

Cincom

## **SUPRA SERVER PDM**

System Administration Guide  
(VMS)

P25-0130-47



---

# SUPRA<sup>®</sup> Server PDM System Administration Guide (VMS)

## Publication Number P25-0130-47

© 1988–1989, 1990, 1993, 1994, 1996–2002 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This document contains unpublished, confidential, and proprietary information of Cincom.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 <sup>®</sup>	iD CinDoc <sup>™</sup>	MANTIS <sup>®</sup>
C+A-RE <sup>™</sup>	iD CinDoc Web <sup>™</sup>	Socrates <sup>®</sup>
CINCOM <sup>®</sup>	iD Consulting <sup>™</sup>	Socrates <sup>®</sup> XML
Cincom Encompass <sup>®</sup>	iD Correspondence <sup>™</sup>	SPECTRA <sup>™</sup>
Cincom Smalltalk <sup>™</sup>	iD Correspondence Express <sup>™</sup>	SUPRA <sup>®</sup>
Cincom SupportWeb <sup>®</sup>	iD Environment <sup>™</sup>	SUPRA <sup>®</sup> Server
CINCOM SYSTEMS <sup>®</sup>	iD Solutions <sup>™</sup>	Visual Smalltalk <sup>®</sup>
 gOOj <sup>™</sup>	intelligent Document Solutions <sup>™</sup>	VisualWorks <sup>®</sup>
	Intermax <sup>™</sup>	

UniSQL<sup>™</sup> is a trademark of UniSQL, Inc.  
ObjectStudio<sup>®</sup> 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 System Administration Guide (VMS)*, P25-0130-47, is dated January 15, 2002. This document supports Release 2.4 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](mailto: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

<b>About this book</b>	<b>ix</b>
Using this document.....	ix
Document organization .....	x
Revisions to this manual .....	xi
Conventions .....	xii
SUPRA Server documentation series .....	xv
<b>SUPRA Server overview</b>	<b>17</b>
SUPRA Server components.....	19
The Physical Data Manager .....	21
The Directory.....	23
The Relational Data Manager .....	24
SUPRA Server administration utilities .....	25
Selecting a SUPRA Server facility.....	27
Related products .....	29
SPECTRA .....	29
MANTIS.....	30
<b>Understanding the Physical Data Manager (PDM)</b>	<b>31</b>
Configuring the PDM.....	33
Groupwide PDM .....	34
Systemwide PDM .....	35
Multiple systemwide PDM .....	36
Systemwide and groupwide PDM .....	37
Multiple systemwide and groupwide PDM.....	38
Initializing the PDM.....	39
Manual PDM initiation .....	40
Automatic PDM initiation .....	42
Initiating the PDM on a network .....	48
Specifying a database .....	50
Using a database prefix .....	51
Writing SUPRA Server PDM user exits .....	55

<b>Defining your operating environment</b>	<b>57</b>
Creating a PDM environment .....	58
Defining logicals for your PDM environment.....	62
LOGICALS.COM .....	63
PDM_LOGICALS_*.COM.....	87
pdmname_USER_INIT.COM.....	109
SUPRA_SYMBOL.COM.....	110
Modifying VMS system parameters .....	111
<b>Entering input parameters</b>	<b>113</b>
Creating a PDM start-up resource file .....	114
Assigning a UIC name to the PDM.....	115
Specifying PDM quotas .....	116
Entering parameters for the PDM input file .....	125
Setting up and using PDM file protection checking .....	141
<b>Communicating with the SUPRA Server PDM</b>	<b>145</b>
Using the PDM operator commands .....	146
Activating an index (ACTIVATE).....	147
Deactivating an index (DEACTIVATE) .....	149
Disabling a database (DISABLE).....	151
Displaying a database (DISPLAY).....	154
Dumping the contents of the System Log for a database (DUMPSLF).....	158
Enabling a database (ENABLE) .....	160
Populating an index (POPULATE).....	162
Specifying read-only access for a database (READONLY).....	164
Shutting down a database (SHUTDOWN) .....	167
Unloading a database (UNLOAD) .....	169
Specifying update access for a database (UPDATE).....	171
Communicating with the SUPRA Server PDM through CSIOPCOM .....	174
Using CSIOPCOM commands .....	180
Running CSIOPCOM in batch .....	182
Automating operator communication.....	183
Restricting use of PDM commands .....	185
Communicating with the PDM through the VMS REPLY command.....	189

