types to the Storage Pools

Before you implementing this design, assign types to the existing storage pools. The design calls for storage pools to have types identical to the values used for the file classification field—PUB, PRO, and PRI.

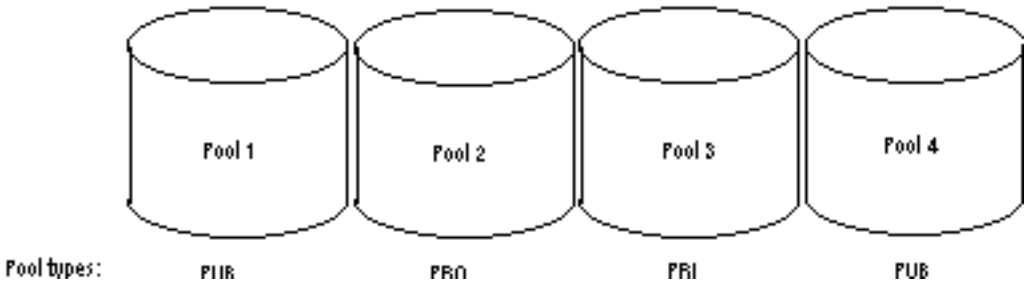
Depending upon the mix of public, project, and private files at your site, you can assign one classification to several pools or two classifications to one pool. Untyped pools are never used since all files have one of the three classifications. For this example, there are four pools, two for public files and one each for private and project files.

Storage pool types are assigned using the `CHGSPT` command. You can use the Vault command line format to assign pool types as follows:

```
CICHGSPT POOLNAME=Pool1 POOLTYPE=PUB CHGTYPE=A
CICHGSPT POOLNAME=Pool2 POOLTYPE=PRO CHGTYPE=A
CICHGSPT POOLNAME=Pool3 POOLTYPE=PRI CHGTYPE=A
CICHGSPT POOLNAME=Pool4 POOLTYPE=PUB CHGTYPE=A
```

The following diagram illustrates the pools with their assigned types.

Figure A-3 Pools with Assigned Types



Step 3: Code the Control Logic

Here is what the control logic file looks like for Example 2:

```
SEQ    SELECTION QUERY  FOUND NOT_FOUND   FOUND_FILTER
NOT_FOUND_FILTER
00000  IS_FILE_PUBLIC    -1     00010  USE_PUB_POOL
00010  IS_FILE_PROJECT    -1         -1   USE_PRO_POOL
USE_PRI_POOL
```

Step 4: Code the Selection Queries

The selection queries can remain the same as in the previous example.

```
IS_FILE_PUBLIC
SELECT COUNT (*)
      FROM DM_FILE_DIRECTORY
      WHERE DM_FILE_ROWID = '&001' AND
      DM_FILE_CLASS = 'PUB';
IS_FILE_PROJECT
SELECT COUNT (*)
       FROM DM_FILE_DIRECTORY
       WHERE DM_FILE_ROWID = '&001' AND
       DM_FILE_CLASS = 'PRO';
```

Step 5: Code the Pool Filters

The pool filters needed for this example must examine pool types and return a pool name and other information for the chosen storage pools. Pool types are in a different table than the pool information selected by the part of this query which precedes the WHERE clause. To examine pool types, you must use a subquery which searches the pool type table and returns appropriate names.

```
USE_PUB_POOL
WHERE DM_POOL_NAME IN
(SELECT DM_POOL_NAME FROM DM_POOL_TYPE
 WHERE DM_POOL_TYPE = 'PUB')
ORDER BY DM_POOL_PCT_USED;
USE_PRO_POOL
WHERE DM_POOL_NAME IN
(SELECT DM_POOL_NAME FROM DM_POOL_TYPE
 WHERE DM_POOL_TYPE = 'PRO')
ORDER BY DM_POOL_PCT_USED;
USE_PRI_POOL
WHERE DM_POOL_NAME IN
(SELECT DM_POOL_NAME FROM DM_POOL_TYPE
 WHERE DM_POOL_TYPE = 'PRI')
ORDER BY DM_POOL_PCT_USED;
```

Step 6: Edit Script File ldedmspl

Now that all three files are coded, you can prepare the script file `ldedmspl` found in the Vault installation directory. Add a line for each file you have prepared. FD1 is the name of the control file, FD2 contains the selection queries, and FD3 contains the pool filters. Here, all file names are prefixed with `EXAMPLE2.` You can name the files anything you like.

```
FD1=EXAMPLE2.POOLQUST
FD2=EXAMPLE2.POOLSRCH
FD3=EXAMPLE2.POOLFLTR
```

Be sure you comment out any other values in the script for `files FD1=, FD2=, and FD3=.`

Step 7: Load the Template

Load the selected template into temporary EDM SQL tables by running the script file `ldedmspl` that you have just edited. If the load fails, check for coding errors such as character data entered into numeric-only fields. Correct the errors and rerun.

