

Cincom

SUPRA SERVER PDM

Logging and Recovery Guide
(OS/390 & VSE)

P26-2223-64



SUPRA[®] Server PDM Logging and Recovery Guide (OS/390 & VSE)

Publication Number P26-2223-64

© 1989–1994, 1997, 1998, 2000, 2002 Cincom Systems, Inc.
All rights reserved

This document contains unpublished, confidential, and proprietary information of Cincom. No disclosure or use of any portion of the contents of these materials may be made without the express written consent of Cincom.

The following are trademarks, registered trademarks, or service marks of Cincom Systems, Inc.:

AD/Advantage [®]	iD CinDoc [™]	MANTIS [®]
C+A-RE [™]	iD CinDoc Web [™]	Socrates [®]
CINCOM [®]	iD Consulting [™]	Socrates [®] XML
Cincom Encompass [®]	iD Correspondence [™]	SPECTRA [™]
Cincom Smalltalk [™]	iD Correspondence Express [™]	SUPRA [®]
Cincom SupportWeb [®]	iD Environment [™]	SUPRA [®] Server
CINCOM SYSTEMS [®]	iD Solutions [™]	Visual Smalltalk [®]
 gOOj [™]	intelligent Document Solutions [™]	VisualWorks [®]
	Intermax [™]	

UniSQL[™] is a trademark of UniSQL, Inc.
ObjectStudio[®] is a registered trademark of CinMark Systems, Inc.

All other trademarks are trademarks or registered trademarks of their respective companies.

Cincom Systems, Inc.
55 Merchant Street
Cincinnati, Ohio 45246-3732
U.S.A.

PHONE: (513) 612-2300
FAX: (513) 612-2000
WORLD WIDE WEB: <http://www.cincom.com>

Attention:

Some Cincom products, programs, or services referred to in this publication may not be available in all countries in which Cincom does business. Additionally, some Cincom products, programs, or services may not be available for all operating systems or all product releases. Contact your Cincom representative to be certain the items are available to you.

Release information for this manual

The *SUPRA Server PDM Logging and Recovery Guide (OS/390 & VSE)*, P26-2223-64, is dated January 15, 2002. This document supports Release 2.7 of SUPRA Server.

We welcome your comments

We encourage critiques concerning the technical content and organization of this manual. Please take the [survey](#) provided with the online documentation at your convenience.

Cincom Technical Support for SUPRA Server PDM

FAX: (513) 612-2000
Attn: SUPRA Server Support

E-mail: helpna@cincom.com

Phone: 1-800-727-3525

Mail: Cincom Systems, Inc.
Attn: SUPRA Server Support
55 Merchant Street
Cincinnati, OH 45246-3732
U.S.A.



Contents

About this book	ix
Using this document.....	ix
Document organization	x
Revisions to this manual	x
Conventions	xi
SUPRA Server documentation series	xiv
Using SUPRA logging	17
Using SUPRA logging	18
Backing off changes.....	19
Restarting tasks	19
Reapplying changes.....	20
Analyzing activity	20
Selecting types of SUPRA PDM logging	21
PDM task logging	21
PDM system logging	23
Identifying your logging needs.....	24
Creating the PDM task log file	25
Reviewing the task log file's attributes	26
Defining the task log file in the bootstrap modules.....	27
Defining the task logging environment with directory maintenance	32
Suppressing task logging	35
Placing the task log file	35
Formatting the task log file	36
Logical record types written in the task log file image blocks.....	37
Logical record types written in the task log file SFT blocks.....	38

Creating PDM system log file groups	39
Choosing logging options	41
Choosing log group characteristics.....	43
Multiple logical volumes.....	43
File attributes	44
Group attributes	45
Buffers	45
Defining system logging in the bootstrap modules	46
Defining the log groups with directory maintenance	47
Placing the disk log files	50
Initializing system log files.....	51
Logical record types written in the System Log File.....	53
Running SUPRA with logging	55
Recovering files	56
Backing up data files.....	57
Committing tasks	57
Saving special data on log files	58
Starting a new logical volume on the PDM system log.....	59
Resetting tasks	59
Purging dangling tasks	60
Restarting tasks.....	61
Warm starting the PDM.....	61
Recovering from PDM task and system failure	62
Task failure	62
System failure	63
Defining special cases and problems in recovery.....	66
Recovering KSDS VSAM files	67
Secondary key populate or depopulate function considerations	68
Warm start or recovery failure on secondary keys	68
Processing isolation with single-task PDM	69
Recovering a file during a format function	69
Multiple PDM recovery.....	70

Recovering SUPRA CICS applications	71
Preparing CICS and SUPRA for recovery	72
Recovering CICS application programs	72
How CICS and the PDM recover	74
Recovering after a CICS task abend.....	75
Recovering RDML applications.....	75
Recovering MANTIS applications	76
Recovering directory maintenance.....	76
Recovering after a CICS system abend.....	77
Recovering RDML applications.....	78
Recovering MANTIS applications	78
Recovering directory maintenance.....	78
Recovering after a PDM system abend	79
Recovering RDML applications.....	80
Recovering MANTIS applications	80
Recovering directory maintenance.....	80
Recovering after both CICS and the PDM abend	81
Recovering RDML applications.....	81
Recovering MANTIS applications	81
Recovering directory maintenance.....	81
 Recovering and restoring the SUPRA PDM from the System Log	 83
Considerations	84
Running recover and restore in the appropriate environment.....	85
Determining the need for the Log File I/O Exit.....	87
Switching system log files	87
Using tape data sets	88
Using disk data sets.....	90
Combining both disk and tape data sets.....	92
 Using the Log File I/O Exit in recovery	 93
Using the Log File I/O Exit under OS/390	94
Creating the load module CSUORCUX	96
Coding UCL for the Log File I/O Exit program	97
Coding job control language	98
Listing data sets	100
Handling errors.....	103
Creating a system log to use with the exit program	104
Processing the Log File I/O Exit program	105

Using the Log File I/O Exit under VSE.....	109
Compiling and link editing modules	111
Coding UCL for the Log File I/O Exit program	112
Coding job control language	113
Coding the volume list	114
Sending messages to the output listing.....	118
Sending messages to the console.....	119
Processing the Log File I/O Exit program.....	120
Glossary of terms	127
Index	131

About this book

Using this document

The guide explains how to plan and implement the several types of SUPRA Server logging and how to use the log files to recover database information after a system problem occurs. It covers the following topics:

- ◆ The SUPRA PDM Task Log File and Task Level Recovery (TLR)
- ◆ The SUPRA PDM System Log
- ◆ The SUPRA PDM System Log multiple logical volumes feature

To effectively use this manual, you should have a general knowledge of SUPRA Server features. You should also know how to plan and implement a SUPRA Server database system.

This document is a reorganized and updated version of the SUPRA Server 1.3.5 *Logging and Recovery Guide*. SUPRA 2.1.6 and higher reflect the following internal improvements in the SUPRA Server software and responses to customer requests:

- ◆ Sections on reviewing and unlocking SUPRA PDM files, format of the UCL Control Section and Recovery Functions, and Writing Recovery Exits have been moved to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260.
- ◆ New sections detail writing logical record types and switching system log files.
- ◆ As of release 2.4: New file level log suppression in the File to Environment Description gives the ability to suppress task logging for a single file.

Document organization

The information in this manual is organized as follows:

Chapter 1—Using SUPRA logging

Describes SUPRA logging and gives an overview of how to use it.

Chapter 2—Creating the PDM task log file

Describes how to define and format the PDM Task Log File dimensions for your operating environment.

Chapter 3—Creating PDM system log file groups

Describes the System Log File and how to change the characteristics to meet your requirements.

Chapter 4—Running SUPRA with logging

Describes procedures to follow before, during, and after logging.

Chapter 5—Recovering SUPRA CICS applications

Describes the recovery facilities of CICS and the SUPRA PDM.

Chapter 6—Recovering and restoring the SUPRA RDM from the System Log

Describes the Recover, Restore, and Log Print functions.

Chapter 7—Using the Log File I/O Exit in recovery

Describes a Log File I/O Exit for running the recovery functions of the SUPRA DBA utilities.

Glossary of terms

Index

Revisions to this manual

The following changes have been made for this release:

- ◆ Some updates have been made to the CICS-related information in Chapter 5, “**Recovering SUPRA CICS applications**” starting on page 71.

Conventions

The following table describes the conventions used in this document series:

Convention	Description	Example
Constant width type	Represents screen images and segments of code.	<pre>PUT 'customer.dat' GET 'miller\customer.dat' PUT '\DEV\RMT0'</pre>
Slashed b (<i>b</i>)	Indicates a space (blank). The example indicates that four spaces appear between the keywords.	BEGN b b b b SERIAL
Brackets []	Indicate optional selection of parameters. (Do not attempt to enter brackets or to stack parameters.) Brackets indicate one of the following situations:	
	A single item enclosed by brackets indicates that the item is optional and can be omitted.	[WHERE <i>search-condition</i>]
	The example indicates that you can optionally enter a WHERE clause.	
	Stacked items enclosed by brackets represent optional alternatives, one of which can be selected.	<u>(WAIT)</u> (NOWAIT)
	The example indicates that you can optionally enter either WAIT or NOWAIT. (WAIT is underlined to signify that it is the default.)	

Convention	Description	Example
Braces { }	<p>Indicate selection of parameters. (Do not attempt to enter braces or to stack parameters.) Braces surrounding stacked items represent alternatives, one of which you must select.</p> <p>The example indicates that you must enter ON or OFF when using the MONITOR statement.</p>	<pre>MONITOR {ON } {OFF }</pre>
<u>Underlining</u> (In syntax)	<p>Indicates the default value supplied when you omit a parameter.</p> <p>The example indicates that if you do not choose a parameter, the system defaults to WAIT.</p> <p>Underlining also indicates an allowable abbreviation or the shortest truncation allowed.</p> <p>The example indicates that you can enter either STAT or STATISTICS.</p>	<pre>[(WAIT)] [(NOWAIT)]</pre> <pre><u>STATISTICS</u></pre>
Ellipsis points...	<p>Indicate that the preceding item can be repeated.</p> <p>The example indicates that you can enter multiple host variables and associated indicator variables.</p>	<pre>INTO :host-variable [:ind- variable],...</pre>

Convention	Description	Example
UPPERCASE lowercase	In most operating environments, keywords are not case-sensitive, and they are represented in uppercase. You can enter them in either uppercase or lowercase.	COPY MY_DATA.SEQ HOLD_DATA.SEQ
<i>Italics</i>	Indicate variables you replace with a value, a column name, a file name, and so on. The example indicates that you must substitute the name of a table.	FROM <i>table-name</i>
Punctuation marks	Indicate required syntax that you must code exactly as presented. () parentheses . period , comma : colon ' ' single quotation marks	<i>(user-id, password, db-name)</i> INFILE 'Cust.Memo' CONTROL LEN4
SMALL CAPS	Represent a required keystroke. Multiple keystrokes are hyphenated.	ALT-TAB
<div style="border: 1px solid black; padding: 2px; display: inline-block; margin-bottom: 5px;">OS/390</div> <div style="border: 1px solid black; padding: 2px; display: inline-block; margin-bottom: 5px;">VSE</div>	Information specific to a certain operating system is flagged by a symbol in a shadowed box (<div style="border: 1px solid black; padding: 2px; display: inline-block;">OS/390</div>) indicating which operating system is being discussed. Skip any information that does not pertain to your environment.	<div style="border: 1px solid black; padding: 2px; display: inline-block; margin-bottom: 10px;">OS/390</div> See the SUPRA Server procedure library member TIS\$RDM for a list of RDM procedures. <div style="border: 1px solid black; padding: 2px; display: inline-block; margin-bottom: 10px;">VSE</div> See the SUPRA Server RDM sublibrary member TXJ\$INDX for a list of JCL.

SUPRA Server documentation series

SUPRA Server is the advanced relational database management system for high-volume, update-oriented production processing. A number of tools are available with SUPRA Server including DBA Functions, DBAID, precompilers, SPECTRA, and MANTIS. The following list shows the manuals and tools used to fulfill the data management and retrieval requirements for various tasks. Some of these tools are optional. Therefore, you may not have all the manuals listed. For a brief synopsis of each manual, refer to the *SUPRA Server PDM Digest (OS/390 & VSE)*, P26-9062.

Overview

- ◆ *SUPRA Server PDM Digest (OS/390 & VSE)*, P26-9062

Getting started

- ◆ *SUPRA Server PDM Migration Guide (OS/390 & VSE)*, P26-0550*
- ◆ *SUPRA Server PDM CICS Connector Systems Programming Guide (OS/390 & VSE)*, P26-7452

General use

- ◆ *SUPRA Server PDM Glossary*, P26-0675
- ◆ *SUPRA Server PDM Messages and Codes Reference Manual (RDM/PDM Support for OS/390 & VSE)*, P26-0126

Database administration tasks

- ◆ *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250
- ◆ *SUPRA Server PDM Directory Online User's Guide (OS/390 & VSE)*, P26-1260
- ◆ *SUPRA Server PDM Directory Batch User's Guide (OS/390 & VSE)*, P26-1261
- ◆ *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260
- ◆ *SUPRA Server PDM Logging and Recovery (OS/390 & VSE)*, P26-2223
- ◆ *SUPRA Server PDM Tuning Guide (OS/390 & VSE)*, P26-0225
- ◆ *SUPRA Server PDM RDM Administration Guide (OS/390 & VSE)*, P26-8220
- ◆ *SUPRA Server PDM RDM PDM Support Supplement (OS/390 & VSE)*, P26-8221
- ◆ *SUPRA Server PDM RDM VSAM Support Supplement (OS/390 & VSE)*, P26-8222
- ◆ *SUPRA Server PDM Migration Guide (OS/390 & VSE)*, P26-0550*
- ◆ *SUPRA Server PDM Windows Client Support User's Guide*, P26-7500*
- ◆ *SPECTRA Administrator's Guide*, P26-9220

Application programming tasks

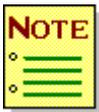
- ◆ *SUPRA Server PDM DML Programming Guide (OS/390 & VSE)*, P26-4340
- ◆ *SUPRA Server PDM RDM COBOL Programming Guide (OS/390 & VSE)*, P26-8330
- ◆ *SUPRA Server PDM RDM PL/1 Programming Guide (OS/390 & VSE)*, P26-8331
- ◆ *SUPRA Server PDM Migration Guide (OS/390 & VSE)*, P26-0550*
- ◆ *SUPRA Server PDM Windows Client Support User's Guide*, P26-7500*

Report tasks

SPECTRA User's Guide, P26-9561



Manuals marked with an asterisk (*) are listed more than once because you use them for multiple tasks.



Educational material is available from your regional Cincom education department.

1

Using SUPRA logging

Logging automatically records a process for the purpose of undoing or redoing that process. Recovery is the undoing or redoing of a process with the help of logged information.

SUPRA logging makes a record of changes to the contents of SUPRA files or to the status of SUPRA files or tasks. SUPRA recovery reverses or reapplies the logged changes.

Using SUPRA logging

Logging permits recovery, restoration, or resetting of a failed database. It assumes a point exists in the processing cycle where the data is valid. Data images and function images may be recorded in a number of ways and at different times in a processing cycle. Logging also captures control information during the processing cycle.

Two types of logging are available: task logging and system logging. You can use either, or both, or no logging at all.



Cincom recommends you use both.

System logging allows the entire system to be recovered if the Task Log File is unreadable or unusable, or if an updated data file is unreadable and must be reloaded. When you use system logging but not task logging, Task Level Recovery is not active (non-TLR).

You define your logging requirements by coding statements in the bootstrap Directory environment description or through Directory Maintenance for your user environment description. For more information on the statements you code in the bootstrap Directory environment description, see [“Creating the PDM task log file”](#) on page 25. For more information on the statements you code in your user environment description, refer to the [SUPRA Server PDM and Directory Administration Guide](#), P26-2250. The following sections discuss both types of logging, the types of log files, and the logical record types logged.

File dependent logging is available in SUPRA Server 2.4 and above. File dependent logging allows you to suppress logging for individual files. Suppressing logging for individual files is suggested only for the MANTIS and SPECTRA Slide files. If logging is suppressed for any other individual files, it is the DBA’s responsibility to ensure that data integrity and recoverability needs are not compromised. To ensure consistent level of recoverability, all interconnected Primary, Related, and Index files must have the same task, system before image, and system after image log options.

Backing off changes

Logging allows you to back off or reverse recent changes to a specified point called a commit point. This ensures that all data and control information affected by the changes are restored to a state where they are internally consistent.

SUPRA Server can use logged data to back off changes made by a task, including updates to file contents and opening or closing of files, to the task's most recent commit point. Each application specifies its own commit points and can back off the changes.

An application can back off its own tasks' changes to the last commit by issuing a reset instruction.

SUPRA PDM automatically backs off the changes of some or all of its tasks to the last commit in response to a forced disconnect command, forced termination command, abnormal termination, or restart following an abnormal termination.

With the Recover function of the SUPRA DBA Utilities, you can back off all the PDM's tasks' changes after an abnormal termination either to the last commit or back through the earliest change logged. For information about coding the Recover function, refer to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260.

Restarting tasks

When a task terminates abnormally for any reason, the task's changes are backed off to its last commit point. If this commit point is not the task's sign-on point, SUPRA Server considers the task to be still logically signed on. Enough information is retained in the Task Log File to restart the task from that commit point.

The capability to back off changes to the last commit in SUPRA Server, combined with the capability to restart a task that ended abnormally, is called Task Level Recovery.

Reapplying changes

Logging also provides the capability to reapply changes that were logged. This becomes necessary when SUPRA PDM files are lost or damaged, for example, through device failure.

To reapply changes, reload the affected files from your backup copies and run the Restore function of the SUPRA DBA utilities. This function reapplies the logged changes starting with the oldest logged change and ending with either the latest logged change or latest commit point for each task.



Cincom recommends restoring to the latest commit points. For information about coding the Restore function, refer to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260.

Analyzing activity

Logging can also provide analysis information on the activity being logged. You can use SUPRA DBA Utilities to print the contents of the PDM System Log or you can write your own applications to read these files.

Selecting types of SUPRA PDM logging

The SUPRA PDM performs the task and system logging. It logs different types of information and data. It is important to use the right type of logging since there are limits on the types of logged data's usefulness.

PDM task logging

The PDM saves enough information for all tasks and transactions to perform task level recovery. The PDM monitors the activity of each task and prevents task interference.

The PDM writes to the Task Log File during normal processing to record a task's activity. When abnormal processing is required (user RESET or task failure), the PDM reads the Task Log File and backs out the task's update activity to its last commit point.

The benefits you gain from using task logging are:

- ◆ One task's failure need not adversely affect subsequent tasks. The PDM ensures data integrity.
- ◆ Updated transactions which have been applied successfully can be backed out. If you do not want the updates applied (even if there was no problem), you can have those updates backed out (or undone) using RESET, provided you have not committed (COMIT) since the updates were done.

The Task Log File consists of three sections:

- ◆ **Task Log ID block** contains a file control record with the last status of the PDM (whether recovery is necessary at all, or if recovery is partially completed). The Task Log ID block functions as the Task Log File's lock record. The block also contains information (PDM name, Directory schema and environment description pair) to ensure the Task Log File is matched with the correct PDM. This is the first block in the Task Log File.
- ◆ **Image blocks** contain information about changes to the database by tasks in the system (file mode before images, record before images, commit records, secondary key change log records, and secondary key block log records).
- ◆ **System File Table (SFT) blocks** contain information about the current status of files and information necessary to rebuild the PDM environment (descriptions of each file in the PDM and its buffering, status of each file, and active environment description).

The SFT starts at the end of the Task Log File and continues toward the start of the Task Log File. If image blocks and SFT blocks meet, the Task Log File is full.