<b>Setting up the SUPRA Server Directory database</b>	<b>193</b>
The SUPRA Server Directory database .....	195
Estimating the SUPRA Server Directory data set sizes .....	196
Setting up the SUPRA Server Directory user names .....	197
Changing the definition of the SUPRA Server Directory database .....	198
Creating a recovery point .....	199
Modifying the SUPRA Server Directory database .....	200
Modifying the SUPRA Server Directory data sets .....	204
<b>Tuning your database</b>	<b>207</b>
Tuning your physical database .....	208
Defining the file access method .....	208
Avoiding fragmented files .....	208
Using data sets .....	209
Defining logical units of work .....	216
Managing buffers .....	217
Improving database performance with PDM cache .....	219
Optimizing Relational Data Manager performance .....	227
Accessing data sets .....	227
Choosing an RDM access method .....	228
Using bound views .....	229
Using Global Views .....	229
Using indexes .....	230
Designing application programs .....	232
Record holding .....	232
Managing record holding .....	234
Preventing a deadly embrace .....	237
Optimizing the frequency of commits .....	237
Understanding client read-ahead buffering .....	238
Context position considerations .....	239
Application programming considerations .....	239
PDM application guidelines .....	242
<b>Migrating a database</b>	<b>243</b>
Migrating into SUPRA Server .....	245
Migrating from SUPRA Server .....	246
Generating a DDL file .....	247
Using the DDL Load Facility .....	248
Signing on to CSDDLLOAD .....	250
Loading the DDL file .....	251
Checking CSDDLLOAD error conditions .....	257
Compiling the database description .....	258
Formatting data sets .....	258
Adding records .....	258

<b>Example user exits</b>	<b>259</b>
COBOL user exits.....	259
COBOL user exit 1.....	260
COBOL user exit 2.....	263
COBOL command file to compile and link the exits .....	266
FORTRAN user exit.....	267
FORTRAN user exit.....	267
FORTRAN command file to compile and link the exit .....	268
<b>PDM statistics output</b>	<b>269</b>
<b>Example mailbox-reading program</b>	<b>275</b>
<b>Optional SUPRA Server logicals</b>	<b>279</b>
<b>SUPRA Server logical names</b>	<b>281</b>
<b>Index</b>	<b>291</b>

---

# About this book

---

---

## Using this document

The manual is written primarily for the system administrator responsible for maintaining and tuning the SUPRA Server system, although DBAs and programmers may wish to use the tuning guidelines in “[Tuning your database](#)” on page 207.

Prerequisites for this manual include an understanding of VMS architecture, the DCL command language and system management, and high-level programming languages such as BASIC, COBOL, FORTRAN, and PASCAL.

This manual describes:

- ◆ SUPRA Server components and related products.
- ◆ The Physical Data Manager (PDM) in full detail.
- ◆ How to define your operating environment using SUPRA Server administration utilities, command procedures, and logicals and symbols.
- ◆ SUPRA Server input parameter files.
- ◆ How to communicate with the PDM using the PDM operator interface.
- ◆ How to set up the SUPRA Server Directory.
- ◆ How to get the best performance from your SUPRA Server database.
- ◆ How to migrate your database to and from other platforms.

The appendices provide sample user exit, a description of PDM statistics output, an example mailbox-reading program, a list of optional SUPRA Server logical names, and a complete listing of the SUPRA logical names.

## Document organization

The information in this manual is organized as follows:

### **Chapter 1—SUPRA Server overview**

Describes SUPRA Server components, administration utilities, and related products.

### **Chapter 2—Understanding the Physical Data Manager (PDM)**

Describes how to configure and initialize the PDM.

### **Chapter 3—Defining your operating environment**

Describes how to create a PDM environment, define logicals, and modify VMS system parameters.

### **Chapter 4—Entering input parameters**

Describes how to create a PDM start-up resource file, enter parameters, and use PDM file protection checking.

### **Chapter 5—Communicating with the SUPRA Server PDM**

Describes how to use the PDM Operator commands and how to communicate with the PDM in various ways.

### **Chapter 6—Setting up the SUPRA Server directory database**

Describes the SUPRA Server directory database and how to change its definition.

### **Chapter 7—Tuning your database**

Describes how to tune your database and optimize performance.

### **Chapter 8—Migrating a database**

Describes how to migrate to and from SUPRA Server, generate a DDL file, format data sets, and add records.

### **Appendix A—Example user exits**

Provides example COBOL and FORTRAN user exits.

### **Appendix B—PDM statistics output**

Describes the statistics written to the log file when you select STATISTICS=Y in the PDM input file.

### **Appendix C—Example mailbox-reading program**

Presents a COBOL program to read the PDM messages from a mailbox.

### **Appendix D—Optional SUPRA Server logicals**

Lists optional SUPRA Server logicals. These logicals change the normal behavior of SUPRA Server.

### **Appendix E—SUPRA Server logical names**

Lists the logical names needed to run SUPRA Server.