Step 8: Make the New Code the Production Version

If you want to make the code just loaded the production version, run script file `edmispl.` This script checks the temporary tables for logic errors. If none are found, it swaps the current production logic with the logic in the temporary tables.

If the install fails, return to step 3, 4, or 5 to correct the error.

If you want to check the new code for logic errors but not install it, run the script `edmespl.`

Step 9: Test the New Production Logic

Test the new production logic. Try storing files that fit every category of storage pool to exercise all pool filters.

If you find errors, you can back out the selection logic most recently installed. Do so by running the script `edmbspl` which restores the system to its status prior to step 8 above.

What Have You Gained?

With this new design, you never have to change the storage pool selection when you add or delete storage pools. Instead, when you add storage pools, you can assign them to one of the file classifications and Vault can begin to use them immediately.

One disadvantage is that two queries are needed when a file does not match the initial file classification. If you use this approach with some criteria for classifying files that has many values instead of the three values for classification, you require many queries and the selection process becomes unwieldy. A solution to this problem is illustrated in the next example.

Example 3: Generalizing Selection with Typed Storage Pools

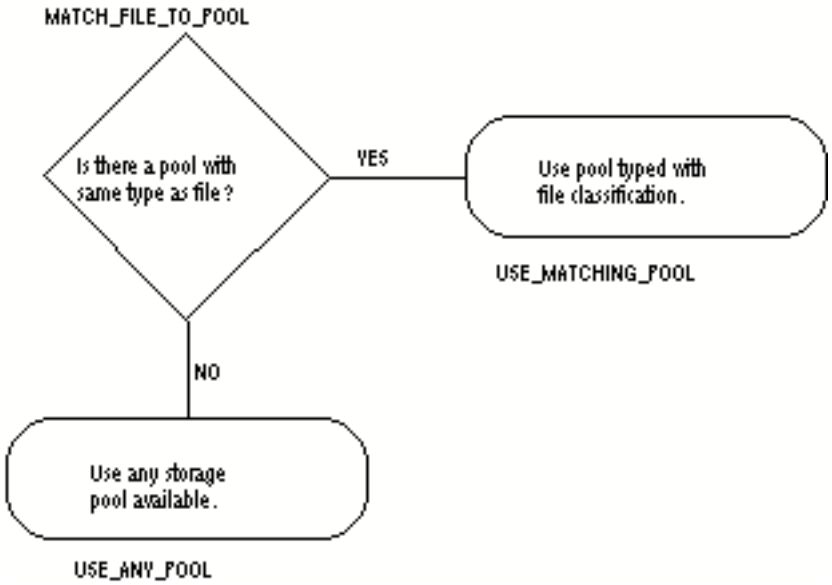
This example uses the same four storage pools types as the previous example. This time, storage pools are selected by matching their types directly to file classification.

Step 1: Design the Selection Flow

One selection query and one pool filter accomplish the same classification done previously with two queries and three filters.

The default branch in this example is chosen only if, at the time the query is executed, there is no storage pool typed with the file classification. The logic is illustrated below.

Figure A-4 Example 3 Selection Logic



Step 2: Assign Types to Storage Pools

Assign the same storage pool types as in the previous example.

```
CICHGSPT POOLNAME=Pool1 POOLTYPE=PUB CHGTYPE=A
CICHGSPT POOLNAME=Pool2 POOLTYPE=PRO CHGTYPE=A
CICHGSPT POOLNAME=Pool3 POOLTYPE=PRI CHGTYPE=A
CICHGSPT POOLNAME=Pool4 POOLTYPE=PUB CHGTYPE=A
```

Step 3: Code the Control Logic

Here is what the control logic looks like for Example 3:

```
SEQ   SELECTION QUERY  FOUND NOT_FOUND FOUND_FILTER
NOT_FOUND_FILTER
00000 MATCH_FILE_TO_POOL  -1        -1 USE_MATCHING_POOL
USE_ANY_POOL
```