If you use task logging and a forced termination or system failure occurs, you must use the same Task Log File and PDM name that was in use when the forced termination or failure occurred the next time you execute SUPRA Server. Also, you must not update your boot schema, environment description, or validation module. By using the same Task Log File and PDM name, the following events occur:

- ◆ All uncommitted task update activity at the time of forced termination or system failure is undone.
- ◆ All user database files and Directory files are returned to the last commit point.
- ◆ All files are returned to the last committed open mode (Closed, Read, EUPD, IUPD, SUPD).
- ◆ Restartable tasks can retrieve data stored at the last COMMIT. This allows tasks to restart from the point of failure. Tasks that are not restartable are backed out to their last commit point and all associated PDM resources are discarded.

Task logging ensures that changes made to a database are written to the Task Log File first. For information on running the PDM with task logging, see [“Defining the task log file in the bootstrap modules”](#) on page 27.

PDM system logging

When you specify system logging, the PDM saves chronological audit information to one or more sequential files. This is called the System Log File Group. You can partially control the type of information (before images, after images, PDM commands, sign-ons, sign-offs) saved on the System Log File in the Directory environment description or the REALM environment description. Refer to the *SUPRA Server PDM Directory Online User's Guide (OS/390 & VSE)*, P26-1260, or the *SUPRA Server PDM Directory Batch User's Guide (OS/390 & VSE)*, P26-1261, for more information on controlling system logging.

The System Log File Group(s) stores the information that system logging gathers in System Log Files. A System Log File is a sequential string of blocks. Each block contains a block prefix, all or part of a log record (or multiple records), and a block suffix. Logical records are variable length, but blocks are fixed length. Fixed-blocked record format makes reading a disk System Log File more efficient. The System Log File(s) is also buffered for better performance.

The PDM writes to the System Log File Group during normal processing, but does not read the System Log File Group. No image records are written to the System Log File Group during a warm start. The Recovery, Restore, and Log Print Utilities read the System Log File Group. Refer to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260, for more information.

System logging optionally provides for:

- ◆ Database file device failure
- ◆ Task Log File device failure
- ◆ History of changes (by task) for debugging

If you specify synchronization of the System Log File, system logging ensures changes made to a database file are written to the System Log File before they are made to the database. The PDM writes any System Log File buffer connected to a database buffer, waits for the write to complete and breaks the connection before writing the database buffer.

If you create your System Log File using an IBM 3480 tape drive, you must code a DCB=OPTCD=W parameter in the System Log File's DD statement. This is needed to ensure the integrity of the System Log File.

The PDM supports redundant system logging (all information goes to separate log files concurrently) to allow for failure of your System Log File. The PDM also supports a logical System Log File through log groups made up of one or more physical data sets (log file switching).

Identifying your logging needs



For the SUPRA PDM, Cincom recommends you specify task logging and system logging, with system logging options that include before-image logging and after-image logging. Task logging and system logging provide the maximum recovery options for PDM database files and for restart of PDM tasks.

Logging creates significant overhead in your SUPRA Server system. To measure the overhead in your installation, you may want to perform test runs with and without logging. After carefully considering the information in this manual, you may decide to do without some of the logging capabilities offered here.

All SUPRA logging and recovery takes place at the physical component level for the SUPRA PDM. Only byte-string files and PDM database files are logged and can be recovered by SUPRA Server in non-CICS environments. For example, RDM can access non-PDM files, such as native VSAM files, but cannot recover such files. The SPECTRA Personal File System (PFS), which can be a PDM database file or a non-PDM file, provides another example. A non-PDM PFS allows faster access, but only a PDM PFS can be logged and recovered.

Logging is not a substitute for backups. You must periodically make backups of all files containing valuable data.

2

Creating the PDM task log file

You may define and format the PDM Task Log File dimensions for your operating environment. If you want to use the Task Log File, you must have PDM task logging defined for your operating system. Your installation tape includes a description which you may use, or you can define your Task Log File to meet your needs.

Reviewing the task log file's attributes

To review the Task Log File description on your installation tape, look at the Modify Schema input transaction used to define it (see “[Defining the task log file in the bootstrap modules](#)” on page 27) or use Interactive Services, as follows:

1. Bring up SUPRA. You can use any existing CSIPARM member that specifies task logging, or you can create and use a new member with these contents:

```
DIRECTORY=( SCHEMA=CSTASCHM, ENVDESC=CSTATLOG ) , END.
```

2. Log on with your user ID and password. At the Software Selection menu, select Interactive Services.
3. At the Interactive Services main menu, select Physical File Services.
4. At the Physical File Services “select option” screen, specify physical file information and the Task Log File name, TLOG.

For more details on Interactive Services, refer to the *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250.

The Task Log File must be a BDAM or ESDS file. You can run Modify Schema to change any of the task log characteristics. See “[Defining the task log file in the bootstrap modules](#)” on page 27 for more information on changing the description of your Task Log File. Your installation tape also includes several bootstrap Directory environment descriptions. Some of these have task logging specified and provide buffering options for the Task Log File. If the installation tape’s description of the buffering does not meet your requirements, you must run the Create Environment Description to change the buffering.

The block size of the Task Log File must be large enough to accommodate the largest amount of information describing a data file to the PDM. See “[Defining the task log file in the bootstrap modules](#)” on page 27 for the Task Log File space calculation.

Before you can use the Task Log File, you must allocate it and then format it using the Format function of the SUPRA DBA utilities. For more information on the Format function of the SUPRA DBA utilities and to see a JCL sample to format the Task Log File, refer to the *SUPRA Server PDM DBA Utilities User’s Guide (OS/390 & VSE)*, P26-6260.

Defining the task log file in the bootstrap modules

To run the PDM with task logging, the following must be true:

- ◆ The Task Log File must be defined in the CSTASCHM boot schema supplied on the installation tape.
- ◆ A boot environment description must have the TASK-LOGGING option set to YES and the Task Log File must be related to a buffer pool. Such an environment description is suitable for running with task logging. The boot environment description CSTATLOG supplied on the installation tape fits these requirements.
- ◆ A boot environment description must have the Task Log File related to a buffer pool and the TASK-LOGGING option must be set to NO. Such an environment description is suitable for formatting the Task Log File. The boot environment description CSTANONE supplied on the installation tape fits these requirements.

If you want to change the characteristics of the Task Log File, change its buffering, or add a new environment description with task logging, you must use some combination of the bootstrap utility programs: Modify Schema, Create Environment Description, and Create Validation Module. The installation tape provides sample input members for these programs. These inputs are used in creating the bootstrap modules provided on the tape.

You must run the bootstrap utility programs in the proper order.

- ◆ If you are changing the boot schema:
 1. Run Modify Schema.
 2. Run Create Environment Description for any boot environment you are changing or adding.
 3. Run Create Validation Module.
- ◆ If you are not changing the boot schema:
 1. Run Create Environment Description for each boot environment description you are changing or adding.
 2. Run Create Validation Module.

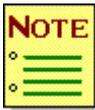
You must run Create Validation Module even if you do not modify its input.

The Modify Schema input that refers to the Task Log File consists of statements similar to the following:

```
FILE=TLOG
ACCESS-METHOD=BDAM
DDNAME=C$TTLOG
BLOCKSIZE=8192
LOGICAL-BLOCKS-PER-TRACK=5
TOTAL-LOGICAL-RECORDS=900
END-FILE
```

If you consider changing the Task Log File's characteristics, be aware of the following:

- ◆ TOTAL-LOGICAL-RECORDS when applied to a Task Log File actually means total blocks.



Cincom recommends a file size of about 1000 blocks. As more tasks execute simultaneously and more updates are performed by each task in a single transaction, more space is used on the file. Space is freed for reuse by the commit at the end of each transaction. If your database activity exceeds the Task Log File's capacity, the PDM terminates abnormally with an appropriate message.



Cincom recommends a block size of at least 3000 bytes. The minimum block sizes are 512 bytes for BDM and 505 bytes for ESDS VSAM. The PDM performs best if the block size is large enough to hold the largest logical records in your database.

- ◆ For an ESDS VSAM Task Log File, the block size should be seven bytes less than the control interval size. The control interval size must be 512, 1024, or a multiple of 2048.



Your choices for ACCESS-METHOD are BDAM and ESDS. Cincom recommends that you use BDAM because it has less access method overhead processing.

- If ACCESS-METHOD is ESDS, you must omit LOGICAL-BLOCKS-PER-TRACK.
- If ACCESS-METHOD is BDAM and you change BLOCKSIZE, you must specify the correct value for LOGICAL-BLOCKS-PER-TRACK. Refer to the appropriate IBM Reference Summary to calculate the correct value for the direct access device you are using.
- If you change ACCESS-METHOD from ESDS to BDAM, you must specify the correct value for LOGICAL-BLOCKS-PER-TRACK.
- If ACCESS-METHOD is BDAM and you do not change BLOCKSIZE, specifying LOGICAL-BLOCKS-PER-TRACK is optional. If you do specify a value for LOGICAL-BLOCKS-PER-TRACK, it must be the correct value.

The Create Environment Description input for the boot environment description CSTATLOG includes statements that are relevant to task logging. These statements are similar to the following:

```
FILE=TLOG,C$BL,SUPD  
BUFFER=C$BL=(15,0,0)  
TASK-LOGGING=YES
```

The only statement you can change is the number of direct buffers in the buffer pool.



Cincom recommends 15, as shown here. You can reduce this number to save memory, but performance may be degraded.

The Task Log File should have its own buffer pool. The number of serial threads and the number of serial buffers per thread in that buffer pool should always be specified as 0, as shown here.

The Create Environment Description input for the boot environment description CSTANONE includes statements that are relevant to task logging. These statements are similar to the following:

```
FILE=TLOG,C$BL,SUPD  
BUFFER=C$BL=(2,0,0)  
TASK-LOGGING=NO
```



Cincom recommends that you change nothing in this input.

The Create Validation Module input includes statements like the following:

```
SCHEMA=CSTASCHM
ENVDESC=CSTANONE
ENVDESC=CSTAREAD
ENVDESC=CSTASUPD
ENVDESC=CSTATLOG
```

If you are adding boot environment descriptions, add ENVDESC= statements with their names. Note that all ENVDESC= statements must follow the corresponding SCHEMA= statement.

For more details on using the bootstrap utility programs, refer to the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250.

The location of the input files for the bootstrap utility programs and the directions for running the programs vary according to your operating system:

- ◆ For OS/390 and VSE, use the sample JCL library members TXJBMODS, TXJBENVD, and TXJBVMOD to run Modify Schema, Create Environment Description, and Create Validation Module, respectively. Each of these members points to its input.

Defining the task logging environment with directory maintenance

Your Directory environment, as well as your boot environment, must contain appropriate definitions to run with task logging. To run Directory-driven with task logging, the following must be true:

- ◆ Your Directory schema must contain a definition for the Task Log File.
- ◆ Your Directory environment description must define a buffer pool and relate the Task Log File to the buffer pool.
- ◆ Your Directory environment description must specify the task-logging option.

The Directory schema BURRYSCH and the Directory environment description BURRYENV, supplied on the Directory on your installation tape, are suitable for running Directory-driven with task logging.

To use your own Directory schema with task logging, add the Task Log File to the schema:

1. At the Directory Maintenance category menu, enter FI to select the file category.
2. At the appropriate prompts on the file command menu, enter AD to select the ADD command and enter the schema name and the Task Log File name (TLOG).
3. At the first file add screen, enter the ddname as C\$TTLOG, file type as TASKLOG, and the access method as the Task Log File's access method (BDAM or ESDS). You need not enter or change any other fields on the screen. You need not access the other two file add screens.
4. Return to the Directory Maintenance category menu.

At run time, the PDM uses the boot schema's specifications for block size, number of blocks, and so on, and ignores those of the Directory schema.

To use your own Directory environment description with task logging, modify the environment description as follows:

1. Specify the task-logging option:
 - a. At the Directory Maintenance category menu, enter ED to select the environment description category.
 - b. At the appropriate prompts on the environment description command menu, enter CG to select the CHANGE command and enter the schema name and environment description name.
 - c. At the first environment description change screen, enter the task log option as Y (yes). You need not enter or change any other fields on the screen. You need not access the second environment description change screen.
 - d. Return to the Directory Maintenance category menu.
2. Add a buffer pool for the Task Log File:
 - a. At the Directory Maintenance category menu, enter BP to select the buffer pool category.
 - b. At the appropriate prompts on the buffer pool command menu, enter AD to select the ADD command and enter the schema name, environment description name, and four-character name you choose for the new buffer pool.
 - c. At the buffer pool add screen, enter the direct buffer count, the serial buffer count, and the serial thread count as 2, 0, and 0, respectively.
 - d. Return to the Directory Maintenance category menu.
3. Relate the Task Log File to the new buffer pool:
 - a. At the Directory Maintenance category menu, enter FI to select the file category.
 - b. At the appropriate prompts on the file command menu, enter RC to select the RELATIONSHIP CHANGE command and enter the schema name and Task Log File name (TLOG).
 - c. At the appropriate prompts on the file relate change screen, enter the environment description name, an open mode of SUPD, and the name of the new buffer pool.
 - d. Return to the Directory Maintenance category menu.

The number of buffers in this new buffer pool is not important. At run time, the PDM ignores this buffer pool and uses the buffer pool related to the Task Log File in the boot environment description.

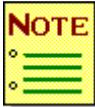
Finally, after you use Directory Maintenance to update a schema, you must consistency check the schema to make it and its environment descriptions usable:

1. At the Directory Maintenance category menu, enter SC to select the schema category.
2. At the appropriate prompts on the schema command menu, enter CK to select the CHECK command and enter the schema name.
3. At the schema check screen, specify Y (yes) for INCON PHYSICAL ENTITIES and ALL LOGICAL ENTITIES. Specify N (no) for the remaining prompts. Allow time for the function to complete.
4. Return to the Directory Maintenance category menu.

For more details on using Directory Maintenance, refer to the *SUPRA Server PDM Directory Online User's Guide (OS/390 & VSE)*, P26-1260, or the *SUPRA Server PDM Directory Batch User's Guide (OS/390 & VSE)*, P26-1261.

Suppressing task logging

When defining an environment description to file relationship in SUPRA Server 2.4 and above, you may choose to suppress task logging for the file. Suppress logging for individual files only for the MANTIS and SPECTRA Slide files. Do not suppress logging for any other files.



You may not suppress logging for a Directory file.

If you suppress task logging for any individual files, then before and after system logging must also be suppressed. It is the DBA's responsibility to ensure that data integrity and recoverability needs are not compromised. For recoverability purposes, all interconnected Primary, Related, and Index files must have the same task, system before image, and system after image log options. The DBA should use Security Groups and Maintenance Restriction to deny users the ability to AD/CG the environment description to file relationships logging options.

Logging suppress options are only considered if the Environment Description specifies that particular type of PDM logging.

To use File Level Log Suppression:

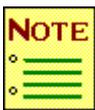
At the Task Logging Option field, enter YES to perform logging per the Environment Description. Enter NO to suppress logging for the file.



If either after image or before image logging is not suppressed, then task logging may not be suppressed. Formatting the Task Log File

Placing the task log file

Log files are the most frequently accessed files in the PDM.



For the most efficient use of your direct access devices, Cincom recommends that the Task Log File reside on a device separate from your most frequently accessed user files.

Formatting the task log file

Use the Format function of the SUPRA DBA utilities to format the Task Log File after its creation and before its first use. Format the Task Log File before running a new release of SUPRA Server. Never format any Task Log File unless the system terminated normally on its last run.

If the PDM is forced down or terminates abnormally, the Task Log File contains recovery information. When you start the PDM again, task level recovery takes place automatically based on that information. If you do not want to recover, destroy the information by formatting the Task Log file.

Your OS/390 or VSE installation tape provides a sample JCL member for formatting the Task Log File. The sample TXJFTLOG points to its own input. The input includes statements that are similar to the following:

```
CONTROL ( BEGIN )
ENV-DESC ( CSTANONE )
SCHEMA ( CSTASCHM )
FUNCTION ( FORMAT )
FILE ( TLOG )
CONTROL ( END )
```

Do not change this input.

For more details on using the Format function, refer to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260.

Logical record types written in the task log file image blocks

The following logical record types are written to the Task Log File image blocks:

- ◆ **Record before images** contain the contents of the database records and information identifying the task and transaction. The record before images are used for backing out to the most recent commit point. Only the first before image for any record updated by a task during a particular transaction is logged. KSDS DEL-M images are an exception to this; they are logged regardless of previous activity to that key.
- ◆ **File mode before images** contain the file name and the open mode at the most recent commit. If you issue a CLOSX or OPENX command and change the open mode of the file, a file mode before image is written to the Task Log File. The file mode before images are also used for backing out to the most recent commit point. The file mode before images are used to complete opens and closes at the end of the transaction.
- ◆ **Commit records** are written when you issue a COMMIT or SINON command. When a commit record is written to the Task Log File, you can supply additional data to be written also. When you issue a RESET command, you get this information back. Commit records point to the beginning of the transaction during which a RESET command is issued. For a warm start, the valid commit records indicate which tasks are active when the system failed. (A SINOF is the end of a task and the task's last transaction. A commit record may be written for a SINOF command if you changed the open mode during processing of that transaction. In any case, the successful completion of SINOF implies no valid commit records still exist for that task.)
- ◆ **Index change log images** contain the secondary key just added, changed or deleted. Task Level Recovery (TLR) performs a logical recovery of these images as opposed to physically applying the secondary key images on the index file.
- ◆ **Index block log records** contain the contents of a physical block in the index file prior to structural maintenance occurring against the index file.

The Task Log File does not contain record after images or function log records.

Logical record types written in the task log file SFT blocks

The following logical record types are written to the Task Log File SFT blocks:

- ◆ **File description records** contain the current open status of the file, the owning task, and the physical description (device, buffer pool, physical fields, internal records, and secondary keys).
- ◆ **Environment description records** contain a copy of the environment description used to initialize the PDM.
- ◆ **Buffer pool records** contain information about a single buffer pool (number of direct buffers, etc.).
- ◆ **File group records** contain information about System Log File groups (which files and in what order) and options.

3

Creating PDM system log file groups

The installation tape contains a description of a System Log File in a Directory bootstrap schema. Use this description only in a non-Directory-driven or bootstrap mode. A System Log File can be a BDAM, ESDS or BSAM file. If this description does not meet your requirements, you must run Modify Schema to change any of the characteristics. See “[Choosing log group characteristics](#)” on page 43 for more information on changing the description of your System Log File(s).

For system logging in a Directory-driven mode, you must select system logging through Directory Maintenance (refer to the [SUPRA Server PDM Directory Online User's Guide \(OS/390 & VSE\)](#), P26-1260, or the [SUPRA Server PDM Directory Batch User's Guide \(OS/390 & VSE\)](#), P26-1261). Before you can use a System Log File, you must allocate it and then (for BDAM or ESDS only) format it using the Format function of the SUPRA DBA utilities.

The PDM writes the log records you select to the System Log File as required. For a description of these logical record types, see “[Logical record types written in the System Log File](#)” on page 53. Applications can put user specified data on the System Log File using the QMARK and MARKL commands. Refer to the [SUPRA Server PDM DML Programming Guide \(OS/390 & VSE\)](#), P26-4340, for the syntax of QMARK and MARKL.

You can use an additional log group to create a shadow log file. For more information on using additional log groups, see [“Recovering and restoring the SUPRA PDM from the System Log”](#) on page 83. While the main log file should be synchronized (to ensure recoverability), it is advantageous for the shadow file to be unsynchronized. This unsynchronization of the shadow file shortens wait time, I/O operations and shadow file space. Note that an unsynchronized System Log File is unsuitable for recovery.

If you want to log selectively (log certain information to one log group but not to another), you must use the logical write exit to do the selection. For example, if you want Group 1 to contain before images but not function records, and Group 2 function records but not before images, you must use a logical write exit. The exit based on log group name and logical record type decides whether to skip writing the given logical records to the given file.

The System Log File can span several tape reels when you log directly to tape; it spans several data sets when you use log groups. This spanning enables you to use only part of the file during system recovery. When you use only part of it, Log Print and Recover to last-commit can run more quickly. Dividing the System Log File into several parts requires an exit program using the PDM’s logical write exit to switch to the next data set. For more information on PDM exits, refer to the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250.