### **Index**

## Revisions to this manual

The following changes were made for this release:

- ◆ Calculations were added to the following parameters:
  - “AST\_LIMIT” on page 118
  - “BUFFER\_LIMIT” on page 119
  - “ENQUEUE\_LIMIT” on page 120
  - “EXTENT” on page 121
  - “FILE\_LIMIT” on page 121
  - “IO\_BUFFERED” on page 122
  - “IO\_DIRECT” on page 122
  - “PAGE\_FILE” on page 123
- ◆ The **CSIINDEX** logical was added on page 74.
- ◆ The **RUNDIRM** logical was added on page 81.
- ◆ The **[xxx\_]SUPRAD\_CSI\_PDM\_MACS** logical was added on page 97.
- ◆ “**WARMSTART\_DATASET\_ERROR**” on page 139.
- ◆ Information was added to number seven under “**Using DBA utilities on UDD files**” on page 206.
- ◆ Information on **MAXIMUM\_WORKING\_SET** was added on page 117.
- ◆ Information about understanding read-ahead buffering was added. See “**Understanding client read-ahead buffering**” on page 238.
- ◆ All references to revision 2.3 changed to read 2.4.
- ◆ The following logical names were added to Appendix E:
  - CSIINDEX
  - CSIPLVS\_DEB
  - RUNDIRM
  - [xxx\_]SUPRAD\_CSI\_PDM\_MACS
  - SUPRA\_LIBRARY
  - SUPRA\_UPGRADE

See “**SUPRA Server logical names**” on page 281 for the complete listing of logicals.

## 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> )	<p>Indicates a space (blank).</p> <p>The example indicates that four spaces appear between the keywords.</p>	<pre>BEGN<b>bbb</b>SERIAL</pre>
Brackets [ ]	<p>Indicate optional selection of parameters. (Do not attempt to enter brackets or to stack parameters.) Brackets indicate one of the following situations.</p> <p>A single item enclosed by brackets indicates that the item is optional and can be omitted.</p> <p>The example indicates that you can optionally enter a WHERE clause.</p> <p>Stacked items enclosed by brackets represent optional alternatives, one of which can be selected.</p> <p>The example indicates that you can optionally enter either WAIT or NOWAIT. (WAIT is underlined to signify that it is the default.)</p>	<pre>[WHERE <i>search-condition</i>]</pre> <pre>[<u>(WAIT)</u> (NOWAIT)]</pre>

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>	<pre>[ (WAIT) ] [ (NOWAIT) ]</pre>
	<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><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 keystroke. Multiple keystrokes are hyphenated.	ALT-TAB

---

## 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 for VMS Systems*, P25-9062.

### Overview

- ◆ *SUPRA Server PDM Digest for VMS Systems*, P25-9062

### Getting started

- ◆ *SUPRA Server PDM VMS Installation Guide*, P25-0147
- ◆ *SUPRA Server PDM VMS Tutorial*, T25-2263

### General use

- ◆ *SUPRA Server PDM Glossary*, P26-0675
- ◆ *SUPRA Server PDM Messages and Codes Reference Manual (PDM/RDM Support for UNIX & VMS)*, P25-0022

### Database administration tasks

- ◆ *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260
- ◆ *SUPRA Server PDM System Administration Guide (VMS)*, P25-0130
- ◆ *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220
- ◆ *SUPRA Server PDM Directory Views (VMS)*, P25-1120
- ◆ *SUPRA Server PDM Windows Client Support User's Guide*, P26-7500\*
- ◆ *SPECTRA Administrator's Guide*, P26-9220\*\*

### Application programming tasks

- ◆ *SUPRA Server PDM Programming Guide (UNIX & VMS)*, P25-0240
- ◆ *SUPRA Server PDM System Administration Guide (VMS)*, P25-0130
- ◆ *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220
- ◆ *SUPRA Server PDM Windows Client Support User's Guide*, P26-7500\*
- ◆ *MANTIS Planning Guide*, P25-1315\*\*

### Report tasks

- ◆ *SPECTRA User's Guide*, P26-9561\*\*



---

Manuals marked with an asterisk (\*) are listed twice because you use them for different tasks.

---



---

Educational material is available from your regional Cincom education department.