Step 4: Code the Selection Queries

This example has only one selection query, but it is complex because it joins two tables. Instead of testing the classification of the file, this query directly tests the storage pool type by comparing it to the file's classification.

```
MATCH_FILE_TO_POOL
SELECT COUNT (*)
      FROM  DM_POOL_TYPE
            DM_FILE_DIRECTORY
      WHERE DM_FILE_ROWID = '&001' AND
            DM_FILE_CLASS = DM_POOL_TYPE;
```

Step 5: Code the Pool Filters

The pool filters also become more complex. The first pool filter combines a subquery with a join of the file directory and pool type tables.

```
USE_MATCHING_POOL
WHERE DM_POOL_NAME IN
      (SELECT DM_POOL_NAME
            FROM  DM_POOL_TYPE,
                  DM_FILE_DIRECTORY
            WHERE DM_FILE_ROWID = '&001' AND
                  DM_FILE_CLASS = DM_POOL_TYPE)
ORDER BY DM_POOL_PCT_USED;
```

The second pool filter finds any available storage pool in the event that none are typed identically to the file classification. It does not need a WHERE clause, but it does need the ORDER BY clause.

```
USE_ANY_POOL
ORDER BY DM_POOL_PCT_USED;
```

Step 6: Edit Script File ldedmspl

Now that all three files are coded, you can prepare the script file `ldedmspl` found in the Vault installation directory. Add a line for each file you have prepared. FD1 is the name of the control file, FD2 contains the selection queries, and FD3

contains the pool filters. Here, all file names are prefixed with `EXAMPLE3.` You can name the files anything you like.

```
FD1=EXAMPLE3.POOLQUST
FD2=EXAMPLE3.POOLSRCH
FD3=EXAMPLE3.POOLFLTR
```

Be sure you comment out any other values in the script for `files` FD1=, FD2=, and FD3=.

Step 7: Load the Template

Load the selected template into temporary EDM SQL tables by running the script file `ldedmspl t`hat you have just edited. If the load fails, check for coding errors such as character data entered into numeric-only fields. Correct the errors and rerun.

Step 8: Make the New Code the Production Version

If you want to make the code just loaded the production version, run script file `edmispl.` This script checks the temporary tables for logic errors. If none are found, it swaps the current production logic with the logic in the temporary tables.

If the install fails, return to step 3, 4, or 5 to correct the error.

If you want to check the new code for logic errors but not install it, run the script `edmespl.`

Step 9: Test the New Production Logic

Test the new production logic. Try storing files that fit every category of storage pool to exercise all pool filters.

If you find errors, you can back out the selection logic most recently installed. Do so by running the script `edmbspl` which restores the system to its status prior to step 8 above.

What Have You Gained?

These queries are more efficient. When you use the approach of assigning storage pools types against file fields that can have many values, for instance project or user ID, you can realize considerable savings.