Choosing logging options

You specify five system logging options in your environment description: the four log record options and the end log option. These options have no effect on task logging.

The four log record options are represented by a four-character string in Directory Maintenance and other places. The choices for each character in this string are as follows:

1. First character: sign-ons (and sign-offs):

N log no sign-ons

U log only sign-ons for update tasks

A log all sign-ons

2. Second character: before-images:

N log no before-images

B log all before-images

3. Third character: physical DML functions (other than sign-ons and sign-offs):

N log no functions

U log only update functions

A log all functions

4. Fourth character: after-images:

N log no after-images

A log all after-images

If the log record options string is *NNNN*, no system logging takes place.

Considerations

- ◆ If you use a log record options value other than *NNNN*, you must have a system log file group defined in the active schema and related to a buffer pool in the active environment description.
- ◆ If you log before images or after images, you must also log update sign-ons.
- ◆ To be able to run the Recover function of the SUPRA DBA utilities to back off updates, you must log at least before images and update sign-ons.
- ◆ To be able to run the Restore function of the SUPRA DBA utilities to reapply updates, you must log at least after images and update sign-ons.



Function logging may be useful for analysis of database activity. Unless you plan such use, Cincom does not recommend using resources on this option.



For maximum recovery capability, Cincom recommends you use task logging and system logging together and use the system logging options ABNA.

The end log option, which you specify in the environment description, also affects system logging. This option tells the PDM what to do in each log group in response to an ENDLG physical DML instruction. Your choices are:

- C Close: close and reopen the current System Log File.
- F Force end-of-volume: switch to the next logical volume if possible; otherwise, ignore the instruction.

If you want to use the System Log to do any recovery, always specify end log option F.

Choosing log group characteristics

There are several possible configurations for log groups and the characteristics you can assign to them at the file, group, and buffer level. For information on defining log group characteristics on the Directory, see [“Defining the log groups with directory maintenance”](#) on page 47.

Multiple logical volumes

The System Log is also called a log group because it can consist of multiple logical volumes that can reside on one or more data sets. The multiple logical volumes feature enables you to break the System Log into portions of manageable size. The System Log can become very large because, unlike a Task Log File, the System Log is a chronologically sequential record of activity. Space in the log is never reused. The longer your PDM runs and performs transactions, the larger the System Log grows.

Without multiple logical volumes, the potentially large size of the System Log leads to two problems:

- ◆ If the System Log consists of a single file and the PDM uses all of the space available on that file, the PDM terminates abnormally.
- ◆ The recovery functions of the SUPRA DBA utilities must read the entire log as input, even though only part of the log may be relevant to the function's purpose.

When you use the multiple logical volumes feature, the System Log is divided into logical volumes as the PDM writes it. Each logical volume starts with information that:

- ◆ Identifies the log, volume, and PDM
- ◆ Describes the environment and each file
- ◆ Gives the current status of each file and task

This means the recovery functions that use the System Log as input do not necessarily need to read the entire log. They can start at the beginning of any logical volume. For more information on System Log Files, see [“Switching system log files”](#) on page 87.

File attributes

Considerations for choosing the characteristics of the file(s) in a log group include the following:

- ◆ You can specify BSAM, BDAM, ESDS, OUTP, and WORK access methods for a System Log File. (OUTP and WORK are for VSE tape devices only.) The RECOVER/RESTORE/LOG PRINT utility accesses a disk System Log File directly regardless of how the SUPRA PDM created the file. For an ESDS file, the utility uses ESDS. For a BSAM or BDAM file, the utility uses BDAM.



Cincom recommends a block size of at least 3000 bytes. The minimum block sizes are 505 bytes for ESDS VSAM and 512 bytes for the other access methods. The PDM performs best if the block size is large enough to hold the largest logical records in your database.

- ◆ All the files in a log group must have the same block size.
- ◆ For an ESDS VSAM file, the block size should be seven bytes less than the control interval size. The control interval size must be 512, 1024, or a multiple of 2048.
- ◆ For VSE FBA users, BDAM is not valid.

Group attributes

At the log group level, set the volume-maximum-RBN for each file in the group and set the wrap and synchronization options.

The volume-maximum-RBN is the highest relative block number to be written to a logical volume before switching to the next logical volume.



Cincom recommends you set this value to about five percent less than the estimated capacity of the disk file or tape reel. Do not set this value (leave it at 0) for an ESDS VSAM file or a BDAM file. The volume-maximum-RBN is then calculated automatically as the highest RBN in the file.

The synchronization option ensures that the before image of each database update is logged before the database is updated and that all data for a transaction is logged when the transaction commits. You must set this option to YES to run the Recover function successfully against this log group. If you set this option to NO, you can save processing time and log file space and still run the Restore function. However, if you want to guarantee that the Restore function restores all updates of the last transactions completed before termination, you must set this option to YES to force the necessary data onto the log at every commit point.



Cincom recommends the use of the wrap option to reuse files in the log group. Be aware that this option is valid only for groups with two to four files all on disk.

Buffers



Cincom recommends 2–5 buffers per log group. The minimum of one buffer per log group degrades performance. (The number of files in a log group is irrelevant to the number of buffers needed.) System Log Files must not share a buffer pool with data files.

Defining system logging in the bootstrap modules

Cincom suggests that you not define system logging in the bootstrap modules for two reasons:

- ◆ The definitions have effect only if you run non-Directory-driven with system logging. That configuration has minimal usefulness. When you run Directory-driven, the PDM ignores the bootstrap definition of system logging in favor of whatever is on the Directory.
- ◆ When you define system logging on the bootstrap, you cannot specify multiple logical volumes, multiple data sets, the logical write exit, or the new volume exit.

For information on how to define a System Log and its options in the bootstrap modules using the bootstrap utility programs, refer to the *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250.

Defining the log groups with directory maintenance

Your Directory environment must contain appropriate definitions to run with system logging. In order to run Directory-driven with system logging, the following must be true:

- ◆ Your Directory schema must contain definition(s) for the System Log File(s).
- ◆ Your Directory environment description must define log group(s) containing the System Log File(s).
- ◆ Your Directory environment description must define buffer pool(s) and relate the System Log File(s) to them.
- ◆ Your Directory environment description must specify system-logging options (other than NNNN) and a suitable end log option.

When you have a Directory schema you want to use with system logging, add each System Log File to the schema as follows:

1. At the Directory Maintenance category menu, enter FI to select the file category.
2. At the appropriate prompts on the file command menu, enter AD to select the ADD command and enter the schema name and System Log File name.
3. At the first file add screen, enter the ddname, file type as SYSTLOG, access method, file device type, logical record length (which in this case means block size), and total logical records (for an ESDS or BDAM file only). You need not access the other two file add screens.
4. Return to the Directory Maintenance category menu.

When you have a Directory environment description you want to use with system logging, do the following:

1. Specify the appropriate environment description options:
 - a. At the Directory Maintenance category menu, enter ED to select the environment description category.
 - b. At the appropriate prompts on the environment description command menu, enter CG to select the CHANGE command and enter the schema name and environment description name.
 - c. At the first environment description change screen, enter the log record options. Enter the end log option as F.
 - d. At the second environment description change screen, enter the logical write exit and the new volume exit (if applicable).
 - e. Return to the Directory Maintenance category menu.
2. Add a log group for the System Log File(s):
 - a. At the Directory Maintenance category menu, enter LG to select the log group category.
 - b. At the appropriate prompts on the buffer pool command menu, enter AD to select the ADD command and enter the schema name, environment description name, and eight-character name you choose for the new log group.
 - c. At the appropriate prompts on the log group add screen, enter the wrap option, synchronization option, System Log File name(s), and volume-maximum-RBN for each file.
 - d. Return to the Directory Maintenance category menu.

3. Add a buffer pool for the System Log File(s) (note that files in a log group can and should share a buffer pool):
 - a. At the Directory Maintenance category menu, enter BP to select the buffer pool category.
 - b. At the appropriate prompts on the buffer pool command menu, enter AD to select the ADD command and enter the schema name, environment description name, and four-character name you choose for the new buffer pool.
 - c. At the buffer pool add screen, enter the direct buffer count, serial buffer count, and serial thread count as 2, 0, and 0, respectively.
 - d. Return to the Directory Maintenance category menu.
4. Relate the System Log File(s) to the new buffer pool:
 - a. At the Directory Maintenance category menu, enter FI to select the file category.
 - b. At the appropriate prompts in the file command menu, enter RC to select the RELATIONSHIP CHANGE command and enter the schema name and System Log File name.
 - c. At the appropriate prompts in the file relate change screen, enter the environment description name, an open mode of SUPD, and the name of the new buffer pool.
 - d. Return to the Directory Maintenance category menu.

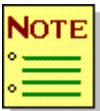
Finally, after you use Directory Maintenance to update a schema, you must consistency check the schema to make it and its environment descriptions usable:

1. At the Directory Maintenance category menu, enter SC to select the schema category.
2. At the schema command menu, enter CK to select the CHECK command and enter the schema name.
3. At the schema check screen, specify Y (yes) for INCON PHYSICAL ENTITIES and ALL LOGICAL ENTITIES. Specify N (no) for the remaining prompts. Allow time for the function to complete.
4. Return to the Directory Maintenance category menu.

You may wish to use File Level Log Suppression to suppress logging for individual MANTIS and SPECTRA files. For more information on this procedure, see “[Suppressing task logging](#)” on page 35.

For more details on using Directory Maintenance, refer to the [SUPRA Server PDM Directory Online User's Guide \(OS/390 & VSE\)](#), P26-1260, or the [SUPRA Server PDM Directory Batch User's Guide \(OS/390 & VSE\)](#), P26-1261.

Placing the disk log files



Log files are the most frequently accessed files in the PDM. For the most efficient use of your direct access devices, Cincom recommends that disk System Log Files reside on a device separate from your most frequently accessed user files and from the Task Log File. This method also protects you against loss of both user and log files from a single device failure.

Initializing system log files

You must format each direct access System Log File (a disk file that uses BDAM or ESDS VSAM access method) after its creation and before its first use. In OS/390, you must initialize a tape before using it as part of a System Log. You need not format System Log Files that reside on tape devices. You need not format BSAM System Log Files, whether on disk or on tape.

Use the Format function of the SUPRA DBA utilities to format each direct access System Log File:

For OS/390

1. Create a new JCL member by copying the sample JCL member for formatting the Task Log File, TXJFTLOG.
2. Change the CSIPARM member name reference in the JCL from TXPBEND to TXPBNONE.
3. Create a new UCLCODE member. Change the UCLCODE member name reference in the JCL from TXUFTLOG to the name of the new member.
4. Put the following statements in your new UCLCODE member. Substitute the name(s) of your Directory schema and environment description (which must define the file(s) you are formatting) and the name(s) of the file(s):

```
CONTROL(BEGIN)
      ENV-DESC(user-schema)
      SCHEMA(user-envdesc)
FUNCTION(FORMAT)
      FILE(slog-1)
      FILE(slog-2)
CONTROL(END)
```

5. Run the JCL.

For VSE

1. Create a new JCL member by copying the sample JCL member for formatting the Task Log File, TXJFTLOG.

2. Change the CSIPARM input in the JCL from END. to:

```
DIRECTORY=( SCHEMA=CSTASCHM, ENVDESC=CSTANONE ), END.
```

3. Change the UCLCODE input in the JCL from:

```
CONTROL(BEGIN)
    ENV-DESC(CSTANONE)
    SCHEMA(CSTASCHM)
FUNCTION(FORMAT)
    FILE(TLOG)
CONTROL(END)
```

to the following. Substitute the name(s) of your Directory schema and environment description (which must define the file(s) you are formatting) and the name(s) of the file(s):

```
CONTROL(BEGIN)
    ENV-DESC(user-schema)
    SCHEMA(user-envdesc)
FUNCTION(FORMAT)
    FILE(slog-1)
    FILE(slog-2)
CONTROL(END)
```

4. Run the JCL.

For more details on using the Format function, refer to the [SUPRA Server PDM DBA Utilities User's Guide \(OS/390 & VSE\)](#), P26-6260.

Logical record types written in the System Log File

The following logical record types are written to the System Log File:

- ◆ **PDM initialization and termination records** indicate the start and end of the System Log File. A PDM initialization record is written to the System Log File when the PDM is initialized, when files within a System Log File Group are changed, or when a logical volume change occurs on a tape System Log File. A PDM termination record is written when you issue an unforced ENDTO (in a multitask operating mode) or a SINOF command (in a single-task operating mode).

An initialization record contains the current logical volume number and the number of the most recently committed volume (the earliest volume that might need to be included in a recover-to-last-commit operation). Every logical volume begins with an initialization record, file-initialization records for each defined file, and task records to each active task.

- ◆ **File initialization records** describe the physical characteristics of a file, along with that file's initial open mode when the System Log File was started. The file initialization record on the System Log File is similar to the System File Table entry on the Task Log File.
- ◆ **Task records** indicate the presence of an active task, the task's COMMIT ID, name, and whether it has uncommitted updates on this System Log File.
- ◆ **Command or function records** for any DML command are generated at the successful completion of a specified DML command. The command record contains the interface name, the task name, and command information. If the DML command is an I/O command, the record which is added, deleted, read or written is also contained in the command record. If the command is not an I/O command, the data written to the System Log File varies. For example, for an OPENX command, the OPENX options are written. Command records (SINON, SINOF, QUIET, QMARK, COMIT, RESET) are used to recover the database to the last commit point. Refer to the [SUPRA Server PDM Directory Online User's Guide \(OS/390 & VSE\)](#), P26-1260, or the [SUPRA Server PDM Directory Batch User's Guide \(OS/390 & VSE\)](#), P26-1261, for information on specifying function logging.

- ◆ **Image records** include before or after images for all updates. They also contain task identification information. The number of before images written to the System Log File depends on whether task logging is active. If task logging is active, one before image is written to the System Log File for a given physical record for a given transaction. If task logging is not active, one before image is written to the System Log File for each data record updated. If you specified before or after image options, a before and after image is logged for every KSDS file add or delete.

With index images and KSDS image records, the System Log File block is flushed immediately (rather than delayed, as for BDAM and ESDS). Therefore, the block on the System Log File may not be fully utilized. Heavy updates of KSDS files could also affect throughput.

- ◆ **Index change log images** containing the secondary key just added, changed or deleted. Task Level Recovery (TLR) performs a logical recovery of these images, as opposed to physically applying the secondary key images on the index file.
- ◆ **Index block log records** containing the contents of a physical block in the index file prior to structural maintenance occurring against the index file.
- ◆ **File mode before image records** are written if task logging is also active (for an OPEN or CLOSE DML command).
- ◆ **System event records** are for future use.
- ◆ **An end of volume record**, if it appears, is the last record in a logical volume. It is used only on a direct access method (BDAM or ESDS) System Log File when the log is going to switch to another data set and there are unused blocks in the current data set.

One PDM initialization record, one PDM termination record (on a normal termination), and one or more file initialization records are written to the System Log File. These records are written for each file defined to the PDM in the bootstrap Directory schema, or in the user schema, or in both (except for the System Log File, the Task Log File, and the Statistics File).

4

Running SUPRA with logging

To take advantage of running SUPRA with logging active, there are procedures to follow before, during, and after running logging.

Recovering files

When you create or modify an application program to use the recover or restart features of the PDM, consider the update conditions for the application's files after a failure. The following list shows possible conditions and how to obtain them:

- ◆ **Updates are backed out to each task's last commit point.** When a task fails, all of its updates since the last commit are undone and the effects of the incomplete transaction are removed from the database. In a task logging environment, this is the only possible outcome, unless there is physical damage to a device containing the Task Log File or one of the updated files. (If there is physical damage to updated files, no damage to the Task Log, and system logging of after images was performed, the affected file(s) can be reloaded from tape backup and restored using the Restore function of the SUPRA DBA utilities. Task Level Recovery proceeds when the PDM is restarted.) The backout (or reset) to commit occurs either at the time of failure or when the PDM is restarted with the same Task Log File and options.
- ◆ **Updates are backed out to the PDM's last quiet point.** Use the Recover function of the SUPRA DBA utilities to last-commit. Quiesces are not supported in a task logging environment, and commits are not supported in a non-task-logging environment.
- ◆ **Contents of update files have been backed out to the point of failures.** Use this option only if the file(s) has been physically damaged and reloaded from backup. For this condition, use the Restore function of the SUPRA DBA utilities. The consistency of affected files, internally or with each other, cannot be assured.
- ◆ **Updates are not important after a failure.** However, the affected files must be unlocked and be consistent internally and with each other. This option usually applies to scratch files, work files, and slide files. Generally, these files can be made consistent by formatting the file, reloading the file, or using one of the first three options above.

Considerations

- ◆ The recover and restore utilities do not recover or restore format functions.
- ◆ If a format is in progress when a failure occurs, you cannot recover the file being formatted.
- ◆ If a format has completed, you can recover the file to the last commit. However, you cannot restore the file or recover to log begin.

Backing up data files

Before starting to run the PDM with logging, back up all important files. You must identify which of your files contain valuable data. You must periodically make backups of these files. Logging is not a substitute for backups.

Scratch files and work files contain no valuable data and need not be backed up or logged.

Committing tasks

Issue commit instructions in your application. Commit points include the start (sign-on) of a task and each point when the task issues a commit instruction. The application should issue commit instructions periodically, but not in the middle of a logical unit of work (that is, not in the midst of a series of updates that are logically closely interrelated). Issuing a commit instruction makes permanent all changes made by the task up to that point.

Saving special data on log files

If you migrated to SUPRA Server from TOTAL, you can use the following physical DML instructions to save user-specified data on PDM log files. If you are using an RDM application, the RDM will perform this function automatically.

- ◆ COMIT can save user data in a commit record on the PDM Task Log File. RESET retrieves this data from the most recent commit. An application can use this feature to backout an incomplete logical unit of work and resume processing. In the event of a task or system failure, work can be resumed at the point of last commit when the task restarts. In order to take advantage of this feature, the application must be made restartable by specifying RESTART=NORMAL in the CSIPARM file for the interface. Cincom's batch Directory Maintenance is such an application.
- ◆ MARKL and QMARK save user data in special records on the PDM System Log. Such a record can serve as a marker indicating the point when a significant event occurred (a file backup, for example). The PDM logical write exit could force a new logical volume at this point. The recovery exit could selectively apply images depending on whether they preceded or followed the marker record.

For more information about these instructions, refer to the *SUPRA Server PDM DML Programming Guide (OS/390 & VSE)*, P26-4340.

Starting a new logical volume on the PDM system log

While running the PDM, switch logical System Log volumes at appropriate times. The PDM completes one logical volume and starts writing to the next when one of the following happens:

- ◆ The relative block number on the current volume reaches a user-specified limit (VOL-MAX-RBN, specified on the Directory).
- ◆ The logical-write user exit returns an action indicator of NVOL.
- ◆ An application issues the physical DML instruction ENDLG (and the end log option is force-end-of-volume).

Resetting tasks

While running the PDM, your applications should issue reset instructions when appropriate. A reset backs off all updates the task made since the last commit. This ensures that all data and control information affected by the changes are restored to a state where they are internally consistent. If a problem is detected in a transaction or the transaction cannot successfully complete, a reset is appropriate.

SUPRA PDM and CICS Connector automatically back off the changes of some or all of their tasks to the last commit in response to a forced disconnect command, forced termination command, abnormal termination, or restart following an abnormal termination.

Purging dangling tasks

When a task known to the PDM terminates abnormally (without signing off) and the PDM is running with Task Level Recovery, the PDM normally treats the task as restartable. The PDM maintains information about the task and holds resources for the task in expectation of restarting the task. The PDM cannot be shut down unforced under these circumstances. If the PDM is forced down, it will retain information about the task in the Task Log File.

If you do not intend to restart tasks, you can specify `RESTART=NONE` in the CSIPARM file for the PDM. This prevents the PDM from holding resources for failed tasks. For information about the `RESTART=` parameter, refer to the *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250.