---

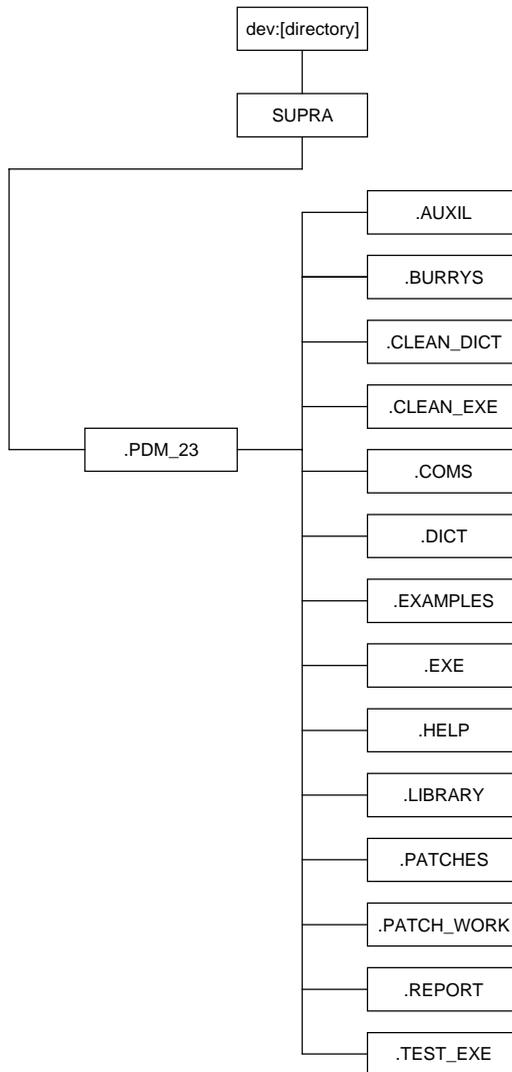
# 1

## **SUPRA Server overview**

SUPRA Server is an interactive database system that allows you to use advanced features for control of data resources and high programming productivity. SUPRA Server accommodates the varying data processing needs of its VMS users by providing a variety of integrated components and related products.

SUPRA Server consists of an intricately connected system of powerful components. Each component has important functionality that allows you to efficiently and effectively administer the SUPRA Server database.

The SUPRA Server directory structure is outlined in the following figure:

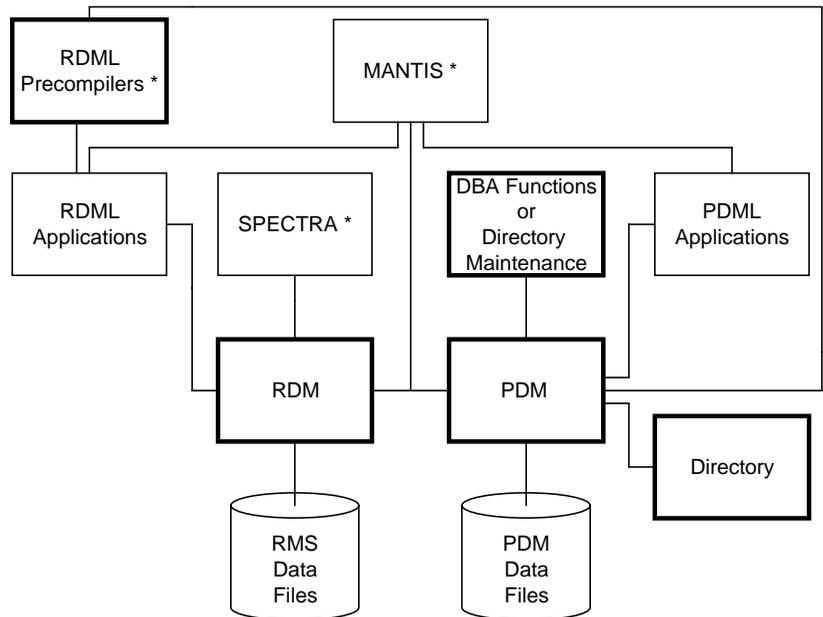


You can customize the locations of these areas by using the SUPRA\_MENU.COM administration utility (see “[Creating a PDM environment](#)” on page 58).

## SUPRA Server components

The integrated components of SUPRA Server combine to give you a fully functioning SUPRA Server system. This chapter introduces these components and prepares you for detailed descriptions of the components that are found in subsequent chapters.

The following figure illustrates the SUPRA Server system, complete with the SUPRA Server components and related products that can help you effectively use your SUPRA Server system. The SUPRA Server components are indicated by the bold boxes; the related products are indicated by an asterisk and are marked optional.



\* Optional



SPECTRA is not available in Alpha environments.

The following table shows the functionality of the SUPRA Server components:

<b>Component</b>	<b>Functionality</b>
Physical Data Manager (PDM)	Recovery Heterogeneous cluster and network support Automatic restart Reads, updates, and deletes data Services requests for data
Directory	DBA functions Fast utilities Batch Directory Maintenance Database verify utilities Batch validate, compile, and print
Relational Data Manager (RDM)	DBAID Global view creation View binding RDML

## The Physical Data Manager

The Physical Data Manager (PDM) runs as a multitasking, detached VMS process. The PDM controls the storage of and access to data in user databases. It is the underlying control method that all other components use to access physical data.

You use the PDM either directly or indirectly through the other SUPRA Server components. Application programmers can write programs that use:

- ◆ Physical Data Manipulation Language (PDML) to directly manipulate data held on SUPRA Server databases.
- ◆ Relational Data Manipulation Language (RDML) statements to invoke the PDM through the Relational Data Manager (RDM).
- ◆ MANTIS to manipulate data on SUPRA Server databases.