Example 4: Selection Using Multiple Storage Pool Types

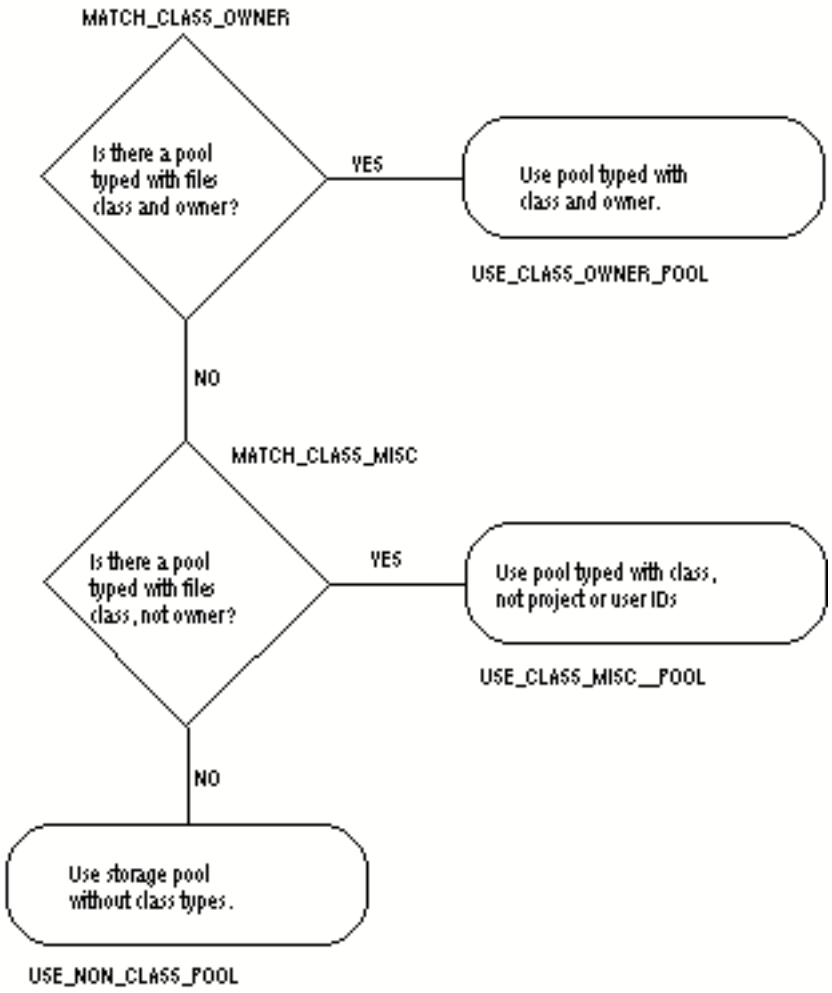
Storage pool selection is often based on multiple tests. Here, project files are placed in pools typed by both PRO and project ID. Private files are placed in pools typed with PRI and user ID. The example is designed to work properly even when a user and project have the same ID. Public files are allowed only in pools not allocated for project or private files.

Step 1: Design the Selection Flow

One selection query determines whether there are any storage pools typed with both the proper file classification and project or user ID. Note that the file directory owner column contains project ID for project files and user ID for private files. A pool with the correct owner but wrong class is not selected.

If no storage pool satisfies the first test, a second test finds any storage pools with the correct classification but not assigned to any projects or users. If the second test fails also, any storage pool not typed PUB, PRO, or PRI is used. The logic is illustrated in Figure A-6.

Figure A-5 Example 4 Selection Logic



Step 2: Assign Types to Storage Pools

The storage pools for this example are typed so that Pool1 is used for all public files, Pool2 for projects PROJ1 and PROJ2, Pool3 for miscellaneous project files, and Pool4 for private files belonging to Wright and Markam. Pool5 picks up anything left over. Since all files are either public, project, or private, the leftovers consist of private files not belonging to Wright or Markam.

Pool3 is not assigned a type of MISC to make the selection for miscellaneous projects easier. This example simply uses a different approach for finding the miscellaneous category.

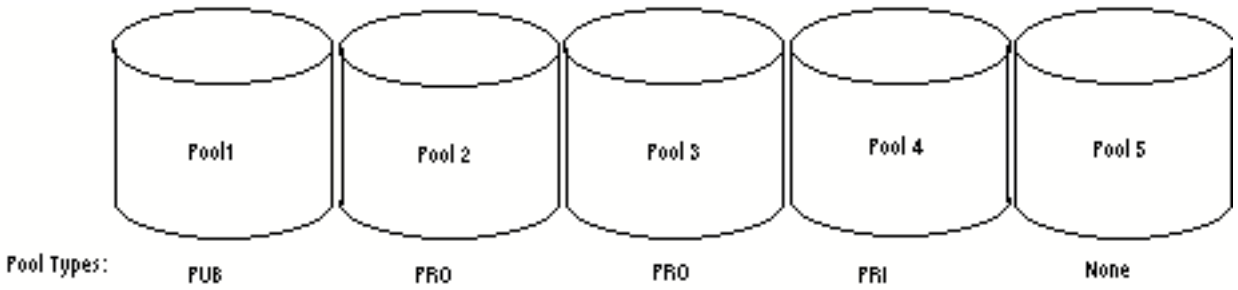
Some of the storage pool types have changed from the previous examples.

If you want to change pool types, the following commands work:

```
CICHGSPT POOLNAME=Pool2 POOLTYPE=PROJ1 CHGTYPE=A
CICHGSPT POOLNAME=Pool2 POOLTYPE=PROJ2 CHGTYPE=A
CICHGSPT POOLNAME=Pool3 POOLTYPE=PRI CHGTYPE=R
CICHGSPT POOLNAME=Pool3 POOLTYPE=PRO CHGTYPE=A
CICHGSPT POOLNAME=Pool4 POOLTYPE=PUB CHGTYPE=R
CICHGSPT POOLNAME=Pool4 POOLTYPE=PRI CHGTYPE=A
CICHGSPT POOLNAME=Pool4 POOLTYPE=WRIGHT CHGTYPE=A
CICHGSPT POOLNAME=Pool4 POOLTYPE=MARKAM CHGTYPE=A
CICHGSPT POOLNAME=Pool5 POOLTYPE=NONE CHGTYPE=A
```

The following diagram illustrates the five pools with their assigned types.