If your CSIPARM specifies (or defaults to) `RESTART=NORMAL`, but you have a failed task you will never restart, you should purge the task from the PDM. Interactive Services can be used to purge tasks for interfaces currently not connected to the PDM:

1. At the Interactive Services main menu, specify PDM services.
2. At the PDM Services menu, specify Task Management.
3. At the Task Management screen, leave the default values in place and press ENTER. A list of interfaces appears.
4. Select the interface to which the task belongs. A list of tasks appears. Follow the directions on the screen to purge the task.

For more details on Interactive Services, refer to the *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250.

Restarting tasks

In order to restart a task, it must be signed on again to the PDM. If the restart is successful, the PDM returns a status of RSTR. If you are using an RDM application, the RDM will perform this function automatically.

Warm starting the PDM

The PDM warm starts when you resubmit a job. When Task Level Recovery is active at the time of a system failure, the PDM automatically warm starts if the environment description, schema, PDM name, or job name (if the PDM name defaults to the job name) have not been changed.

You cannot warm start a central mode PDM under OS/390 in another step of the same job that brought it up. You must warm start it as a separate job.

Recovering from PDM task and system failure

Two types of uncontrolled termination or failure can occur during processing: a task failure or a system failure. This section explains how to recover and restore your database in case of failure.

Task failure

A task failure occurs when a task or its interface fails. A task may fail in either of the following two environments:

- ◆ **Task logging is active.** When the PDM is running under Task Level Recovery (TLR) and a batch task failure occurs, the PDM can recover the database files to the last commit point. If a CICS task fails, the CICS backout exit program issues a RESET and SINOF for the task.

If CICS or the CICS Connector fails, the PDM recovers the database files for all of the tasks signed on to the interface. Resources necessary to restart a task are kept by the PDM for restartable tasks. All resources are discarded for non-restartable tasks.

- ◆ **Task logging is not active.** When the PDM is running with no task logging (non TLR), the PDM cannot recover database files for the failing task or interface. All PDM resources controlled by the task and interface are freed. The PDM continues processing, and the database may be corrupt due to work not completed by the failed task.

If Task Level Recovery is not enabled, the PDM releases only files that were opened EUPD. You are responsible for logical integrity of your database. System logging is recommended in this environment.

System failure

For purposes of logging and recovery, a system failure may occur when anything fails that causes the PDM to fail. The following table outlines the symptoms, causes, and procedures you should follow to correct failures under several uncontrolled terminations.

In a system failure, the PDM dumps additional information along with the normal system dump. This information is in the form of snap dumps. Snap dumps are selective dynamic dumps.

For OS/390 and VSE, the PDM issues snap dump requests in various places in its abend processing. Each snap has a snap title and snapped data. In VSE, which does not support snap titles, the title is snapped before the snapping of the data. This results in two snaps for each snap point, instead of one as for OS/390.

OS/390 In OS/390 the snapped data is output to a PDMSNAP data set. You must include a PDMSNAP DD card in the JCL for the PDM job step.

VSE In VSE, the PDUMP Facility does not control the destination of the snapped data. It is determined by VSE.

When recovering a system with task logging, warm start the PDM with the same environment description (that is, use the same CSIPARM file with its contents unchanged). Also, use the same Task Log File so you can restore your environment. (In OS/390, if the PDM is in central or attached central operating mode, you cannot run two PDMs in the same job.)



If one or more of your KSDS VSAM files was opened for update at the time of a system failure, do not follow the steps outlined in the following table; see [“Recovering KSDS VSAM files”](#) on page 67.

Symptom	Cause of failure	Procedure to correct failure
Status of IOER— no abend	Device Failure and the PDM continues processing (reading primary/related files). The IOER was not on a physical database update, so the PDM can continue because integrity of the database is not compromised.	<ol style="list-style-type: none"> a. Restore the affected database files from the system backups. b. Run the RESTORE function of the SUPRA DBA utilities against the affected database files to the desired point (last commit or log end) for all Systems Logs since the last system backup.
Status of IOER followed by U998	Device failure where the PDM terminates if writing or reading the Task Log File.	<p>Attempt to warm start PDM.</p> <ol style="list-style-type: none"> a. If successful warm start: terminate the PDM; fix the Task Log File. b. If unsuccessful warm start: run the RECOVER function of the SUPRA DBA utilities to the last commit. Format the Task Log File; cold start the PDM.
Status of IOER followed by U998 abend	Device failure and the PDM terminates. The IOER was on a physical database update. The PDM cannot continue because database integrity would be compromised.	<p>TLR:</p> <ol style="list-style-type: none"> a. Restore the pack or affected database files from the system backups. b. Run the RESTORE function of the SUPRA DBA utilities against the affected database files to last commit. Run the RECOVER function for all other files. Format the Task Log File. c. Cold start the PDM. <p>NON- TLR:</p> <ol style="list-style-type: none"> a. Restore the pack or affected database files from system backups. b. Run the RESTORE function of the SUPRA DBA utilities against the affected database files to log end. c. Run the RECOVER function of the SUPRA DBA utilities to last quiet on all files. d. Cold start the PDM.

Symptom	Cause of failure	Procedure to correct failure
Status of IOER followed by U998	Device failure where the PDM terminates if writing to the Statistics File.	<p>TLR:</p> <ol style="list-style-type: none"> Resolve the IOER to the statistics File Warm Start the PDM. <p>NON-TLR:</p> <ol style="list-style-type: none"> Run the RECOVER function of the SUPRA DBA utilities to the last quiet. Fix the Statistics File problem. Start the PDM.
Status of IOER followed by U998	Device failure where the PDM terminates if writing to the System Log.	<p>TLR:</p> <p>If you have a subsequent failure, determine whether you will need the System Log.</p> <ol style="list-style-type: none"> If you need the System Log for restore capabilities, determine if it is intact: If it is intact, warm start the PDM. If it is not intact, warm start the PDM, terminate the PDM and run the system backups. If you do not need the System Log for restore capabilities, warm start the PDM. <p>NON-TLR:</p> <p>Determine if the System Log can be used for recovery:</p> <ol style="list-style-type: none"> If it can be used, run the RECOVER function of the SUPRA DBA utilities to the last quiet; start the PDM. If it cannot be used, restore all database files from the system backups; run the RESTORE function of the SUPRA DBA utilities one or more times for all System Logs since the last backup; start the PDM and redo all transactions from this PDM execution.
Power failure, CPU failure, PDM software failure	Nothing affecting the PDM's recovery is damaged (Task Log File, System Log).	<p>TLR:</p> <p>Warm start the PDM.</p> <p>NON-TLR:</p> <ol style="list-style-type: none"> Run the RECOVER function of the SUPRA DBA utilities to the last quiet. Start the PDM.

Defining special cases and problems in recovery

There are circumstances where, because of the type of activity on the PDM at the time of failure, the standard recovery procedures are inapplicable or insufficient.

Considerations:

- ◆ If a failure occurs after a Format function and a Recover or Restore function is required, unpredictable results occur unless you simply recover the file to the last commit.
- ◆ If a failure occurs during a Format function, you must reformat the file since its current state is unpredictable.

Recovering KSDS VSAM files

KSDS VSAM database files present a special recovery problem. When you attempt to add a record to a KSDS VSAM file and the control interval in which the new record belongs is full, VSAM splits the control interval. VSAM divides the records in the full control interval between two control intervals. VSAM updates the index and then adds the records.

If VSAM is aborted between the time the control interval is split and the time the index is updated, the resulting KSDS VSAM file is inconsistent internally and cannot be fixed.

To aid the Recover function in determining whether the VSAM write has been completed, the PDM automatically writes both before and after images to the System Log File.

The Recover function matches all before and after combinations. If a before image is not matched with an after image, the Recover function produces a message indicating an error may have occurred during a control interval split. Therefore, if the PDM fails for any reason and you had been adding records to KSDS VSAM files at the time of the failure, do the following:

1. Do not warm start the PDM.
2. Run the Recover function of the SUPRA DBA utilities. If a control interval split may have been in progress at the time of failure, the Recover function produces a message to that effect. If a control interval split was not in progress at the time of failure, you are finished.
3. If the Recover function detects possible control interval splits, reload the previous backup of the affected KSDS VSAM files. Then execute the Restore function to restore the affected KSDS VSAM file(s) from the system backups to the point to which the rest of the database was recovered (last commit, last quiet, or log end).

Secondary key populate or depopulate function considerations

When running the Recover function against a System Log to recover a PDM that was performing a Populate or Depopulate function at the time of failure, execute the Recover function to last commit and normal recovery proceeds. If the Populate or Depopulate function is unfinished, you should depopulate and repopulate after recovery if necessary.

If the PDM fails during a populate or depopulate function, the index file's free chain may be corrupt. To be sure the chain is intact, reformat the index file and populate all secondary keys related to that index file.

If you did not execute the Recover function to last commit, depopulate the secondary key and repopulate after recovery if necessary.

If a system failure occurs during a Populate or Depopulate function, any secondary key undergoing utility maintenance at the time of failure must be depopulated and purged. After purging, populate the secondary key again. You may need to reformat the index file to recover parts of the free space chain. Warm Start or Recovery Failure on Secondary Keys.

Warm start or recovery failure on secondary keys

Failure during a warm start or recovery has unpredictable effects on secondary keys. If a failure occurs during a warm start or recovery while doing an index split:

- ◆ The index file free-space chain may be damaged.
- ◆ You may need to format the index file, or some blocks may be inaccessible permanently.
- ◆ The secondary keys may be damaged and unrecoverable. In this case a depopulate and repopulate does not work. You must reformat the index file.

Rerunning a warm start or recovery may cause data file index file synchronization errors on the secondary keys. If this occurs:

- ◆ The index file freespace chain is undamaged.
- ◆ The secondary keys are structurally intact.
- ◆ If you did not perform any maintenance on the base files, the secondary keys are logically intact.

Processing isolation with single-task PDM

When recovering a single-task PDM, a sign-on with ACCESS=RECOVER results in processing isolation of base and index files. This means that:

- ◆ Opening the base file does not automatically open the index file.
- ◆ You can open the index file.
- ◆ Changes to index files are not automatically reflected in base files, and changes to base files are not automatically reflected in index files.
- ◆ If you issue a READX, unpredictable results occur.

Recovering a file during a format function

When recovering a file during a Format function, follow these considerations:

- ◆ If the file did not exist before the Format function:
 - Execute the Recover function to log beginning.
 - The system is as it was prior to the existence of that file. However, the file contains invalid data. Either delete or reformat the file.
- ◆ If the file existed before the Format function but did not contain data:
 - Recover to log beginning.
 - The file contains invalid data.
 - You must create and format the file before executing the Restore function to log end.
- ◆ If the file existed before the Format function and contained data, do not restore to log end or last commit.

Multiple PDM recovery

If you update file(s) using multiple PDMs serially and execute the Recover or Restore function, the file can be seriously damaged. If you execute the Recover function to last commit, no damage occurs. If you execute the Recover function to log begin or perform any type of Restore function, results are unpredictable since you must get the data from multiple PDM log files.

If a PDM updates and subsequently closes a file, and another PDM accesses and updates the same file, the System Log Files for both PDMs together make up a logical System Log File. You must run two recovers and restores (one for each file) in the correct order.

If the situation just described occurs, and the first PDM then reopens and updates the file after the second PDM has closed the file, the System Log Files (if used) can corrupt the database. Depending on the sequence of events, the recovery process can corrupt the database if proper procedures are not established. Each user must examine the overall system and develop recovery procedures which use Cincom-supplied recovery techniques to ensure database integrity.

5

Recovering SUPRA CICS applications

The recovery facilities of CICS and the SUPRA PDM are used to recover SUPRA CICS applications. The CICS and PDM facilities should be synchronized for optimum operation. Recovery can be made from task and system abends for RDM, SPECTRA, MANTIS, and Directory Maintenance



CSTXOPRM options SYNC=C or SYNC=U allow the user to gain some control over PDM and CICS synchronization. For details on these options, refer to the *SUPRA Server PDM CICS Connector Programming Guide (OS/390 & VSE)*, P26-7452

Preparing CICS and SUPRA for recovery

The PDM requires task logging. The SPECTRA Personal File System data set must be a protected CICS resource or PDM file to be guaranteed recoverable.

Recovering CICS application programs

Recovery of CICS applications requires an understanding of the CICS connector component of SUPRA Server, the available exit points and supplied exit routines, and the operator control commands for the connector. For details on these items, refer to the *SUPRA Server PDM CICS Connector Programming Guide (OS/390 & VSE)*, P26-7452.

Briefly, for the CICS connector to function, you must perform the following steps. Once they are done, these facilities are available for recovery purposes.

- ◆ Define any native VSAM files (non-PDM VSAM files) or DL/I databases you access through CICS as CICS-protected (recoverable) resources. These can include native KSDS VSAM files or DL/I databases you access through RDM or RRDS VSAM Personal File Data for SPECTRA. CICS then recovers these files when a CICS task abend occurs.

The Cincom provided Task Related User Exit (TRUE) issues a RESET DML instruction to the PDM when a CICS task abend occurs, so that any PDM updates by that task are rolled back to the last commit point. For more information on CICS exits and the CSTXOPRM macro, refer to the *SUPRA Server PDM CICS Connector Systems Programming Guide (OS/390 & VSE)*, P26-7452.



Cincom recommends that you use PDM task logging for any SUPRA PDM that performs updates, regardless of whether the update requests come through CICS.

If the SPECTRA Personal File System (PFS) data set is a PDM file, it can be recovered by the PDM after a task or system abend. If the PFS data set is an RRDS VSAM file and is defined as a CICS-protected resource, it can be recovered by CICS after a task abend. If the PFS data set is not a PDM or RRDS VSAM file, its recovery after an abend depends on the timing of the failure; if the failure occurs during PFS processing, the PFS data set may contain uncommitted updates.

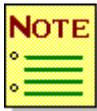
For recovery purposes, all CICS application programs:

- ◆ Must synchronize CICS sync points/rollbacks and PDM commits/resets. PDM and CICS use different recovery systems, and these systems are not synchronized. If applications depend on more than one recovery mechanism in a single logical unit of work, recovery cannot be guaranteed, because the commits to the different recovery systems are not synchronized. (For instructions on using the COMMIT command, refer to the *SUPRA Server PDM RDM Administration Guide (OS/390 & VSE)*, P26-8220.)

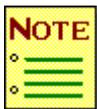
The synchronization can be done by the programmer in each program's logic. One method for handling the PDM commits and resets is to use a DBA-written exit (DATBASXT) in the DATBASC linked with each program.

Another method is to let the CICS connector exit logic handle the PDM side of synchronization. (Refer to the *SUPRA Server PDM CICS Connector Systems Programming Guide (OS/390 & VSE)*, P26-7452, for information on CICS Connector exits and the CSTXOPRM macro.)

- ◆ Must check for synchronization during recovery of PDM and CICS resources after a failure. The program can compare the sync point data retrieved from a user CICS journal with the commit data from the PDM. If the CSIPARM Option RESTART is coded as RESTART=NORMAL the task receives a RSTR status from the PDM upon re-SINON, and can retrieve the saved commit data with the RESET command. If the sync point data does not match the commit data, the failure occurred during the time between the CICS sync point and the PDM commit point. This is termed the recovery window, and means that the two are not synchronized.



To use RESTART=NORMAL every CICS transaction must be written to handle recovery, and must attempt to re-SINON after a failure. If this is not the case you should use RESTART=NONE in your CICS CSIPARM.



You can synchronize the two and eliminate the recovery window only by controlling your own COMMIT/RESET.

How CICS and the PDM recover

When a task abends, the following occurs:

- ◆ CICS recovers CICS-recoverable resources to the most recent sync point.
- ◆ The PDM issues a RESET and SINOF on behalf of the abending task.

When CICS or the PDM abends, the following occurs:

- ◆ CICS recovers to the most recent sync point the resources you defined as recoverable for each active task.
- ◆ The PDM recovers its resources to the most recent commit point for each signed-on task. You define a PDM commit point in one of two ways:
 - Explicitly by a COMIT command (if task logging is active)
 - Implicitly by a SINON or SINOF command.

When the operations staff subsequently reconnects the CICS connector to the PDM after a CICS or PDM abend, the following occurs:

If the CSIPARM RESTART option = NORMAL:

- Tasks which are signed on but have not committed are recovered to the point of sign-on. Then they are signed off. All other tasks remain signed on to the PDM. However, a CICS Connector STATUS operator control command will not show these tasks in its displayed count fields.
- For any tasks which have not issued COMIT commands, the PDM returns a **** status to any SINON commands issued after the reconnect.
- For any tasks which have issued COMIT commands, the PDM returns an RSTR (restart) status to all SINON commands issued after the reconnect. The RSTR status indicates that the SINON command has completed correctly and that the database is restarting the task from the Task Log File. If you specified commit data on the COMIT command, issue a RESET command to retrieve that commit data. Your application may require special processing to handle the RSTR status.

If the CSIPARM RESTART option = NONE:

- Tasks which are signed on are recovered to the point of last commit. Then they are signed off.

Recovering after a CICS task abend

After a task abend, CICS provides automatic recovery for the CICS resources you define as recoverable. To recover after a CICS task abends, reissue your transaction.

Recovering RDML applications

If a Relational Data Manipulation Language (RDML) task abends, the RDML interface recovers PDM resources to the most recent RDML commit point. CICS rolls back CICS protected resources.

If C\$VOOPTM SYNCTYP=Y is coded:

- ◆ If you issue an RDML RESET instruction, it requests a CICS ROLLBACK and a PDM RESET request.
- ◆ An RDML COMMIT or sign-off instruction also issues a PDM COMMIT and a CICS sync point. The CICS sync point commits any non-PDM resources that are CICS protected. This is especially important if you use RDM to update native VSAM files or DL/I databases.



C\$VOOPTM SYNCTYP=Y and CSTXOPRM SYNC=C address the same problems. If both are coded as above you will duplicate the PDM/CICS comits and syncpoints. (Refer to the *SUPRA Server PDM CICS Connector Systems Programming Guide (OS/390 & VSE)*, P26-7452 and the *SUPRA Server PDM RDM Administration Guide (OS/390 & VSE)*, P26-8220)

Every application must issue a sign-off instruction when it is terminating from CICS. Pseudoconversational tasks doing a pseudo-conversational return to CICS do not have to sign-off at that point. They should issue a commit instruction instead. Both pseudo-conversational and conversational applications must issue a sign-off on their final return to CICS. Failure to issue a sign-off leaves the task as an active task in the PDM and needlessly ties up PDM resources.

Recovering MANTIS applications

If a MANTIS application task fails, the CICS Exit Program backs out any CICS and PDM updates performed during the incomplete logical unit of work and displays an error message. The MANTIS task terminates. You must sign back on to the MANTIS system.

VSAM file or DL/I database updates are reset to the last sync point if the files have been defined as recoverable to CICS.

For more information on MANTIS recovery, refer to your MANTIS documentation.

Recovering directory maintenance

If an abend occurs while you are running Directory Maintenance transactions and there are no uncommitted updates, CICS returns an abend message and the CICS connector returns the message “Backout Successful.” To reset, you must reissue the Directory Maintenance transaction to bring Directory Maintenance back up. You receive a standard sign-on screen, and the files are restored to the most recent commit point.

If an abend occurs while you are running Directory Maintenance transactions and there are uncommitted updates, CICS returns an abend message and the CICS connector returns the message “Backout Unsuccessful.” A Directory Maintenance Restart screen appears.

If a terminal READ or WRITE failure occurs, there is no message because the terminal is out of service. To determine the cause of the failure, review the CICS Dump Data Set and the CICS Connector Activity Audit Trail. Check whether reset and sign-off instructions were issued and check for reset failure. Refer to the IBM documentation for information on how to print the dump data set. To recover from a READ or WRITE failure, process as above for the messages “Backout Successful” and “Backout Unsuccessful.”

Recovering after a CICS system abend

To recover after CICS abends, you must do the following:

- ◆ Emergency restart CICS.
- ◆ Reconnect the CICS connector to the PDM in one of these ways:
 - Automatically, if you installed the automatic connect option at CICS initialization.
 - Manually, by issuing the CONNECT operator control command.