End users on VAX systems use SPECTRA to access the PDM through RDM views and queries.

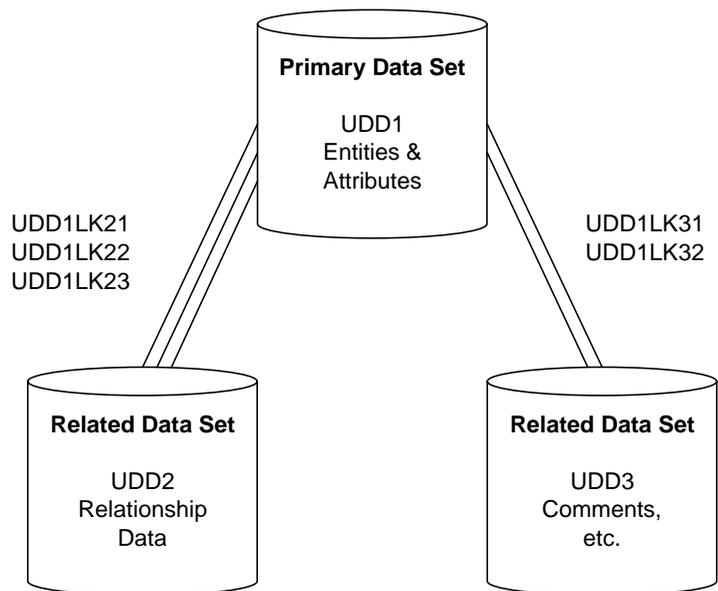
Regardless of the access method, the PDM provides the following additional features:

- ◆ **Automatic restart.** If a database is running on a machine that fails, and a user attempts to access a database on the failed machine, the PDM is initiated on the first available machine. The first available machine is determined from the preferred machine list. See [“Defining your operating environment”](#) on page 57 for a description of automatic restart.
- ◆ **Heterogeneous cluster and network support.** Provided by the PDM for each database by using a preferred machine list. Heterogeneous Cluster Support allows a task from one machine to access a database running on another machine, where both machines form part of the same cluster. SUPRA Server Network Support enables a task from one node to access a database running on another node, where both nodes form part of the same network. These facilities enable the transfer of data between machines on a cluster, or nodes on a network. [“Defining your operating environment”](#) on page 57 describes how to set up your PDM to start on any machine in a cluster or network in descending order of preference. This order of preference depends on the order of machines as specified in the preferred machine list.
- ◆ **Recovery methods**, including:
  - Task level recovery, which resets the database to the last successful commit point after a system or task failure.
  - System level recovery, which restores the database to its state just prior to a device failure by reapplying logged after images.
  - Shadow recording, which duplicates each data set change on a shadow data set, and switches to the duplicate data set when a failure occurs.
  - Recovery unit journaling, which is optionally available for task level recovery for the RMS data sets you can use with the RDM.

## The Directory

The SUPRA Server Directory is the central point for all SUPRA Server components. It contains all the VMS system information and all the data descriptions (metadata) required by the PDM. The SUPRA Server Directory also contains the relational entities required by the RDM. Although it does not contain user data (user databases contain this), it contains a common repository of what data exists and where. See [“Setting up the SUPRA Server Directory database”](#) on page 193 for more information on the Directory and how to maintain it.

The following figure shows the structure of the SUPRA Server Directory:



## The Relational Data Manager

The database administrator (DBA) designs base views and derived views of data held in SUPRA Server databases. The RDM then enables application programmers and end users to use these views to access the database without concerning themselves with its physical structure. They do not need to know anything about navigation through the database; the RDM retrieves the required data while providing database security and integrity. In addition, the RDM allows the DBA to change and restructure the database without requiring programmers to rewrite or recompile their programs. The RDM also provides programmers with a simple Relational Data Manipulation Language (RDML) for retrieving and modifying the database contents. Refer to the *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220, for details of RDM processing and use.

The RDM provides the following facilities to help the DBA define and maintain views:

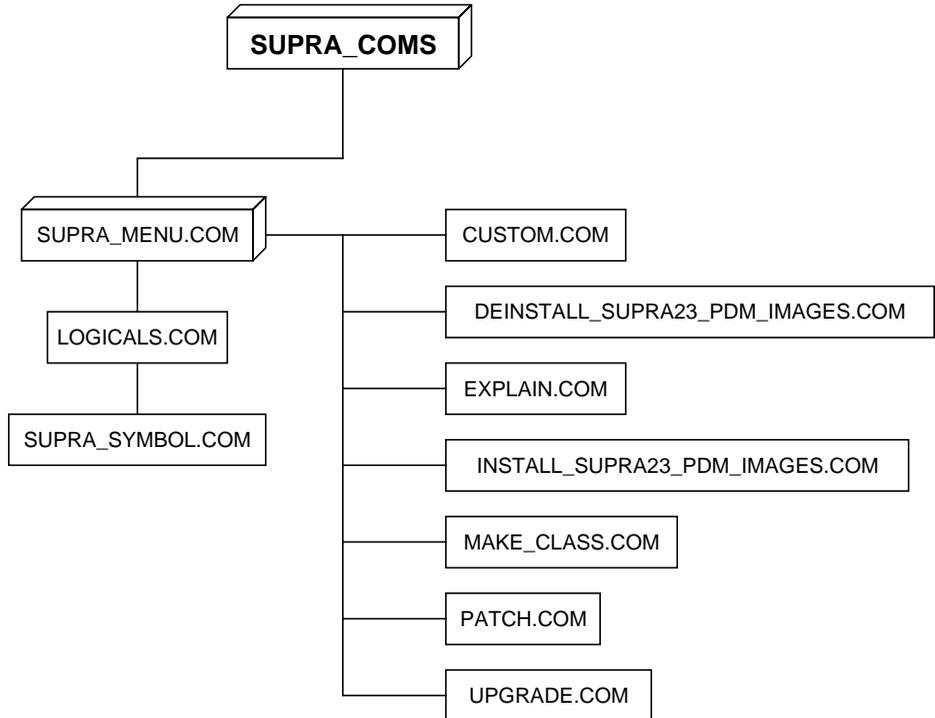
- ◆ **DBAID.** Used by the DBA to define new views, open views, issue RDML statements, and examine the results. The DBA can then change the view and immediately test it again. DBAID can execute in batch or online.

Non-DBA users can use the programmer's subset of DBAID commands to test whether a logical view meets application requirements before production runs. Refer to the *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220, for a description of the DBAID test facility.

- ◆ **View binding.** Binding stores preopened copies of logical views on the Directory. Logical View functions in DBA or the DBAID BIND command can be used to bind logical views. Each subsequent view access uses the bound copy, thus saving the processing time required to open the view. “[Tuning your database](#)” on page 207 discusses the performance advantages of using bound views.
- ◆ **Global view creation.** The global view facility lets the DBA place a preopened copy of frequently used logical views in global memory. This allows all users to share a permanently loaded open view definition. “[Tuning your database](#)” on page 207 discusses the performance advantages of using global views. Refer to the *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220, for details of how to use the global view facility.

## SUPRA Server administration utilities

SUPRA Server comes with a set of administration utilities that help you administer SUPRA Server in your VMS environment. These utilities are located in the [...SUPRA.PDM\_24.COMS] subdirectory. Once you define your environment, the logical SUPRA\_COMS points to this directory. The following figure lists some of the files that are stored in SUPRA\_COMS:



As a result of creating a PDM environment, the following files are created in the SUPRA\_LIBRARY directory:

- ◆ PDM\_LOGICALS\_\*.COM (this file is now located in the SUPRA\_LIBRARY directory)
- ◆ PDM\_START\_\*.COM
- ◆ *pdmname*\_USER\_INIT.COM
- ◆ PDM\_OPTIONS\_\*.INP

See “[Creating a PDM environment](#)” on page 58 for a complete description of these files.

SUPRA Server administration utilities allow you to configure your environment with logical and symbol definitions. Details of the administration utilities and their corresponding logicals and symbols are in “[Defining your operating environment](#)” on page 57.

The following command procedures help you configure your PDM environment:

- ◆ **SUPRA\_SYSTEM.COM.** When you define your first PDM system, this procedure is created in the SUPRA\_COMS directory. This command procedure sets up all of the standard SUPRA Server logicals for each PDM system defined. It does this by making the appropriate calls to LOGICALS.COM and PDM\_LOGICALS\_\*.COM (see below).
- ◆ **LOGICALS.COM.** Defines general SUPRA Server logicals in the logical name table specified.
- ◆ **PDM\_LOGICALS\_\*.COM.** Defines logicals needed for a specific PDM environment and the specific database(s) associated with it. This procedure is created when you use SUPRA\_MENU.COM to create a PDM environment. This file is now located in the SUPRA\_LIBRARY directory.
- ◆ **SUPRA\_SYMBOL.COM.** Defines VMS symbols that you use to access various utilities from the DCL command line. You can also access many of these facilities from the SUPRA Server Facilities Menu detailed in “[Defining your operating environment](#)” on page 57.



---

Throughout this manual, an asterisk represents either the 6-character UIC group number for a groupwide PDM, 000000 for a systemwide PDM, or the 1- to 8-character name of a multiple systemwide PDM.