Figure A-6 Storage Pools and Assigned Types



Step 3: Code the Control Logic

Here is what the control logic looks like for Example 3:

```
SEQ   SELECTION QUERY  FOUND NOT_FOUND FOUND_FILTER
NOT_FOUND_FILTER
00000 MATCH_CLASS_OWNER   -1     00010 USE_CLASS_OWNER_POOL
00010 MATCH_CLASS_MISC    -1       -1 USE_CLASS_MISC_POOL
USE_NON_CLASS_POOL
```

Step 4: Code the Selection Queries

This example has two selection queries, each with multiple subqueries. The first determines if there are any pools typed with both the file's classification and it's owner.

For a precise count, specify SELECT COUNT (DISTINCT DM_POOL_NAME), but this is not necessary for determining existence. The second query determines if there are any pools typed with the correct classification but not typed with the ID of any project or user.

```
MATCH_CLASS_OWNER
SELECT COUNT (*) FROM DM_POOL_TYPE
  WHERE DM_POOL_NAME IN
    (SELECT DM_POOL_NAME
      FROM DM_POOL_TYPE, DM_FILE_DIRECTORY
    WHERE DM_FILE_ROWID = '&001' AND
        DM_FILE_CLASS = DM_POOL_TYPE)
  AND DM_POOL_NAME IN
    (SELECT DM_POOL_NAME
      FROM DM_POOL_TYPE, DM_FILE_DIRECTORY
     WHERE DM_FILE_ROWID = '&001' AND
        DM_FILE_OWNER_ID = DM_POOL_TYPE);
MATCH_CLASS_MISC
SELECT COUNT (*) FROM DM_POOL_TYPE
  WHERE DM_POOL_NAME IN
    (SELECT DM_POOL_NAME
      FROM  DM_POOL_TYPE, DM_FILE_DIRECTORY
     WHERE DM_FILE_ROWID = '&001' AND
        DM_FILE_CLASS = DM_POOL_TYPE)
  AND DM_POOL_NAME NOT IN
    (SELECT DM_POOL_NAME FROM DM_POOL_TYPE
      WHERE DM_POOL_TYPE IN
        (SELECT DM_USER_ID FROM DM_USER))
  AND DM_POOL_NAME NOT IN
    (SELECT DM_POOL_NAME FROM DM_POOL_TYPE
      WHERE DM_POOL_TYPE IN
        (SELECT DM_PROJ_ID FROM DM_PROJECT));
```

Step 5: Code the Pool Filters

The pool filters also become more complex, but they use the same subqueries as the selection queries.

```
USE_CLASS_OWNER_POOL
WHERE DM_POOL_NAME IN
  (SELECT DM_POOL_NAME
    FROM DM_POOL_TYPE, DM_FILE_DIRECTORY
   WHERE DM_FILE_ROWID = '&001' AND
      DM_FILE_CLASS = DM_POOL_TYPE)
AND DM_POOL_NAME IN
  (SELECT DM_POOL_NAME
    FROM DM_POOL_TYPE, DM_FILE_DIRECTORY
  WHERE DM_FILE_ROWID = '&001' AND
      DM_FILE_OWNER_ID = DM_POOL_TYPE)
ORDER BY DM_POOL_PCT_USED;

USE_CLASS_MISC_POOL
WHERE DM_POOL_NAME IN
  (SELECT DM_POOL_NAME
    FROM  DM_POOL_TYPE, DM_FILE_DIRECTORY
    WHERE DM_FILE_ROWID = '&001' AND
      DM_FILE_CLASS = DM_POOL_TYPE)
```

```
AND DM_POOL_NAME NOT IN
  (SELECT DM_POOL_NAME FROM DM_POOL_TYPE
    WHERE DM_POOL_TYPE IN
      (SELECT DM_USER_ID FROM DM_USER))
AND DM_POOL_NAME NOT IN
  (SELECT DM_POOL_NAME FROM DM_POOL_TYPE
    WHERE DM_POOL_TYPE IN
      (SELECT DM_PROJ_ID FROM DM_PROJECT))
ORDER BY DM_POOL_PCT_USED;

USE_NON_CLASS_POOL
WHERE DM_POOL_NAME NOT IN
  (SELECT DM_POOL_NAME FROM DM_POOL_TYPE
    WHERE DM_POOL_TYPE IN ('PUB','PRO','PRI'))
ORDER BY DM_POOL_PCT_USED;
```