CICS resources are recovered to the most recent sync point of each task. The PDM automatically recovers SUPRA Server resources when it detects that CICS abended. If the PDM is operating in the attached operating mode, the PDM recovers when it is restarted. PDM resources are recovered to the most recent commit point for each signed-on task.

After you emergency restart CICS, reconnect the CICS connector to the PDM. Depending on the options you installed at CICS initialization, this can be done automatically. You can also use the CONNECT operator control command.

If you emergency restart CICS and there are no uncommitted updates, the Sign-on screen is returned.



You must use the same terminal (TCT TERMINAL-ID) as the original task for RESTART to be detected.

Recovering RDML applications

To recover resources for all active tasks at the time of an abend, emergency restart CICS. The PDM automatically recovers its resources if task logging is active. CICS recovers VSAM resources if you included VSAM recovery in CICS. If you did not include VSAM recovery in CICS, the PDM updates are backed out, but the corresponding VSAM updates are not backed out. This is a problem only if the RDM application mixes PDM and VSAM updates. For recovery of DL/I databases in a CICS environment, refer to the appropriate IBM CICS IMS manuals.

Recovering MANTIS applications

When CICS abends, the PDM detects the abend and backs out uncommitted SUPRA PDM updates automatically. CICS emergency restart will recover VSAM files if you defined the files as recoverable. If you did not define the VSAM files as recoverable, the PDM updates are backed out, but the corresponding VSAM updates are not backed out. This is a problem only if a MANTIS application program mixes PDM and VSAM updates in the same logical unit of work. Refer to your MANTIS documentation for more information on MANTIS recovery. For recovery of DL/I databases in a CICS environment, refer to the appropriate IBM CICS IMS manuals.

Recovering directory maintenance

When CICS abends, restart CICS and enter the Directory Maintenance transaction ID. If the task issued a commit instruction, Directory Maintenance may receive a RESTART status and display the Restart screen.

Recovering after a PDM system abend

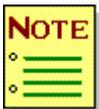
If CICS is not connected to the PDM when the PDM abends, no recovery is necessary for the CICS connector.

If CICS is connected to the PDM when the PDM abends, all current commands and those issued after the PDM abends receive a NOTO status. To recover, you must take the following steps:

- ◆ Disconnect the CICS connector from the PDM with the DISCONNECT operator control command.
- ◆ Restart the PDM if the PDM is operating in a central operating mode.
- ◆ Reconnect the CICS connector to the PDM in one of these ways:
 - Automatically, if you installed the automatic connect option at CICS initialization.
 - Manually, by issuing the CONNECT operator control command.

If the CICS Connector is connected to the PDM when the PDM abends, all current PDML instructions and all RDML instructions issued after the point of PDM failure receive a NOTO status.

To recover, disconnect the CICS Connector with the DISCONNECT command. If the PDM is operating in central mode, restart the PDM, then reconnect the interface using the CONNECT command. Upon restart, the PDM recovers all PDM resources for all tasks signed on at the time of PDM failure. If the PDM is operating in attached or attached central mode when the abend occurs, issue the CONNECT command to restart the PDM.



After you restart the PDM in a central operating mode, you must synchronize any updates being performed by tasks not running under CICS.

Recovering RDML applications

RDM application programs receive a fatal status from the CICS connector when a PDM system abends. A RDM RESET instruction cannot be used because the PDM is not active. PDM files are recovered when the PDM is restarted. If the application is updating VSAM files or DL/I databases with RDM, issue a CICS ROLLBACK command to recover the files.

Recovering MANTIS applications

If the PDM abends, the MANTIS application task receives a NOTO status, which is converted to an error message for the application program. Retry the task.

When you restart and reconnect the PDM, MANTIS may still receive notice that the task is not signed on and pass an error message to the application program. If this occurs, terminate the MANTIS task to ensure synchronization between PDM and CICS resources.

Recovering directory maintenance

If the PDM fails during a Directory Maintenance session, you receive the message "Backout Unsuccessful." When it reconnects, the PDM interrogates the Task Log File and issues a warm connect. If the task issued a commit, the PDM returns a RESTART status to Directory Maintenance after the sign-on instruction is issued, and Directory Maintenance displays a Restart screen showing the last update successfully completed.

If a commit was not issued, you receive a normal sign-on screen. You must use the same terminal (terminal ID) as the original task.

Recovering after both CICS and the PDM abend

To recover after concurrent CICS and PDM abends, you must take the following steps:

- ◆ Emergency restart CICS.
- ◆ Restart the PDM to recover the PDM resources.
- ◆ Reconnect the CICS connector to the PDM in one of these ways:
 - Automatically, if you installed the automatic connect option at CICS initialization.
 - Manually, by issuing the CONNECT operator control command.

Recovering RDML applications

If you are using RDML when CICS and the PDM abend concurrently, emergency restart CICS to recover CICS resources and restart the PDM to recover PDM resources.

Recovering MANTIS applications

Retry the MANTIS task after the PDM is restarted and reconnected. CICS resets CICS file updates if you defined the files as recoverable. The PDM backs out uncommitted PDM updates when you restart it.

Recovering directory maintenance

No task level recovery is attempted at the time of an abend. Recovery is performed to the beginning of all logical units of work when the SUPRA PDM is restarted. Log on to Directory Maintenance and resume your work from the last screen entered before system failure.

6

Recovering and restoring the SUPRA PDM from the System Log

Three SUPRA DBA utilities functions use the PDM System Log as input: the Recover function, which backs off changes, the Restore function, which reapplies changes, and the Log Print function, which prints the log's contents. You use these functions when the PDM has terminated and is not running.

When you cannot use the Task Log File to recover the database, you can use the Recover function to back off updates to the last commit. The Recover function has two phases. In the analysis phase, the Recover function reads the System Log file from the beginning, collecting and optionally printing information. In the image application phase, the Recover function reads the System Log file backwards from the end, applying before images to the database. The Recover function stops reading and applying images either at the last commit for each task or at the start of the file.

When PDM files are lost or damaged, you can use the Restore function to reapply updates to the last commit. You must reload the affected files from your backup copies before running the Restore function. The Restore function has two phases. In the analysis phase, the Restore function reads the System Log file from the beginning, collecting and optionally printing information. In the image application phase, the Restore function reads the System Log file forward from the beginning, applying after images to the database. The Restore function stops reading and applying images either at the last commit for each task or at the end of the file.

You can use the Log Print function to print selected information from the System Log file without updating the database. The Log Print function has one phase, the analysis phase, which corresponds to the analysis phase of the Recover and Restore functions. The Log Print function reads the System Log file from the beginning, collecting and optionally printing information.

For information on coding Recover, Restore, and Log Print, refer to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260.

Considerations

- ◆ The Recover function depends on before-image log records. For the Recover function to work, the PDM's active environment description must specify the before-image logging option.
- ◆ The Restore function depends on after-image log records. For the Restore function to work, the PDM's active environment description must specify the after-image logging option.
- ◆ The Recover and Restore functions cannot recover system events, such as formatting, loading, populating, and depopulating, because these activities are not logged. The Recover function to last commit is not affected by system events.
- ◆ If you changed a file's definition with Active Schema Maintenance during the PDM's run, attempting to recover that file with either the Recover function to log begin or the Restore function has unpredictable results.
- ◆ When you run the Recover function against a tape System Log under the VSE operating system, you must have a write-enable ring on the tape.
- ◆ With the VSE operating system using tape, RECOVER/RESTORE/LOG PRINT must have the tape drive positioned on the data file. Without exits, the utility does not process the tape label.
- ◆ For FBA users, to RECOVER/RESTORE/LOG PRINT from disk VSAM, ESDS must be used. Otherwise the FBA log file can be copied to a labeled tape for processing.
- ◆ When you run the Recover, Restore, or Log Print function against a tape System Log under the OS/390 operating system, the tape must be initialized before the System Log records are written or copied onto it. If the tape is not initialized, the function may fail with the IBM message IOS000I. When this happens, copy the System Log onto initialized tape(s) and try the function again.
- ◆ When you are not doing physical recovery you are not assured that the recovery will be physically exactly the same. Depending on how you specified Secondary Keys, it is possible that following the recovery the Index File may not look exactly the same as before. However, the Index File will be logically complete and consistent. This means that if your application is order entry dependent, you may notice minor differences.

Running recover and restore in the appropriate environment

The Recover and Restore functions must be run in an appropriate environment: non-Directory-driven, non-task-logging mode. The CSIPARM must include the DIRECTORY= parameter but not the REALM= parameter. The bootstrap environment description must specify, either explicitly or by default:

- ◆ TASK-LOGGING=NO
- ◆ STATISTICS=NO
- ◆ LOGGING=NNNN
- ◆ OPENX-OPTION=PROCESS

It must also specify an open mode of NONE for the files. The file definitions in the bootstrap schema must be identical to the corresponding definitions used by the terminated PDM. An example of a suitable CSIPARM member follows:

```
DIRECTORY=( SCHEMA=CSTASCHM, ENVDESC=CSTANONE ), END.
```

In some cases, you can run the Recover, Restore, and Log Print functions against part of the System Log instead of the whole log.

Considerations

- ◆ To run the Recover, Restore, and Log Print functions against part of the System Log, you must be using the multiple logical volumes feature, and the portion of the log you use must start at the beginning of a logical volume.
- ◆ You can run the Log Print function against a portion of the log starting with any logical volume.
- ◆ You can start to run the Recover function to last commit against a portion of the log starting with any logical volume. (If the portion does not include a commit point for every task, the Recover function terminates after the analysis phase and gives you a message identifying the volume with which to start.)
- ◆ You would normally run the Recover function to log begin or the Restore function against the entire log, starting with the first logical volume (volume number 0). These functions will not execute starting with any other volume unless there is an exit present.

If your operating system is OS/390 or VSE, your installation tape provides the following sample JCL members for the Recover, Restore, and Log Print functions. Use these samples as models for your JCL:

- | | |
|----------|--|
| TXJPMSLG | Run the Log Print function against a System Log that consists of multiple data sets. |
| TXJPSSLG | (VSE only) Run the Log Print function against a System Log that consists of a single data set. |
| TXJRCVLF | Run the Recover function to last commit against a System Log that consists of a single data set. |
| TXJRCVLG | Run the Recover function to last commit against a System Log that consists of multiple data sets. |
| TXJRSTLG | (VSE only) Run the Restore function to last commit against a System Log that consists of multiple data sets. |
| TXJRSTOR | Run the Restore function to last commit against a System Log that consists of a single data set. |

Determining the need for the Log File I/O Exit

You may need the Log File I/O Exit to run the Recover, Restore, or Log Print function. Needing this exit depends on the device type of your System Log, the number of data sets in the log (or in the portion of the log you run against), and the operating system you run under. [“Using the Log File I/O Exit in recovery”](#) on page 93 discusses the Log File I/O Exit and its use in detail.

Switching system log files

The recovery functions (Recover, Restore, and Log Print) use the System Log File to recover the Directory files and the PDM database files. When you execute the Restore function or Recover to log-begin, you must use the entire System Log File.

However, when you execute Recover to last-commit, you have the option to use only the last part of the System Log File. Also when you use the Log Print function, you are able to use whatever part of the System Log File you want. When you use only part of the file, those functions can execute faster than if you use the entire file.

To take advantage of this time saving feature, you must construct your System Log File so that it is divided into multiple logical volumes (several tape reels or disk data sets). Then you can select only the tape reel or disk data set you need. Another advantage of using several tapes or disks is that your System Log File never fills up. When the PDM fills one reel or disk, it just writes to another one. Then the PDM never abnormally terminates from a full system log file.

When you create a System Log File with several reels or disks, however, the recovery functions cannot read it. They can read only one tape reel or disk data set. To enable them to switch to the next reel or data set requires an exit program. For information on the PDM's logical write exit, refer to the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250.

Cincom provides exit points for maximum flexibility, so you can switch reels or data sets as you like. The only restriction is that you must pass them in sequential order.

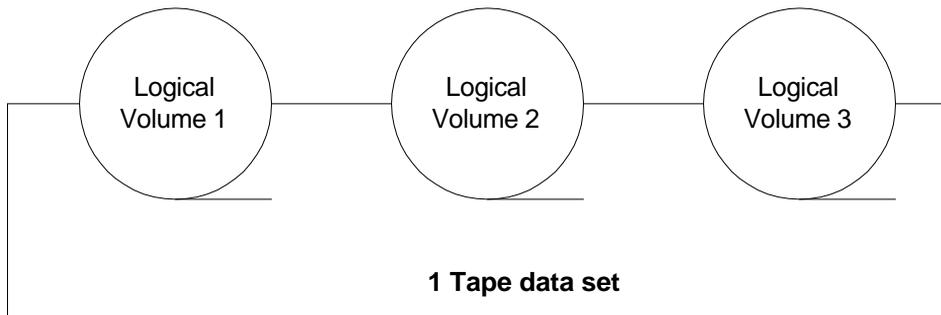
For more information on the Log File I/O Exit, [“Using the Log File I/O Exit in recovery”](#) on page 93.

Using tape data sets

You determine the number of reels in a logical log volume by the way you set the VOL-MAX-BLOCKS parameter in Directory maintenance. If you set the maximum number of blocks in a volume to less than the capacity of a tape reel, you have only one reel per volume. If you set the maximum number higher, you have more than one in each volume.

When you create the tape data set on Directory maintenance, you must declare the type of file to be either WORK or SYSTLOG. If you code WORK, you must set the VOL-MAX-BLOCKS parameter to less than the size of a reel. When you code WORK, the PDM abnormally terminates if you let it write to the end of the tape reel. The advantage of coding WORK is that the PDM executes faster because it can overlap the I/O.

The following figure shows an example of type 2, a tape data set. This type of System Log File is not divided into data sets and can be used by both WORK and SYSTLOG. In order for you to select a part of it for recovery, the PDM divides it into logical log volumes. In this case, it makes each reel a logical log volume of one data set.



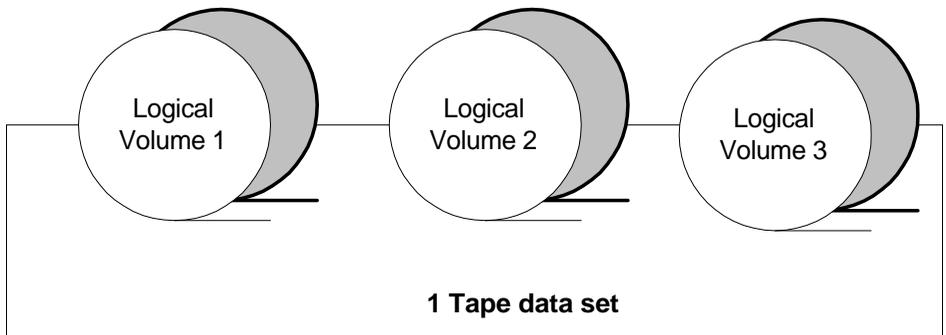
One tape data set containing three logical volumes

If you code SYSTLOG in type 2, the PDM cannot overlap I/O, but it can automatically switch to the next reel without abending. In this case, the PDM creates an infinite number of tape reels. Then you can let the PDM write to the end of a tape, and you can set the maximum blocks in a volume to a number larger than the capacity of a reel.

When you have more than one reel in a logical volume, you need to keep track of the reels so you can submit the ones you want to use in recovery to the exit program. To keep track of the reels in a logical volume, you can use the console messages the PDM prints when it begins and ends a logical volume.

When you submit the list of reels to the exit program, you must include all the reels in a logical volume. For example, if you need only the last reel for a Recover to last-commit, you could submit reel three in the preceding figure or reel six in the following figure. If you need the last two reels, you could submit reels two and three in the preceding figure. You must submit all reels in the Logical Volume in the following figure. If you leave out any reels, the Recover function stops soon after it begins executing.

To figure out which logical volumes to submit, start by submitting the last one. If Recover needs more, it prints an error message stating the number of the logical volume it needs.



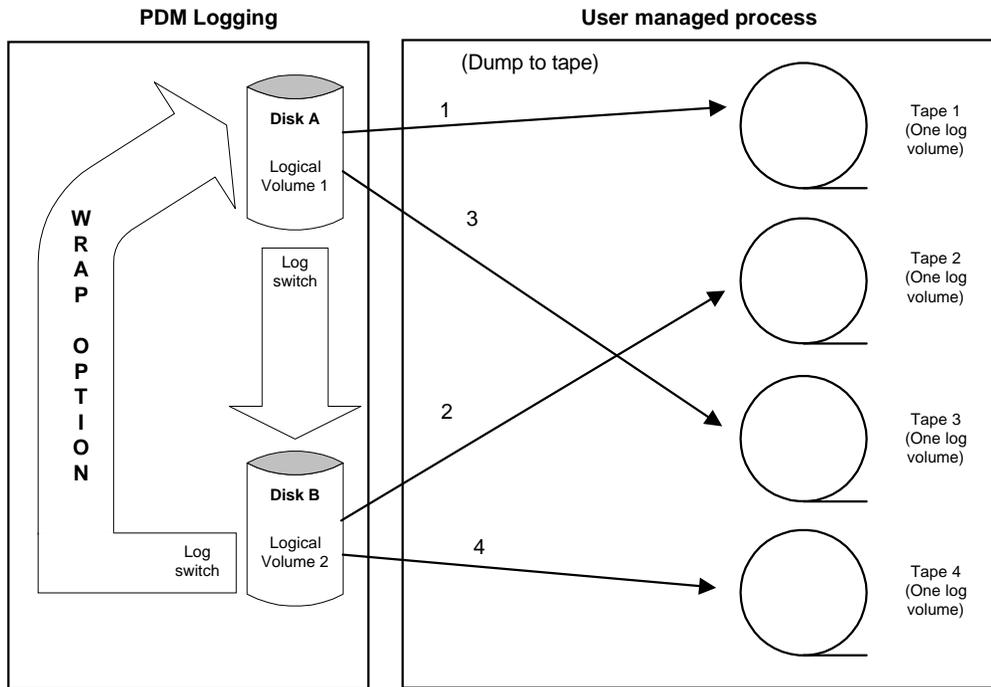
**One tape data set with six tapes
in three logical volumes**

Using disk data sets

The following figure shows an example of type 3, disk data sets. In this example, you have defined two disks and selected the wrap option. When the PDM starts logging, it writes to disk A.

When the PDM has filled A, it signals you to begin dumping to tape. If B has not been used, the PDM will begin logging to B immediately. If B has been used, the PDM will write a message to the system console requesting to use B. An operator reply is required.

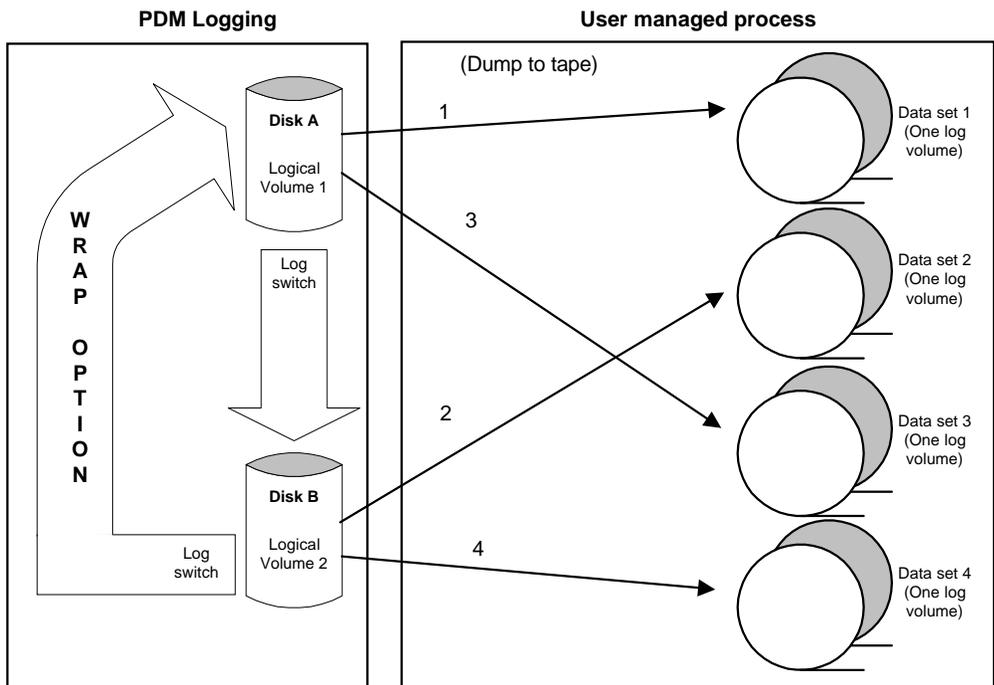
When B has completed dumping to tape, execute log switching and wrap log files by replying yes at the system console prompt. After filling B, you dump it to another tape as you did for A above, and the PDM wraps around to fill A again. Thus, the PDM can continue indefinitely filling data sets, so the System Log File never fills up, and you can select only the data set you need for recovery.