---



The following table shows where to find more documentation for each option on the menu:

Option	Documented in
DBA function	<i>SUPRA Server PDM Database Administration Guide (UNIX &amp; VMS)</i> , P25-2260
Global view creation	<i>SUPRA Server PDM RDM Administration Guide (VMS)</i> , P25-8220
SPECTRA	<i>SPECTRA User's Guide</i> , P26-9561 (available only in VAX environments)
DBAID facility	<i>SUPRA Server PDM RDM Administration Guide (VMS)</i> , P25-8220
BASIC preprocessor	<i>SUPRA Server PDM Programming Guide (UNIX &amp; VMS)</i> , P25-0240
COBOL preprocessor	<i>SUPRA Server PDM Programming Guide (UNIX &amp; VMS)</i> , P25-0240
FORTRAN preprocessor	<i>SUPRA Server PDM Programming Guide (UNIX &amp; VMS)</i> , P25-0240
MANTIS	MANTIS manuals

---

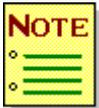
## Related products

The following related products may help you effectively use your SUPRA Server system:

### SPECTRA

End users use SPECTRA, an advanced query language, for reporting and updating data. These users need no programming knowledge to write SPECTRA processes to interrogate databases, nor do they need to use application programs. This reduces the burden on the application programmers. In addition, programmers and the DBA can use SPECTRA to create and store processes for use by any authorized user.

SPECTRA uses the views defined by the DBA on the SUPRA Server Directory. The same views can be used by application programs, SPECTRA, and the DBAID facility. Thus, all the advantages of the Relational Data Manager (RDM) are available to end users, as well as to programmers.



---

SPECTRA is not available in Alpha environments.

---

## MANTIS

MANTIS is an application development tool with facilities to design programs, scenarios, files, screens, prompters, and interfaces to applications written in other languages. The MANTIS programmer can access all SUPRA Server components. MANTIS offers high programming productivity and fourth-generation prototyping facilities. In addition, MANTIS applications are portable across hardware platforms.



---

SPECTRA is designed as an ad hoc query tool, not an application development environment. Those users that require a full application development environment should consider using MANTIS.

---

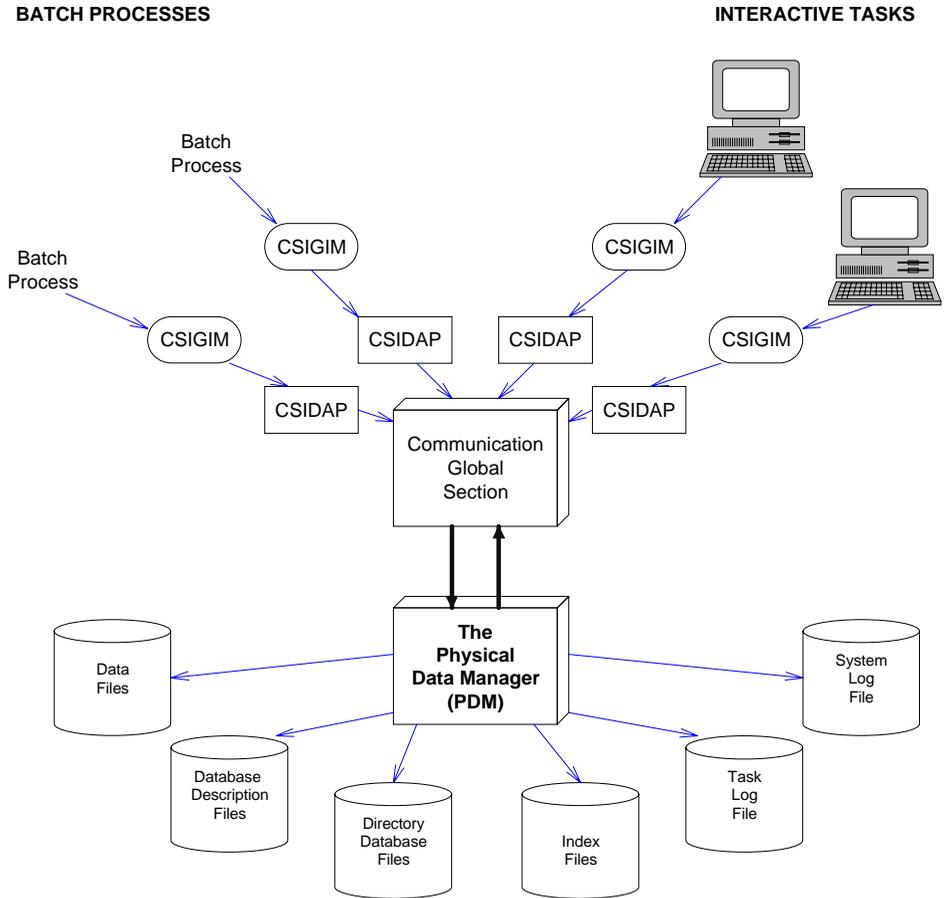
# 2

## Understanding the Physical Data Manager (PDM)

The Physical Data Manager (PDM) runs as a detached VMS process. It controls the storage of and access to data.