Step 6: Edit Script File ldedmspl

Now that all three files are coded, prepare the script file `ldedmspl` found in the Vault installation directory. Add a line for each file you have prepared. FD1 is the name of the control file, FD2 contains the selection queries, and FD3 contains the pool filters. Here, all file names are prefixed with `EXAMPLE4.` You can name the files anything you like.

```
FD1=EXAMPLE4.POOLQUST
FD2=EXAMPLE4.POOLSRCH
FD3=EXAMPLE4.POOLFLTR
```

Be sure you comment out any other values in the script for `files` FD1=, FD2=, and FD3=.

Step 7: Load the Template

Load the selected template into temporary EDM SQL tables by running the script file `ldedmspl` that you have just edited. If the load fails, check for coding errors such as character data entered into numeric-only fields. Correct the errors and rerun.

Step 8: Make the New Code the Production Version

Run script file `edmispl,` to make the code just loaded the production version. This script checks the temporary tables for logic errors. If none are found, it swaps the current production logic with the logic in the temporary tables.

If the install fails, return to step 3, 4, or 5 to correct the error.

If you want to check the new code for logic errors but not install it, run the script `edmespl`.

Step 9: Test the New Production Logic

Test the new production logic. Store files that fit every category of storage pool to exercise all pool filters.

Back out the selection logic most recently installed, if you find errors. Do so by running the script `edmbspl` which restores the system to its status prior to step 8 above.

Appendix B

Command Abbreviations

This appendix lists Vault and IQF (Interactive Query Facility) command abbreviations. These abbreviations are primarily used as identifiers in messages.

- Vault Command Abbreviations
- IQF Command Abbreviations

Vault Command Abbreviations

The following table lists the Vault command abbreviations.

Table B-1 Vault Command Abbreviations with their descriptions

Abbreviation	Command	Description
AAG	ADDAG	Add an authority group to Vault
AAS	ADDASET	Add an attribute set to Vault
AAT	ADDATTR	Add an attribute to Vault
ACL	ADDCL	Add a command list to Vault
ADM	ADMCOPY	Administrative copy
ADP	ADDP	Add a project to Vault
ADS	ADDS	Add a status level to an authority scheme
ADT	ADDT	Add a user-defined table to the control of Vault
ADU	ADDU	Add a user to Vault
AFS	ADDFS	Add a file set
ALS	ADDUSA	Add a user list/status code association
AMA	ADDMAS	Add a member to an attribute set
AML	ADDMUL	Add members to a user list
AMS	ADDMFS	Add a member to a file set
ARC	ARCHIVE	Move files marked for archiving from Vault to tape
ARL	ADDRULE	Add a classification rule to Vault
ARS	ADDRS	Add a revision code sequence
ASP	ADDSP	Add a storage pool to Vault
AUL	ADDUL	Add a user list name
AUP	ADDUP	Add a user to a project
CAG	CHGAG	Change entries in an authority group
CCL	CHGCL	Change entries in a command list
CFA	CHGFA	Change file(s) attributes
CFC	CHGFCL	Change file(s) classification
CFP	CHGFPW	Change file(s) password
CFR	CHGFREV	Change file(s) revision code
CFS	CHGFSC	Change file(s) status code
CLT	CLST	Close a user-defined table
CMA	CHGMAS	Change a member of an attribute set
CPA	CHGP	Change the attributes of a project