Two disk data sets with wrap option

The following figure shows another example of disk data sets. In this example, you have made the disk data sets so large that they require two tapes.

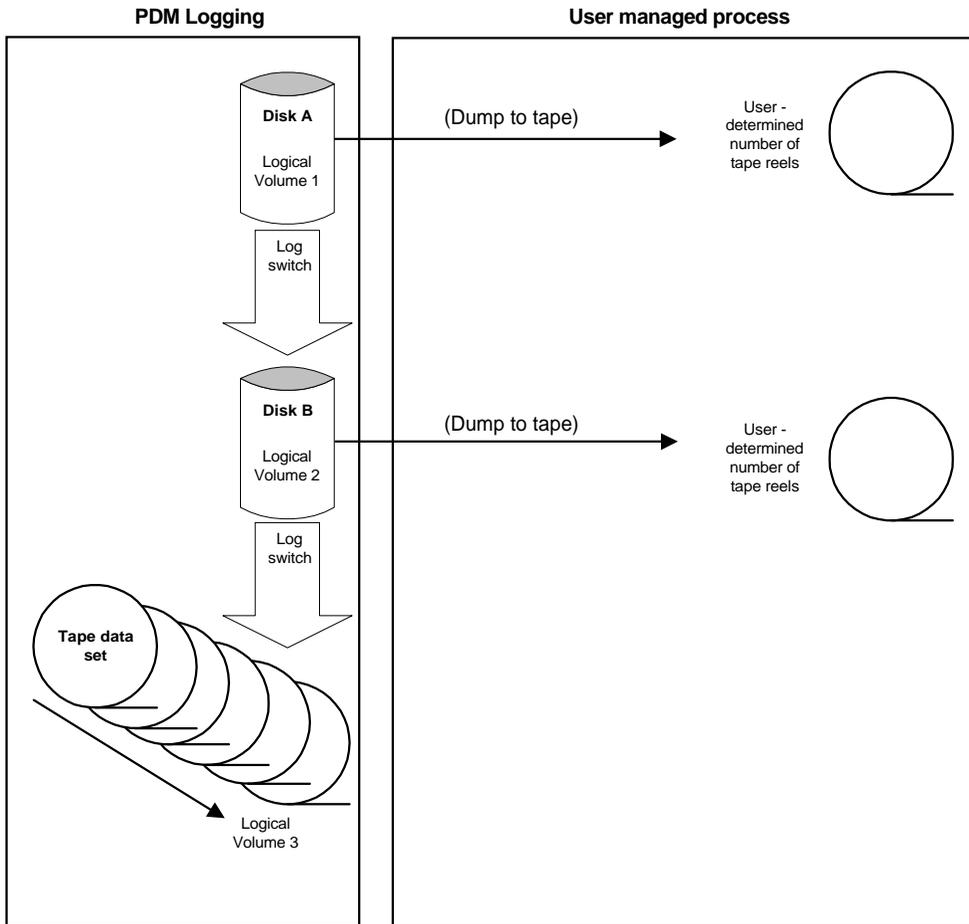
When you dump disks to tapes, the PDM makes a logical log volume with the tapes dumped from one disk. For example in the preceding figure, when you dumped the first disk to a tape, the PDM made it one log volume. When you dumped a disk to two tapes in the following figure, it made a logical log volume of them. Thus, you determine the number of tape reels in a logical log volume when you set the size of the disk data set. If you use a small disk data set, it is easier to keep track of the reels in a logical volume.



Two disk data sets that dump to two tapes each

Combining both disk and tape data sets

The following figure shows an example of type 4, a combination of disk and tape data sets. Each time the PDM fills a disk, you dump it to tape. You determine the number of reels in a logical log volume the same way as when you have disk data sets only. The main difference is that, in this case, the PDM ignores the wrap option. After the PDM fills the last disk, it begins filling the tape data set. Then it operates the same way as when you have a tape data set only. You set the number of reels in a logical volume the same way as when you have a tape data set.



Two disk data sets and one tape data set

7

Using the Log File I/O Exit in recovery

Cincom provides a Log File I/O Exit for the OS/390 and VSE operating systems. You may need this or an equivalent exit to run the recovery functions of the SUPRA DBA utilities: Recover, Restore, and Log-Print. Needing this exit depends on the device type of your System Log, the number of data sets in the log or the portion of the log you run against, and the operating system, as shown in the following table.

If your System Log portion consists of:	You need the exit when running under:	
	OS/390	VSE
A single disk data set	NO	NO
A single tape data set on one reel	NO	NO
A single tape data set on several reels	NO	YES
Several data sets	YES	YES

You can use the Log File I/O Exit in one of three ways:

- ◆ You can use the exit unchanged.
- ◆ You can modify the exit.
- ◆ You can use the exit as a model to write your own.

While the Log File I/O Exit is flexible enough to use in many situations, it can read only tape reels. If you do not modify the exit, you must move your disk data sets to tape. The OS/390 exit requires tapes with standard labels, and the VSE exit requires nonlabeled tapes. However, you may want to modify the exit program so it can read disk data sets.

The uses of the OS/390 and VSE exits are explained in [“Using the Log File I/O Exit under OS/390”](#) below and [“Using the Log File I/O Exit under VSE”](#) on page 109, respectively.

Using the Log File I/O Exit under OS/390

Before using the sample exit program, perform the steps summarized below and described in the following sections:

1. If your System Log includes disk data sets, move them to tape. The System Log must consist of tape data sets with standard labels.
2. You must create the load module CSUORCUX, which is not on the release tape. The figure at the end of “[Processing the Log File I/O Exit program](#)” on page 105 illustrates CSUORCUX and its components. For more information on creating CSUORCUX, see “[Creating the load module CSUORCUX](#)” on page 96.
3. In the UCL, invoke the load module CSUORCUX as a standard exit program. For more information, see “[Coding UCL for the Log File I/O Exit program](#)” on page 97.
4. In the JCL, code files in addition to the ones normally needed to execute the recovery functions. The additional DD statements are for the output and data set list files and for each System Log File data set you want to use in recovery. For more information, see “[Coding job control language](#)” on page 98.
5. In the data set list file, enter a record containing the ddname of each of the data sets that you want to use in the current run of recovery. For more information, see “[Listing data sets](#)” on page 100.



The examples in each section work together to make one example. The UCL example under “[Coding UCL for the Log File I/O Exit program](#)” on page 97, the JCL samples referenced on your installation tape, and the data set list example in the examples under “[Listing data sets](#)” on page 100 illustrate executing the Recover function to last-commit using three data sets.

Considerations

- ◆ All data sets must be tape files with standard labels and the same block size. Data sets may have one or more physical tape reels.
- ◆ The sample exit handles a maximum of 100 data sets. If you try to execute with more than 100 data sets, you get an error message. The recovery functions stop processing before they use any data sets. If you need to use more than 100 data sets, you must change the maximum size of the array, DATASET-LIST, in the source code in CSUUXRCO. You may make the array size smaller than 100. After changing the source code, you must recompile CSUUXRCO, then link edit the CSUORCUX load module again.
- ◆ When you are using a single tape data set without an exit and you code either STATISTICS (EXTENDED) or STATISTICS (ALL) in the UCL, you get volume information in the extended statistics report at the end of the log analysis phase. When you use an exit to perform the log file I/O, you do not get this volume information.

Creating the load module CSUORCUX

Before using the exit program, you must create the CSUORCUX load module, which is not on the release tape. Please ignore the object members for CSUUXCAP, CSUUXRCO, and CSUUXRAS on the release tape. To create the load module, follow these three steps:

1. Assemble the modules, CSUUXCAP and CSUUXRAS, with the source members provided.
2. Compile the COBOL module, CSUUXRCO, with the source members provided. You must specify the APOST option in your compiler options.



To help in problem diagnosis, Cincom recommends you specify the PMAP option. When you compile, concatenate the appropriate COBOL library in the SYSLIB JCL statement.

3. Link edit the CSUZRCUX link deck to create the CSUORCUX load module. When you link edit, concatenate the appropriate COBOL library in the SYSLIB JCL statement.

When you execute a recovery function, you do not have to concatenate a COBOL load library in the STEPLIB DD statement.

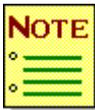
Coding UCL for the Log File I/O Exit program

This sample exit is dynamically loaded into memory at execution time from the STEPLIB DD libraries. When you execute a recovery function, the CSUORCUX exit load module must reside in a library concatenated in the STEPLIB DD statement.

To invoke the sample exit, code the CSUORCUX exit program in the STANDARD-EXIT statement of the UCL, as shown in the following example:

```
CONTROL (BEGIN)
**
**      RECOVER TO LAST-COMMIT
**      USER EXIT PERFORMS SYSTEM LOG FILE I/O
**
LOG-FILE ( )
  SEQ-ERROR (ERROR)
FUNCTION (RECOVER)
  STATE (LAST-COMMIT)
  STANDARD-EXIT (CSUORCUX)  STATISTICS (ALL)
  FILE (ALL)
**
CONTROL (END)
```

The recovery functions use only BLOCK-SIZE, SEQ-ERROR, PDM-ID-ERROR, and LOG-ID-ERROR statements in the control section of the UCL. (For information on coding these statements, refer to the [SUPRA Server PDM DBA Utilities User's Guide \(OS/390 & VSE\)](#), P26-6260.) The recovery functions ignore any values you code in the ACCESS-METHOD, DEVICE, and LOG-FILE statements. The values for these statements are supplied in the source code of the exit program. You must code the LOG-FILE statement because you code statements subordinate to it, such as the SEQ-ERROR statement.



Caution: You must code SEQ-ERROR (ERROR) to keep the recovery function from corrupting the database when a System Log data set is missing or out of order.

A recovery function considers a missing or out-of-order data set to be a sequence error and takes the action you coded in the SEQ-ERROR statement. If you code SEQ-ERROR (ERROR), the recovery function stops while analyzing the log file, that is, before applying any images. Applying images when a data set is missing or out of order could corrupt the database.

The Recover and Restore functions can corrupt the database only while applying images, not while analyzing the log file. The Log-Print function cannot corrupt the database because it only analyzes the file.

Coding job control language

In addition to the declaration statements already required to execute a recovery function, you must code DD statements for two special files the sample exit uses and for each data set in the System Log you want to use.

Two DD statements are for the following special files:

Statement name	I/O	Description
CSUUXOUT	output	General output file: You use the file to echo the records read from the CSUUXDSN file and to print any warning or error messages the exit generates. You must code this file in the JCL.
CSUUXDSN	input	Data set list file: You use the file to hold the list of data sets in the System Log that you will use in recovery. You must code this file in the JCL. It is described in more detail in “Listing data sets” on page 100.

The CSUUXIN file, declared in the source code of CSUUXRCO, is commented out. This file provides an example of a general input file. You can omit this file from the JCL.

The remaining DD statements are for the data sets in the System Log. You need to consider the following information when coding them:

- ◆ You must code a DD statement for each data set in the System Log that you want to use in the current recovery run. For example, if you want to use only the last two data sets in a System Log of ten data sets, you need DD statements for only those two. If you code all ten, the PDM ignores the ones it does not need.
- ◆ If you want to use only the last logical volume(s) in a tape data set, as in the illustration under “Listing data sets” on page 100, you must also code the volume serial number parameter (VOL=SER=nnnnnn) after the data set name parameter.
- ◆ The recovery functions read the data sets serially, so the job requires only one tape drive. If you have as many tape drives as data sets, you can allow the job to have more tape drives. If you code all the data sets in the System Log with UNIT=TAPE, the operating system tries to allocate a tape drive for each data set in the JCL.
- ◆ To use only one tape drive serially, use the volume affinity subparameter (AFF) of the UNIT parameter in the JCL. For the first data set, code UNIT=TAPE. For the second data set, code UNIT=AFF=ddname1 where “ddname1” is the DDNAME of the first data set in the list. For the third data set, code UNIT=AFF=ddname2, and so on.

Your installation tape provides JCL samples. For more information, refer to these samples on the installation tape:

TXJPMSLG Logprint-multiple data set system log file.

TXJRCVLG Recover to last commit using multiple data set system log.

Listing data sets

In the data set list file, you must code all the data sets in the System Log that you want used in the current run of the recovery function. Use a //CSUUXDSN DD statement to identify the data set list file in the JCL. In the CSUUXDSN file, identify the following:

- ◆ The data set DD statements in the JCL you want to use in the recovery function. You must code one data set record for each data set you want to use.
- ◆ The order of data sets on the System Log File. You must code the records in the same order as the data sets on the System Log.

In the data set list file, you code data set records and comment records. Construct the records in the file as shown in the following figure. Use an asterisk (*) in column 1 to indicate a comment record. The recovery functions ignore all other characters in that record. The recovery functions assume that any record not marked with an asterisk is a data set record.

Comment record	1	2-80
----------------	----------	-------------

Column 1	length 1	asterisk
Column 2-80	length 79	ignored

Data set record	1-8	9	10-53	54-80
-----------------	------------	----------	--------------	--------------

Column 1-8	length 8	DDNAME - must match corresponding DD statement in JCL
Column 9	length 1	filler
Column 10-53	length 44	DDNAME - reserved for future use; currently ignored
Column 54-80	length 27	filler

Record layout in CSUUXDSN file

The data set record contains two fields: DDNAME and DSNAME. The DDNAME field contains the DDNAME that corresponds to the DD statement in the JCL that describes the actual data set. The DSNAME field contains the DSNAME that corresponds to the DD statement in the JCL. The DSNAME field is not used by the exit and is reserved for future use. To visually verify that the data set record points to the correct DD statement, use the field to hold the correct DSNAME. For example, if the DD statement looked like this:

```
//SLOG0001 DD DISP=OLD,DSNAME=SYSLOG.D87FEB01
```

the DDNAME field would contain SLOG0001. The DSNAME field could contain SYSLOG.D87FEB01, comments, or blanks.

For example, on a Recover function to last-commit, uncommitted records might be on only the last three data sets. Only those three data sets need DD statements in the JCL and need to be in the data set list file.

The recovery functions can determine when a data set from the System Log is in the wrong order or is missing from the data set list. When these errors occur, recovery takes the action you coded in the SEQ-ERROR statement in the UCL. To stop the utilities from processing, code SEQ-ERROR (ERROR). For more information, see [“Coding UCL for the Log File I/O Exit program”](#) on page 97.

The following code listings show data set list examples. Both examples accomplish the same result because both contain the necessary data set records with the correct DDNAME field. The second example has no comment records and contains blanks in the DSNAME field.

Example 1

```
*
* THE FOLLOWING ARE TAPE DATASETS IN THE SYSTEM LOG
* THEY BEGIN AT 01 AUG 1988
*
* (THE DATASETS COMMENTED OUT ARE NOT USED IN THE RECOVERY)
*
*
*SLF0001  SYSLOG.TAP86001.ONE
*SLF0002  SYSLOG.TAP86002.ONE
*SLF0003  SYSLOG.TAP86002.TWO
*SLF0004  SYSLOG.TAP86002.THREE
.
.
.
*SLF0022  SYSLOG.TAP86011.TWO
*SLF0023  SYSLOG.TAP86012.ONE
*
* (THE FOLLOWING DATASETS ARE USED IN THE RECOVERY FUNCTION)
*
SLF0024  SYSLOG.TAP86012.TWO
SLF0025  SYSLOG.TAP86013.ONE
SLF0026  SYSLOG.TAP86013.TWO
*
* END OF DATASET LIST
*
```

Example 2

```
SLF0024
SLF0025
SLF0026
```

Handling errors

At execution time, the exit prints error or warning messages in the CSUUXOUT output listing file. The exit program echoes all the records from the CSUUXDSN input data set list file to the output file exactly as it reads them. After this output, the exit prints any error or warning messages. The recovery function stops after the exit prints an error message, but continues processing after the exit prints a warning message.

Messages contain a standard Cincom message identifier, an identifier (or nested identifiers) to trace which routines generated the message, and the message text. For example, the following message implies that the routine OPENSLF called the routine OPENDSN, and the open error occurred in OPENDSN:

```
CSUU2937E: OPENSLF: OPENDSN: OPEN ERROR FOR DATASET WITH
          DDNAME=ddddddd
```

In some cases, the IBM macro SYNADAF generates a message. The message has the following structure:

```
rrrrrrr: JJJJJJJJ,SSSSSSS,UUU,TT,DDDDDDDD,OOOOOO,EEE ...
          EEE,RRRRRRR,AAAAA
```

The following table explains the structure.

Field	Length	Description
r	7	Message header refers to the name of a higher-level routine within CSUUXRAS.
J	8	Job name
S	8	Step name
U	3	Unit address
T	2	Device type
D	8	DDNAME
O	6	Operation attempted
E	15	Error description
R	7	Relative block number
A	5	Access method

For more information about the messages returned, refer to the [SUPRA Server PDM Messages and Codes Reference Manual \(RDM/PDM Support for OS/390 & VSE\)](#), P26-0126.

Creating a system log to use with the exit program

The following is a suggested method of creating a System Log with several data sets:

- ◆ Define a log group for your System Log with two to four BSAM disk data sets.
- ◆ As data sets fill up, switch to the next data set in the circular log group and create a tape data set from the disk data set that fills.
- ◆ As you create each data set, add a data set record to the CSUUXDSN file and comment it out by adding an asterisk at the beginning of the record. Add a DD statement to the JCL stream you will use during a recovery function and comment it out.
- ◆ Make sure the DDNAMEs and DSNAMEs in the CSUUXDSN file match those in the JCL.
- ◆ When you want to execute a recovery function, uncomment out only those data sets you need for that run by removing the asterisk from the beginning of the record. The recovery function ignores all the other data set records in the CSUUXDSN file and all the DD statements the function does not use.
- ◆ When you execute recovery, code SEQ-ERROR (ERROR) under the LOG-FILE statement in the control section, and code CSUORCUX in the STANDARD-EXIT statement in the FUNCTION section. Code the remaining UCL as you would in any other execution of a recovery function.

For more details on creating System Logs, see [“Creating PDM system log file groups”](#) on page 39.

Processing the Log File I/O Exit program

The Log File I/O Exit program contains code to read from the System Log and to switch from one tape data set to another. The program is provided in source code, so you can see the logic the program uses.

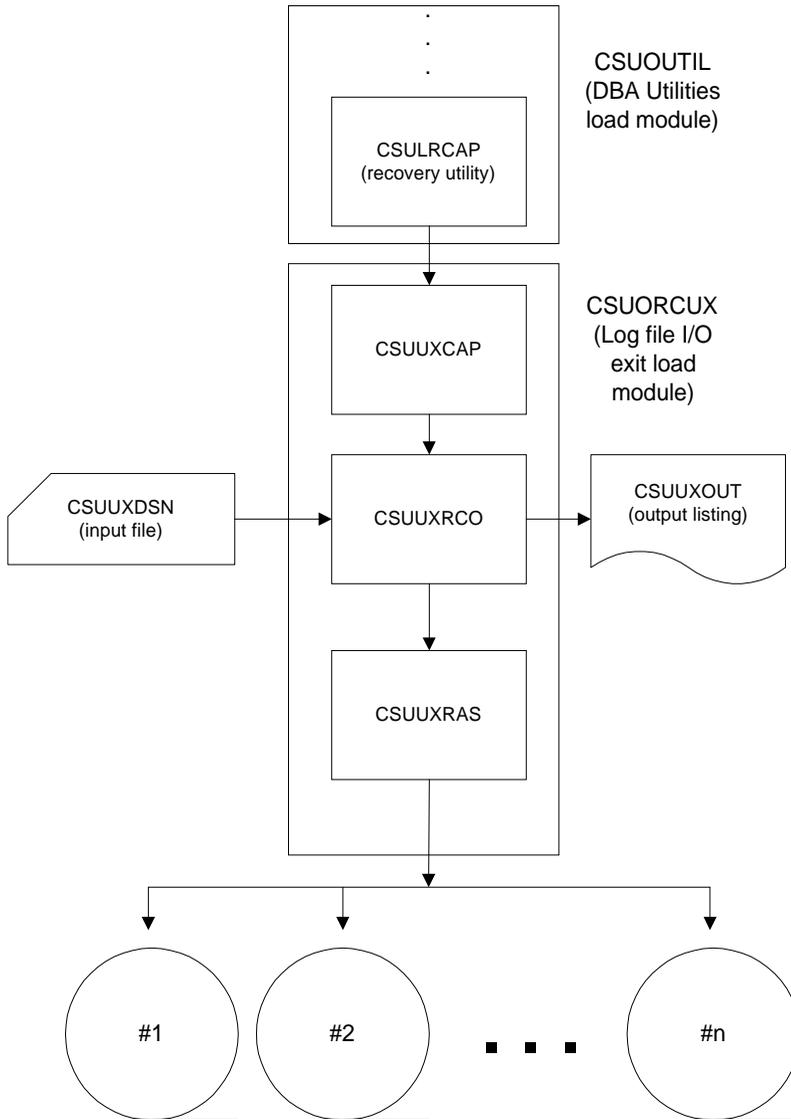
You can use this program in three ways:

- ◆ You can incorporate this program's logic into exit programs you already have.
- ◆ You can use this program as a prototype for writing your own program.
- ◆ You can add your own code to this program to use in any of the other recovery exit points.

The sample program contains logic to handle each of the 15 exit points called from the recovery functions. The program's logic is described below, and its structure is illustrated in the figure at the end of this section. For more detail, see the code and the documentation blocks within the code.

The sample exit program consists of the following modules:

- | | |
|----------|--|
| CSUUXCAP | A small cap routine, written in assembler: On the first call, the routine calls the COBOL procedure, ILBOSTP0, and then CSUUXRCO. On subsequent calls, the routine calls only CSUUXRCO. ILBOSTP0 tells COBOL that it is not a mainline program to keep COBOL from reinitializing itself each time it is called. CSUUXCAP contains a simple set of logic that is transparent to the recovery function and exit program. |
| CSUUXRCO | The COBOL portion of the exit: CSUUXRCO handles all exit points, initializes and terminates, and calls CSUUXRAS for more specific processing at the four Log File I/O Exit points. CSUUXRCO contains general logic so you can expand it to use with other exit points. |
| CSUUXRAS | The assembler portion of the exit: CSUUXRAS handles the actual log file I/O and System Log switching. CSUUXRAS is driven by CSUUXRCO. CSUUXRAS contains specific logic, but you can add more log file I/O logic. |
| CSUZRCUX | The recovery functions' exit link deck: CSUZRCUX is used in a composite link-edit to create the CSUORCUX load module that is the actual exit program. This load module contains the above modules and the ILBOSTP0 routine called by CSUUXCAP. |



Piece of system log you are using

Structure of the log file I/O exit in MVS

Processing in CSUUXRCO

The COBOL portion of the exit program, CSUUXRCO, contains a separate program for each exit point. At most of the exit points, the programs do no processing. CSUUXRCO includes only the logic necessary to read the data sets in the System Log and switch to the next data set. The following table summarizes its logic:

Exit	Exit name	Description of logic
1	Initialization	Open output message file, CSUUXOT. Turn on exit points 1, 2, 5, 8, 12, 13, 14, and 15 (so the recovery function calls them at the appropriate times). Turn off all other exit points.
2	Termination	Close output message file, CSUUXOUT
3	Analysis phase: Read a block	No processing
4	Analysis phase: Print a block	No processing
5	Analysis phase: Read a record	If an end-of-volume record is read, set an internal flag.
6	Analysis phase: Print a record	No processing
7	Application phase: Read a block	(Not called from the Log-Print function) No processing
8	Application phase: Read a record	(Not called from the Log-Print function) No processing
9	Application phase: Apply a record	(Not called from the Log-Print function) No processing
10	Application phase: Print an applied image	(Not called from the Log-Print function) No processing
11	Application phase: Buffering technique	(Not called from the Log-Print function) No processing
12	Log file open	Open data set list (CSUUXDSN) file. For each data set in the CSUUXDSN File: - Read from data set list file. - Echo into output message file. - Store in an array. Close data set list file. Call routine in CSUUXRAS to open first data set in System Log File.
13	Log file read	Call routine in CSUUXRAS to read forward (or backward) one block from the System Log File, switching to next (or previous) data set when necessary.
14	Log reset	(Not called from the Log-Print function) Call routine in CSUUXRAS to reset the System Log. Restore: Reset log file to read forward from beginning of file. Recover: Reset log file to read backward from end of file.
15	Log file close	Call routine in CSUUXRAS to close the last data set in the System Log.

Processing in CSUUXRAS

The previous module, CSUUXRCO, calls this assembler portion of the exit, CSUUXRAS, at each of the four Log File I/O Exit points. The following table summarizes the logic for the CSUUXRAS module:

Routine	Called by	Calls	Description of logic
OPENSFLF	CSUUXRCO exit #12	GETDSN OPENDSN	<ul style="list-style-type: none"> - Initialize system context. - Get first data set (GETDSN). - Open first data set (OPENDSN). - Get buffer space for all reads.
READSLF	CSUUXRCO exit #13	CLOSDSN GETDSN OPENDSN	<p>Read next/previous block (depending on whether reading forward or backward).</p> <p>If end/beginning of current data set:</p> <ul style="list-style-type: none"> - Close current data set (CLOSDSN). - Get next/previous data set (GETDSN). - Open next/previous data set (OPENDSN).
RSETSLF	CSUUXRCO exit #14	CLOSDSN GETDSN OPENDSN	<p>If the function is Restore:</p> <ul style="list-style-type: none"> - Close current data set (CLOSDSN). - Set system context to read forward from first data set - Get first data set (GETDSN). - Open first data set (OPENDSN). <p>If the function is Recover:- Set system context to read backward from current position in current data set.</p>
CLOSSLF	CSUUXRCO exit #15	CLOSDSN	<ul style="list-style-type: none"> - Close final data set (CLOSDSN). - Free buffer space.
GETDSN	OPENSFLF READSLF RSETSLF	None	<p>(Performs no processing)</p> <p>(Reserved for future use)</p> <p>(Currently, all data sets are in the JCL and do not require any logic to get them.)</p>
OPENDSN	OPENSFLF READSLF RSETSLF	None	<ul style="list-style-type: none"> - Open current data set. - Set system context to point to beginning/ending of data set.
CLOSDSN	READSLF RSETSLF CLOSSLF	None	Close current data set.

Using the Log File I/O Exit under VSE

Before using the sample exit program, you must perform the steps summarized below and described in the following sections:

1. If your System Log includes disk data sets, move them to tape. The System Log must consist of tape data sets on nonlabeled tapes.
2. You must create the load module CSUODUX, which is not on the release tape. For an illustration of CSUODUX and its components, see the illustration at the end of “[Processing the Log File I/O Exit program](#)” on page 120. For more information, see “[Compiling and link editing modules](#)” on page 111.
3. In the UCL, invoke the load module CSUODUX as a standard exit program, and code the correct block size of the System Log. For more information, see “[Coding UCL for the Log File I/O Exit program](#)” on page 112.
4. In the JCL, you must code files in addition to the ones normally needed to execute the recovery functions. The additional statements are for the output and volume list files and for each System Log File data set you want to use in recovery. For more information, see “[Coding job control language](#)” on page 113.
5. Enter as a record in the volume list file each of the physical reels of tape that you want to be used in the current run of recovery. For more information, see “[Coding the volume list](#)” on page 114.



The examples given in each section work together to make one example. The UCL example in “[Coding UCL for the Log File I/O Exit program](#)” on page 112, the JCL samples referenced on your installation tape, and the volume-list example at the end of “[Coding the volume list](#)” on page 114 show how to execute the Recover function to last-commit with three volumes.

Considerations

- ◆ You must put all volumes on tape reels with the same block size.
- ◆ The sample exit can handle a maximum of 100 tape reels. If you try to run with more than this number of volumes, you get an error message and recovery stops processing before it uses any volumes. If you need to use more than 100 reels, you must change the maximum size of the array, DATASET-LIST, in the source code in CSUUXDCO. You may also change the size of the array to less than 100. After changing the source code, you must recompile CSUUXDCO, then link edit the CSUODUX load module with the CSUZDUX link module again.
- ◆ When you are using a single volume tape without an exit and you code either STATISTICS (EXTENDED) or STATISTICS (ALL) in the UCL, you get volume information in the extended statistics report at the end of the log analysis phase. When you use an exit program to perform the log file I/O, you do not get this volume information.
- ◆ The sample exit program stops processing for only two reasons:
 - An internal logic error occurred. When this happens, the recovery function prints a message to the CSUUXOT file describing the error.
 - The operator aborts the utility. When this happens, the function does not print a descriptive message in the CSUUXOT file. Therefore, you must keep track of the messages sent to the console.

Compiling and link editing modules

When you first receive a release tape, you must create the CSUODUX load module. The release tape contains object members for CSUUXDCP, CSUUXDAS, and CSUUXDCO. Ignore all except CSUUXDAS.

1. Assemble the CSUUXDCP module with the source member provided. (You cannot assemble the CSUUXDAS module because you are not given the source code, only the object code.)
2. Compile the COBOL source module, CSUUXDCO. You must specify the APOST option in your compiler options. To help in problem diagnosis, Cincom recommends you specify the PMAP option. When you compile, concatenate the appropriate COBOL library in the LIBDEF statement in the JCL.
3. To create the CSUODUX load module, link edit the CSUZDUX link deck. When you link edit, concatenate the appropriate COBOL library in the LIBDEF statement in the JCL.

When you execute recovery, you do not have to concatenate a COBOL library in the LIBDEF statement in the JCL.

Coding UCL for the Log File I/O Exit program

This sample exit is dynamically loaded into memory at execution time from the library(s). When you execute a recovery function, the CSUODUX exit load module must reside in the core image library concatenated in the LIBDEF statement.

To invoke the sample exit, code the CSUODUX exit program in the STANDARD-EXIT statement in the UCL used to run the recovery function. You must also code the correct block size of the System Log in the BLOCK-SIZE statement. For an example of the UCL, see the following code sample.

The recovery functions use only BLOCK-SIZE, SEQ-ERROR, PDM-ID-ERROR, and LOG-ID-ERROR statements in the control section of the UCL. (For information on coding these statements, refer to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260.) The recovery functions ignore any values you code in the ACCESS-METHOD, DEVICE-ADDRESS, DEVICE, and LOG-FILE statements. The values for these statements are supplied in the source code of the exit program. You must still code the LOG-FILE statement because you must code statements subordinate to it.

```
CONTROL (BEGIN)
**
**      RECOVER TO LAST-COMMIT
**      USER EXIT PERFORMS SYSTEM LOG FILE I/O
**
LOG-FILE( )
BLOCK-SIZE (32767)
**
FUNCTION (RECOVER)
STATE (LAST-COMMIT)
STANDARD-EXIT (CSUODUX)
STATISTICS (ALL)
FILE (ALL)
**
CONTROL (END)
```

Coding job control language

In addition to the declaration statements already required when executing a recovery function, you must include declaration statements in the JCL for two special files the exit program uses and for the tape reels you want to use from the System Log.

You need ASSGN statements for the following special files:

Statement name	I/O	Description
CSUUXOT	output	General output file: This file echoes the records read from the CSUUXDS file and prints any warning or error messages the exit generates. The file is tied to device SYS040, but you can change this assignment by altering the file definition in CSUUXDCO. You must code this file in the JCL.
CSUUXDS	input	Volume list file: This file reads the list of volume serial numbers in the System Log that you will use in the recovery run. The file is tied to device SYS041, but you can change this assignment by altering the file definition in CSUUXDCO. You must code this file in the JCL. For more information on this file, see “Coding the volume list” on page 114.

The CSUUXIN file, found in the source code of CSUUXDCO, is commented out. This file provides an example of a general input file. Omit this file from the JCL.

You need one TLBL statement and one ASSGN statement for the System Log. You need to consider the following information when coding them:

- ◆ These declarations are used by the set of tape reels in the System Log that you want to use in the current recovery run.
- ◆ The recovery functions use the tape reels serially, so the job requires only one tape drive.
- ◆ This file is tied to device SYS010, but you can change this assignment by altering the file's value that is coded into CSUUXDCO.
- ◆ You can also change the TLBL name, LOGFILE, by altering the value of the TLBL name that is coded into CSUUXDCO.

Your installation tape provides JCL samples. For more information, refer to these samples on the installation tape:

- | | |
|----------|--|
| TXJRCVLG | Recover function to LAST-COMMIT on a multiple data set PDM System Log using the Log File I/O Exit program. |
| TXJRSTLG | Restore function to LAST-COMMIT on a multiple data set PDM System Log using the Log File I/O Exit program. |
| TXJPMSLG | Print contents of a multiple data set PDM System Log using the Log File I/O Exit program. |

Coding the volume list

The volume list is a list of volume serial numbers that drives the exit program to tell the operator which tape reels to mount.

When you create the volume list file for the JCL, you must code one entry for each physical reel of tape you use in the recovery functions. In some cases, you may not want to use all of the reels in the System Log. If so, you need to keep track of the tape reels that make up each logical log volume so you can specify the reels you need.

Use an assignment statement to indicate the volume list file in the JCL. In the volume list file, CSUUXDS, you identify the following:

- ◆ Number of tape volumes you want to use in the recovery function
- ◆ Volume serial numbers
- ◆ Order of those volumes on the System Log

Code comment records and volume records in the volume list file. Distinguish comment records by putting an asterisk in column 1. The recovery function ignores all other characters in that record. The function assumes that any record not marked with an asterisk is a volume record.

In the volume list file, code one volume record for each tape you want to use in the recovery function by putting its volume serial number in the VOL-SER field. You code the records in the same order as the tapes in the System Log.

The following figure illustrates constructing the comment and volume records.

Comment record	1	2-80
----------------	---	------

Column 1	length 1	asterisk
Column 2-80	length 79	comments (ignored)

Volume record	1-6	7-80
---------------	-----	------

Column 1-6	length 6	volume-serial number of tape
Column 7-80	length 74	filler (ignored)

Record layout in CSUUXDS file

Create the volume list as follows:

- ◆ As the PDM creates each volume, add a volume record to the CSUUXDS file and comment it out (that is, add an asterisk to the beginning of the record).
- ◆ When you want to execute recovery, uncomment (that is, remove the asterisk from the beginning of the record) only those volumes you need for that execution. The recovery function ignores all the other volume records in the CSUUXDS file.

For example, on a Recover function to last-commit, the uncommitted records might be on only the last three volumes. In that case, only those three need to be in the volume list file. For a sample volume list, see the example listings at the end of this section.

If you make errors on the volume list, the exit program can identify some of them are because it is driven by the list. The exit program can identify the following problems:

- ◆ When the operator has mounted the wrong tape
- ◆ When you listed a tape in the wrong order
- ◆ When you left a tape out of the middle of the list

If the operator mounts the wrong tape, the exit program sends the operator a console message indicating which tape to mount. If the list is incorrect, the operator should abort the recovery function, correct the list, and execute it again.

The exit program cannot tell if you omitted tapes from the beginning or end of the list. Thus, the exit program cannot warn you if you do not use the entire System Log when executing a Restore function or a Recover function to Log-begin.



Caution: If you do not use the entire System Log with the Restore function or the Recover function to Log-begin, you risk corrupting your database.

If you are executing a Recover function to last-commit, you must use a set of contiguous logical volumes that includes the last volume. For example, if you have a System Log of three logical log volumes, you cannot list just the tapes in the first and second logical volume. You must list the tapes in all three volumes. Likewise, if you list the tapes in the second logical volume, you must list those in the third volume.



Caution: If you list tapes in the middle, but not the end, of the System Log, you risk corrupting your database.

Example 1

```
(mono)*
* THE FOLLOWING ARE TAPE DATASETS IN THE SYSTEM LOG
* (THE DATASETS COMMENTED OUT ARE NOT USED IN THE RECOVERY)
*003231
*012345
*005678
.
.
.
*000096
*006667
*
* (THE FOLLOWING DATASETS ARE USED IN THE RECOVERY FUNCTION)
001234
002345
003456
*
* END OF DATASET LIST
```

Example 2

```
(mono)001234
002345
003456
```

Both examples accomplish the same result because both contain the necessary volume records with the correct VOL-SER field. The second example has no comment records.

Sending messages to the output listing

The exit program echoes all the records from the volume list file, CSUUXDS, to the output listing file, CSUUXOT, exactly as it reads them. (For an illustration of these files, see the figure at the end of “[Processing the Log File I/O Exit program](#)” on page 120.) Later during execution, the recovery function prints error or warning messages into the CSUUXOT output file. The recovery function aborts after the exit prints an error message but continues processing after the exit prints a warning message.

Messages from the exit program to the output listing contain a standard Cincom message identifier, a trace of which program(s) generated the message, and the message text.

For more information about the messages returned, refer to the [SUPRA Server PDM Messages and Codes Reference Manual \(RDM/PDM Support for OS/390 & VSE\)](#), P26-0126.

Sending messages to the console

The exit program sends messages that contain a standard Cincom message identifier and message text to the console.

During the execution of a recovery function, the exit sends messages to the console to request the mounting of each System Log tape by its volume serial number as it is needed. The exit determines the need for each tape volume by its position in the System Log as indicated in the volume list file.

After sending the mount message, the exit waits for your reply. To allow the function's processing to resume, you must mount the correct tape and reply GO. To terminate the function's processing, you must reply ABORT.



You must mount and ready a tape before replying GO. Even if you are going to mount the same volume, the tape must be unloaded and remounted.



You have the option to stop processing with an ABORT reply only during the log analysis phase of the function. Once the function has begun the image application phase, the exit no longer accepts a reply of ABORT.

If, after you have mounted the tape and replied GO, the exit detects that the tape is invalid or out of sequence, the exit sends an appropriate error message to the console. The exit then sends the mount message again and waits for your reply.

Processing the Log File I/O Exit program

The Log File I/O Exit program contains code to read the System Log and to switch tape reels. The CSUUXDCP and CSUUXDCO modules are released to you in source code, and the CSUUXDAS module is in object code. With the source code, you can see the logic the program uses. With the object code, you can use the module without assembling it.