Table B-1 Vault Command Abbreviations with their descriptions

Abbreviation	Command	Description
CPS	CHGSPS	Change storage pool status
CPT	CHGSPT	Change storage pool type
CPW	CHGUPW	Change your Vault user password
CPY	COPY	Copy file(s) to new Vault file(s)
CSA	CHGS	Change the attributes of a status level
CTL	CHGCTL	Change command trigger list
CUA	CHGU	Change an Vault user's attributes and/or authority
CUP	CHGUP	Change a user's project authority
DAG	DELAG	Delete an authority group from Vault
DAS	DELASET	Delete an attribute set from Vault
DAT	DELATTR	Delete an attribute from Vault
DCL	DELCL	Delete a command list from Vault
DEL	DELETE	Delete files marked for deletion
DIR	LISTDIR	List file(s) on local or remote Vault node
DLG	DELLOG	Delete Vault audit log entries
DLP	DELP	Delete a project from Vault
DLS	DELS	Delete a status level from an authority scheme
DLU	DELU	Delete a user from Vault
DLV	DELOV	Delete old versions of files
DRL	DELRULE	Delete a classification rule from Vault
DRS	DELRS	Delete a revision code sequence
DUL	DELUL	Delete a user list
EDM	EDM	General Vault messages not related to any command
FSP	CHGFSP	Change a file's storage pool
GET	GET	Sign out and copy file(s); modifications allowed
HLP	HELP	Vault help command
IBU	IBKUP	Back up new and modified files
IQF	IQF	Interactive Query Facility
LOA	LOAD	Load file(s) from tape to Vault
MKA	MARKA	Mark file(s) to be archived

Table B-1 Vault Command Abbreviations with their descriptions

Abbreviation	Command	Description
MKD	MARKD	Mark file(s) to be deleted
MKR	MARKR	Mark file(s) to be restored
OPT	OPNT	Open a user-defined table
PUR	PURGE	Delete file(s) not previously marked for deletion
REA	READ	Copy file(s); modifications not allowed
REP	REPLACE	Save modified file(s); ability to modify ends
RES	RESTORE	Restore files from tape to Vault
RFS	REMFS	Remove a file set
RLS	REMUSA	Remove a user list/status code association
RMA	REMMAS	Remove a member from an attribute set
RMG	READMSG	Read messages
RML	REMMUL	Remove a member from a user list
RMS	REMMFS	Remove a member from a file set
RMT	REMT	Remove a user-defined table from Vault control
RSF	RECSF	Recover a single file
RSP	RECSP	Recover a storage pool
RST	RESET	Cancel signing out of file(s); changes are lost
RSV	RESERVE	Define and sign out an Vault file name for future use
RUP	REMUP	Remove a user from a project
RVP	RSVP	Respond to a review request
RVW	REQRVW	Request review of file(s)
SCN	SCANTAPE	List the contents of an Vault tape
SMG	SENDMSG	Send a message
SOF	SIGNOFF	Sign off and exit Vault session
SON	SIGNON	Sign on to Vault or another Vault user ID
SOT	SIGNOUT	Sign out file(s) to modify; file(s) not copied
STR	STORE	Add a new file
UBU	UBKUP	Back up entire Vault database
UMK	UNMARK	Unmark previously marked files
UNL	UNLOAD	Unload files to tape
UPT	UPDATE	Save modified file(s); continue modifications

IQF Command Abbreviations

The following table lists the IQF command abbreviations.

Table B-2 IQF Command Abbreviations with their descriptions

Abbreviation	Command	Description
BAK	BACKWARD	Scrolls a display towards the beginning
BOT	BOTTOM	Scrolls a display so the end is visible
BRK	BREAK	Inserts one or more empty lines into the display
CIF	COMMAND INTERFACE	Calls the command interface
COL	COLUMN	Scrolls a display left to the specified column
EDT	EDIT	Accepts multiple IQF commands and then executes all
END	END	Ends display of a query result or help text
EXT	EXIT	Terminates IQF
FIL	FILE	Copies query result to a local file
FMT	FORMAT	Formats a display
FWD	FORWARD	Scrolls a display towards the end
HLP	HELP	Displays online IQF help
LFT	LEFT	Scrolls the display to the left
LST	LIST	Displays the current query
PRT	PRINT	Sends the query result to a printer
REC	RECALL	Retrieves the last successful SELECT command
RET	RETRIEVE	Retrieves the previous IQF command
RUN	RUN	Executes the current query
RYT	RIGHT	Scrolls the display to the right
SAV	SAVE	Copies the current query to a local file
SEL	SELECT	Specifies information to view
SET	SET	Assigns values to format parameters
SHO	SHOW	Displays the current format parameter settings
STA	START	Executes a current or stored query
TIF	TERMINAL INTERFACE	Calls the terminal interface
TOP	TOP	Scrolls a display towards the beginning
TYP	TYPE	Displays an IQF command file

Index

Numerics

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V

W