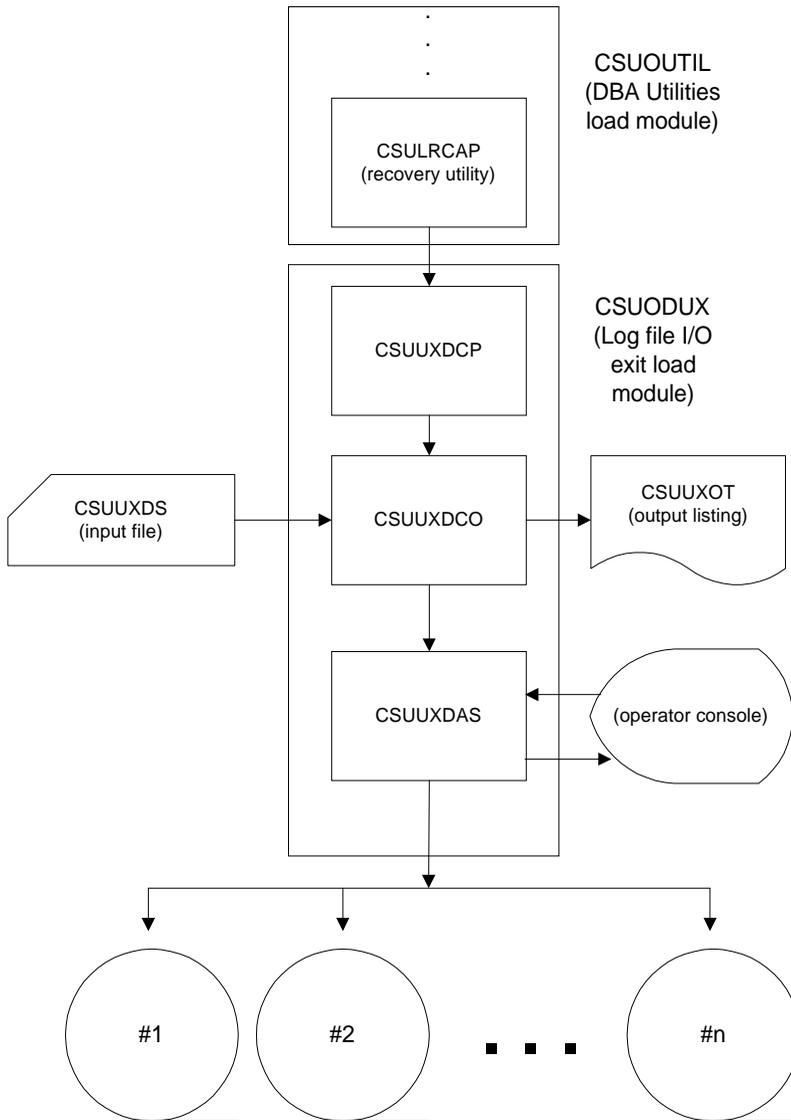
You can use this program in three ways:

- ◆ You can incorporate this program's logic into exit programs you may already have.
- ◆ You can use this program as a prototype for writing your own program.
- ◆ You can add your own code to this program to use in any of the other recovery exit points.

The sample program contains logic to handle each of the 15 exit points in the recovery functions. The program's logic is described below, and its structure is illustrated in the figure at the end of this section. For more detail, see the code and the documentation blocks within the code.

The sample exit program consists of the following modules, which are on your release tape:

- CSUUXDCP A small cap routine, written in assembler: On the first call, the routine calls a COBOL procedure, ILBDSET0, and then CSUUXDCO. On subsequent calls, the routine calls only CSUUXDCO. ILBDSET0 tells the COBOL exit program that it is not a mainline program and so keeps COBOL from reinitializing itself each time it is called. CSUUXDCP contains a simple set of logic that is meant to be transparent to recovery and the exit program.
- CSUUXDCO The COBOL portion of the exit program: CSUUXDCO handles all the exit points, initializes and terminates, and calls CSUUXDAS for more specific processing at the four Log File I/O Exit points. CSUUXDCO contains general logic so you can expand it and use it with the other exit points. However, if you leave CSUUXDCO as it is on the release tape, you may need to specify the APOST option when you compile it for COBOL under VSE.
- CSUUXDAS The assembler portion of the exit: CSUUXDAS reads the System Log and switches tape reels. It is driven by CSUUXDCO. CSUUXDAS contains specific logic, but it is meant to be expandable to add more log file I/O logic.
- CSUZDUX The recovery functions exit link deck: CSUZDUX is used in a composite linkedit to create the CSUODUX load module, which is the actual exit program. This load module contains the above modules and some internal Cincom modules.



Piece of system log you are using

Structure of the log file I/O exit in VSE

Processing in CSUUXDCO

The COBOL portion of the exit program, CSUUXDCO, contains a separate routine for each exit point. At most of the exit points, no processing is performed. CSUUXDCO includes only the logic necessary to perform log file I/O with System Log File switching. The following table is a summary of the logic for the CSUUXDCO module:

Exit	Exit name	Description of logic
1	Initialization	Open output message file (CSUUXOT). Turn on exit points 1, 2, 5, 8, 12, 13, 14, and 15 (so recovery function will call them at the appropriate times). Turn off all other exit points.
2	Termination	Close output message file (CSUUXOT)
3	Analysis phase: Read a block	No processing
4	Analysis phase: Print a block	No processing
5	Analysis phase: Read a record	If you read an end of volume record, set an internal flag.
6	Analysis phase: Print a record	No processing
7	Application phase: Read a block	(Not called from the Log-Print function) No processing
8	Application phase: Read a record	(Not called from the Log-Print function) Set an internal flag if you read an end-of-volume record and you are executing the Recover function. Ignore the record if you are executing the Restore function.
9	Application phase: Apply a record	(Not called from the Log-Print function) No processing
10	Application phase: Print an applied image	(Not called from the Log-Print function) No processing

Exit	Exit name	Description of logic
11	Application phase: Buffering technique	(Not called from the Log-Print function) No processing
12	Log file open	Open volume list (CSUUXDS) file. For each volume in the CSUUXDS file: - Read from volume list file. - Echo into output message (CSUUXOT) file. - Store in an array. Close volume list file. Call OPENSFLF routine in CSUUXDAS to open first volume in System Log.
13	Log file read	Call READSLF routine in CSUUXDAS to read forward (or backward) one block from the System Log, switching to next (or previous) volume when necessary.
14	Log file reset	(Not called from the Log-Print function) Call RSETSLF routine in CSUUXDAS to reset the System Log File. Restore: Reset System Log to read forward from beginning of file. Recover: Reset System Log to read backward from end of file.
15	Log file close	Call CLOSSFLF routine in CSUUXDAS to close the last volume in the System Log.

Processing in CSUUXDAS

CSUUXDCO calls the assembler portion of the exit, CSUUXDAS, at each of the four Log File I/O Exit points. The following table is a summary of the logic for the CSUUXDAS module.

Routine	Called by	Calls	Description of logic
OPENSLF	CSUUXDCO exit #12	OPENDSN	<ul style="list-style-type: none"> - Initialize system context. - Store address of volume list array. - Open file used for writing to operator. - Get buffer space for all reads. - Open first volume (OPENDSN).
READSLF	CSUUXDCO exit #13	CLOSDSN OPENDSN	<ul style="list-style-type: none"> - Handle end of volume if forced by an end of volume record. - Read next/previous block (depending on whether you are reading forward or backward). - Handle I/O error, if necessary. - Handle wrong length record, if necessary. <p>If end/beginning of current volume:</p> <ul style="list-style-type: none"> - If last/first volume in volume list: - Indicate end-of-file exit action. - Close current volume (CLOSDSN). - Open next/previous volume (OPENDSN).
RSETSLF	CSUUXDCO exit #14	CLOSDSN OPENDSN	<p>If reset is premature, abort with message.</p> <p>If the function is Restore:</p> <ul style="list-style-type: none"> - Close current volume (CLOSDSN). - Set system context to read forward from first volume. - Open first volume (OPENDSN). <p>If the function is Recover:</p> <ul style="list-style-type: none"> - Set system context to read backward from current position in current volume. - Reposition tape if necessary, (for example, back up across invalid record). - Write tapemark on tape (needed when opening a file for reading backwards). - Open the tape for reading backwards.
CLOSSLF	CSUUXDCO exit #15	CLOSDSN	<ul style="list-style-type: none"> - Close final volume (CLOSDSN). - Close file used for writing to operator - Free buffer space.

Routine	Called by	Calls	Description of logic
<p>OPENDSN</p>	<p>OPENSFLF READSLF RSETSLF</p>	<p>None</p>	<ol style="list-style-type: none"> 1. Write to operator to mount tape and await reply: If file open from previous mount attempt: <ul style="list-style-type: none"> - Close file. - Tell operator to dismount closed tape. - Tell operator to mount current tape. - Ask for reply of GO or ABORT. - If GO reply, go to (2). If ABORT reply: - If application phase, ABORT reply is invalid, go to (1) - Request confirmation of ABORT reply. - If second ABORT reply, abort recovery. - If reply is neither GO nor ABORT, ask for reply again (up to ten times, then assume GO). 2. Open current volume: <ul style="list-style-type: none"> - Set up DTF for forward/backward open. - If backward, position to end of tape. - Open current volume. - If open not successful, allow operator to mount again, go to (1). 3. Check for invalid or out-of-sequence volumes: <ul style="list-style-type: none"> - Read firstlast record on volume. - If I/O error, allow operator to mount again, go to (1). - If wrong length record, allow operator to mount again, go to (1). - If End of File found, allow operator to mount again, go to (1). <p>If this is first volume in volume list: THEN</p> <ul style="list-style-type: none"> - Store log file ID, PDM ID, and volume number (assume values from first vol. are correct). <p>ELSE</p> <ul style="list-style-type: none"> - If log file ID is not correct, display mount again; that is, go to (1). - If PDM ID is not correct, display mount again; that is, go to (1). - If volume number is out of sequence, display mount again; that is, go to (1). - Reset volume to position it was in before first/last record was read. 4. Set system context: <ul style="list-style-type: none"> - Set system context to point to beginning/ending of volume.
<p>CLOSDSN</p>	<p>READSLF RSETSLF CLOSSLF</p>	<p>None</p>	<ul style="list-style-type: none"> - Close current volume. - Write to operator to dismount current tape. - Accumulate relative block number across volumes.

Glossary of terms

block

The portion of a file which is read or written in a single physical input/output operation. The portion of a file which is read into or written from a single buffer.

commit point

For a given task, a point during PDM processing when the task commits, signs off, or is reset. A commit point occurs because of a command (commit, reset, or sign-off). At the commit point, a task's updated buffers are flushed, and temporary resources held by the task, such as held records, are released.

data record

A record that contains data, especially user data. It is distinguished from records that contain only control information, and from free-space records that contain no information.

log group

A PDM System Log File Group, synonymous with a PDM System Log, which is a group of one or more files (data sets). The Log Group entity, defined on the Directory as part of an environment description, describes a System Log File Group.

logical record

1. In a relational context, a tuple or row in a relation. Tuple is the preferred term.
2. A record; a logical subdivision of a file. It is distinguished from the physical block, which is that portion of a file which is read or written in a single physical input/output operation.

logical volume

A division of a PDM System Log. Every PDM System Log consists of one or more numbered logical volumes, starting with volume zero. The transition from one logical volume to the next usually coincides with the transition from one log tape to the next, or from one log data set to the next.

physical block

A block. The portion of a file which is read or written in a single physical input/output operation. The portion of a file which is read into or written from a single buffer.

physical record

1. In a relational context, a record; a subdivision of a file. It is distinguished from a tuple in a relation.
2. In IBM terminology, a block. The portion of a file which is read or written in a single physical input/output operation. The portion of a file which is read into or written from a single buffer.

quiet point

A point during PDM processing when no physical DML instruction (except the QUIET instruction) is in progress, all updated buffers are flushed, and all temporary resources, such as held records, are released. This point can only be caused by the QUIET DML instruction and only in a non-task-logging environment.

RBA

Relative byte address.

RBN

Relative block number.

record

A subdivision of a file.

relative block number (RBN)

A number identifying one block in a given file, relative to zero. The first block in a file has the RBN zero, the second has the RBN one, and so on.

relative BYTE address (RBA)

The location of a field or record in a given file, identified by the number of the byte at the beginning of the field, relative to zero. A field beginning with the first byte in a file has the RBA zero, a field beginning with the second byte has the RBA one, and so on.

relative record number (RRN)

A number identifying one record in a given file, relative to zero. The first record in a file has the RRN zero, the second has the RRN one, and so on. RRNs are meaningful only when all the file's records have the same fixed physical length.

RRN

Relative record number.

SFT

System File Table.

system file table (SFT)

The portion of the PDM Task Log File containing system-level rather than task-level information. SFT information includes the environment description, file descriptions, and the open status of each file.

system log file group

Synonymous with PDM System Log, which is a group of one or more files (data sets). The Log Group entity, defined on the Directory as part of an environment description, describes the System Log File Group.

task level recovery (TLR)

The capability to back off a task's updates to its last commit point and to restart the task at that point after an abnormal termination.

TLR

Task Level Recovery.

transaction

All the activity of a PDM task between one commit point and the next.

tuple

In a relational context, a logical record is a tuple or row in a relation.

VOL-MAX-RBN

The maximum relative block number (RBN) allowed for a given logical volume in a System Log File Group. Acts as a limit on the size of the volume; if VOL-MAX-RBN is 100, the maximum number of blocks in the volume is 101.

Index

A

Active Schema Maintenance 84
activity analysis 20
after image logging 35

B

back off changes 19
BDAM file 26, 39, 44
before image logging 35
block size
 PDM System Log(s) 44
 PDM Task Log File 26
bootstrap modules
 recovery functions 85
 scheme, task logging 27
 system logging 46
 task logging 26, 27, 30
 utilities, task logging 27
 validation module (VALMOD),
 task logging 27
BSAM file 39
buffers
 buffer pool records 38
 PDM Task Log File 27, 30

C

changes, backing off, reapplying,
 restarting 19
CICS
 CONNECT command 77, 79
 DISCONNECT command 79
 exits 72
 Program Control Table (PCT)
 72
 recovery 72
 ROLLBACK command 75
 syncpoint 73
COMIT 21, 58

command records 53
Create Environment Description.
 See bootstrap
Create VALMOD. See bootstrap
CSIPARM file
 Interactive Data Services 26
 recovery function 85
CSUORCUX, OS/390, creating
 load module 96
CSUUXDAS, VSE, processing
 log file I/O Exit in 125
CSUUXDCO
 VSE, processing log file I/O
 Exiting 98
 VSE, processing Log File I/O
 Exiting 123
CSUUXDRAS
 OS/390, processing log file I/O
 Exiting 108
CSUUXDS, OS/390
 coding volume list 114
 record layout in 115
CSUUXDSN, OS/390, record
 layout 100

D

data set list, OS/390 examples
 102
DATA sets, listing 100
depopulate function, secondary
 key 68
Directory Environment
 Description(s)
 CICS, recovery after task
 abend 74
 log groups, defining 47
 system logging 47
 task logging 32
Directory Maintenance
 CICS, recovery after system
 abend 78, 80
Directory Schema(s)
 system logging 47
 task logging 32
disk data sets, using 90
DL/I Database, access 72
DML command 53, 58

- E**
- Environment Description(s)
 - bootstrap, recovery functions 85
 - bootstrap, task logging 30
 - Directory, system logging 47
 - Directory, task logging 32
 - records 38
 - errors
 - messages 103
 - ESDS VSAM file(s)
 - PDM System Log File 39, 44, 45
 - PDM Task Log File 26
 - exits
 - CICS Exit Program 72
 - Log File I/O. *See* Log File I/O Exit
- F**
- file backup 57
 - description record 38
 - forced termination 22
 - group record 38
 - initialization records 53
 - mode before image record 37
 - recovering 56
 - File Level Log Suppression 35
 - Format function
 - recovering during 69
 - system log file 51
 - task log file 36
- G**
- group attributes 45
- I**
- I/O Exit. *See* Log File I/O Exit
 - image blocks 22
 - image record 54
 - index block log record 37, 54
 - index change log images record 37, 54
 - index split, failure during 68
 - initializing System Log files 51
- installation tape
 - description 39
 - JCL samples supplied 114
 - Interactive Data Services
 - PDM Task Log File attributes 25
 - PDM task purging 60
- K**
- KSDS VSAM file(s), recovery 67
- L**
- load module, creating
 - CSUORCUX 96
 - Log File I/O Exit
 - console messages 119
 - general 93
 - OS/390 94
 - coding JCL for 98
 - coding UCL for 97
 - CSUUXRAS processing 108
 - CSUUXRCO processing 107
 - determining need of 93
 - error handling 103
 - module creation 94
 - program logic 105
 - structure 105
 - output file messages 118
 - VSE 109
 - coding JCL for 113
 - coding UCL for 112
 - determining need of 93
 - module creation 111
 - structure 121
 - log file switching 87
 - log group(s) 39, 47
 - logging
 - options, PDM system Log 41
 - PDM. *See* PDM system logging; PDM task logging
 - selecting types of 21
 - uses 18
 - logical record types 37
 - logical volumes 43
 - logical write exit 87

M

MANTIS applications, CICS
 recovery after system abend 80
 recovery after task abend 76
 Slide # files 18, 35
 MARKL 58
 modify schema. *See* bootstrap
 utilities
 multiple logical volumes,
 switching 87
 multiple PDM Recovery 70

N

new volume exit 46

P

PDM initialization and termination
 records 53
 PDM System Log File Group(s).
See PDM system logging
 PDM system logging
 analyzing need of 23
 block size 44
 bootstrap modules 46
 characteristics 23, 43
 creating for exit program 104
 definition 18, 23
 Directory Maintenance 47
 end log option 42
 file description 39
 file group 23
 format 51
 I/O exit. *See* Log File I/O Exit
 initializing 51
 log record options 41
 logical volumes 43
 options 41
 PDM exits 39, 87
 record types 53
 redundant system logging 23
 switching 87
 synchronization option,
 description 23, 45
 wrap option, description 45, 90
 PDM task logging
 access method 27
 analyzing need of 21
 allocate 26
 attributes, changing 27

attributes, reviewing 25
 benefits of 21
 bootstrap modules 27
 bootstrap utility program 27
 buffers 27, 30
 defining in the bootstrap
 modules 27
 defining with Directory
 Maintenance 32
 definition 18, 21
 file size 28
 formatting 36
 log file 21
 logical record types 37
 placing 35
 reviewing 25
 S.F.T. blocks 38
 suppressing 35
 Personal File System (PFS),
 SPECTRA, recovery 72
 Physical Data Manager (PDM)
 multiple, recovery 70
 populate function, secondary
 keys 68
 purging tasks 60
 system failure 63
 task failure 62, 64
 warm start 61
 write log records 39
 purging, dangling tasks 60

R

record before image record 37
 recovery
 bootstrap environment
 description 85
 CICS 72
 CSIPARM file 85
 Directory Maintenance 76
 during format 69
 exits, general 93
 KSDS VSAM files 63
 Log File I/O Exit 93
 multiple PDMs 70
 PDM 83
 populate or depopulate 68
 secondary keys 64
 special cases 66
 system log 83
 task level 60, 62
 task Level 18

RESTART= 60
restartable tasks 61

S

schema(s)
 bootstrap, tasks logging 27
 Directory, system logging 47
 Directory, task logging 32
 SFT blocks 38
secondary key,
 populate/depopulate 68
SEQ-ERROR, recovery function
 UCL statement 101
SPECTRA
 Personal File System (PFS)
 recovery 72
 Slide files 18, 35
STANDARD-EXIT, recovery
 function UCL statement 112
STATE, recovery function UCL
 statement 97
STATISTICS, recovery function
 UCL statement 110
SUPRA migrating 58
switching System Log Files 87
system failure 63
System File Table (SFT) 22

T

tape data sets, using 88
task failure 62, 64
Task Level Recovery 18, 37, 60,
 62
Task Log File, PDM. *See* PDM
 task logging
Task Log ID Block 22
task(s)
 CICS, abends and recovery 72
 committing 19, 56, 57
 dangling 60
 failure 62, 64
 logfile 26
 purging 60
 records 53
 resetting 59
 restartable 61
 restarting 19
TLR. *See* Task Level Recovery
TOTAL migrating 58

U

UCL. *See* Utilities Command
 Language
utilities
 bootstrap 27
 format function, formatting PDM
 System Log File(s) 51
 format function, formatting PDM
 Task Log File 36
 format function, recovering 69
Utilities Command Language
 (UCL)
 Log File I/O Exit 97
 OS/390 97
 VSE 112

V

validation module, task logging
 27
VALMOD. *See* validation module
Volume List, example 116
VSAM file(s)
 ESDS
 PDM System Log File 45
 KSDS, recovery of 67
 native, recovery by CICS 72
 RRDS, SPECTRA PFS,
 recovery by CICS 72
VSE FBA users 44

W

warm start
 index split 68
 PDM 61
 secondary keys 68