The PDM runs as a multithreaded process. Multithreading allows numerous applications to use the PDM at the same time and leads to efficient use of system resources.

The following figure shows the structure of PDM processing and the types of files handled by the PDM:



## Configuring the PDM

How you configure your PDM depends on how you want users to have access to your database(s). Decide on the following options before you generate your PDM system:

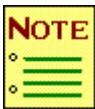
- ◆ **Groupwide PDM.** This PDM can be accessed exclusively by members of the specific VMS UIC group. It can only load groupwide databases for the specific UIC group.
- ◆ **Systemwide PDM.** This PDM can be accessed by users in different VMS UIC groups. It can load systemwide or groupwide databases. If users in different UIC groups need to access the same data, you must use a systemwide database that is serviced by a systemwide PDM.
- ◆ **Multiple systemwide PDM.** Multiple PDMs can be accessed by users in different VMS UIC groups. If you need more than one PDM that is systemwide, you must use multiple systemwide PDMs. This PDM configuration is a systemwide PDM and it can service both systemwide and groupwide databases.

This configuration permits insulation between two or more systemwide PDMs. For example, a crash of a multiple systemwide PDM used for test purposes will not affect a different multiple systemwide PDM used for production.

The PDM runs as a detached process. All logical definitions that will be used by the PDM process must be available to it. This requires the logical definitions for a groupwide PDM to be made at the group level. For a systemwide database, they must be made at the system level. For a multiple systemwide PDM, certain logicals must be at the system level and the rest must be in the special logical name table `CSI_PDM_pdmname`, as described in “[Defining logicals for your PDM environment](#)” on page 62.

If `CSI_PDMID` is in the group table, the PDM is groupwide. If `CSI_PDMID` is in the system table and not in the group table, the PDM is systemwide. See “[PDM\\_LOGICALS\\_\\*.COM](#)” on page 87. If the logical `CSI_SYSPDMID` is defined, the PDM is multiple systemwide. See “[pdmname\\_USER\\_INIT.COM](#)” on page 109.

You can combine groupwide PDMs with systemwide and multiple systemwide PDMs. You can find examples of this in “[Systemwide and groupwide PDM](#)” on page 37 and “[Multiple systemwide and groupwide PDM](#)” on page 38.




---

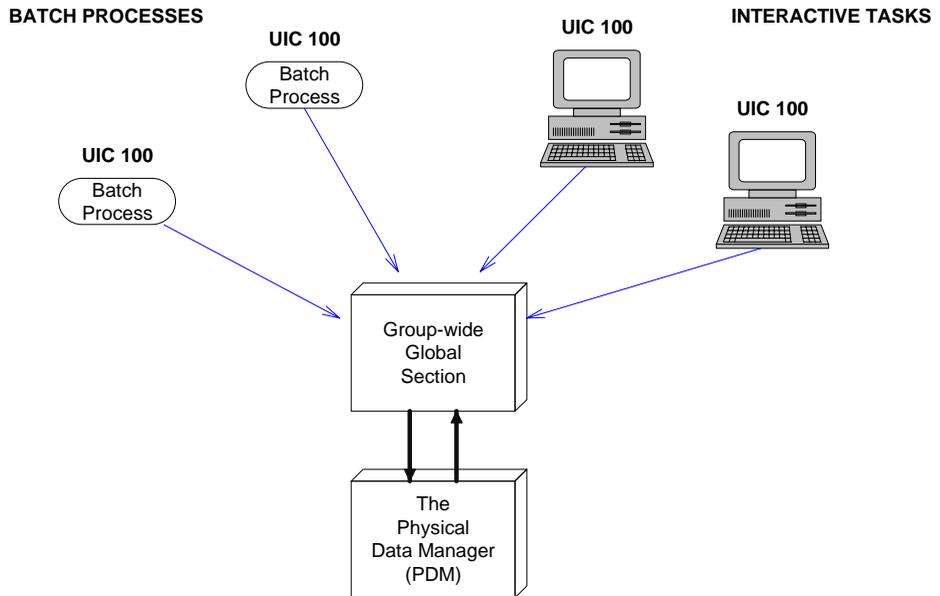
If your processing requirements change and you need to reconfigure your PDM environment, use `SUPRA_MENU`.

---

## Groupwise PDM

A groupwise PDM and the databases it services are only available to users that are in the same VMS UIC group as the PDM. Each groupwise PDM has a specific corresponding VMS Global Section. The logicals of a groupwise PDM are in the group logical name table LNM\$GROUP\_\*, where \* is the number of the VMS UIC group.

The following figure illustrates a VMS system with a groupwise PDM environment:



Logicals reside in the LNM\$GROUP\_000100 table.

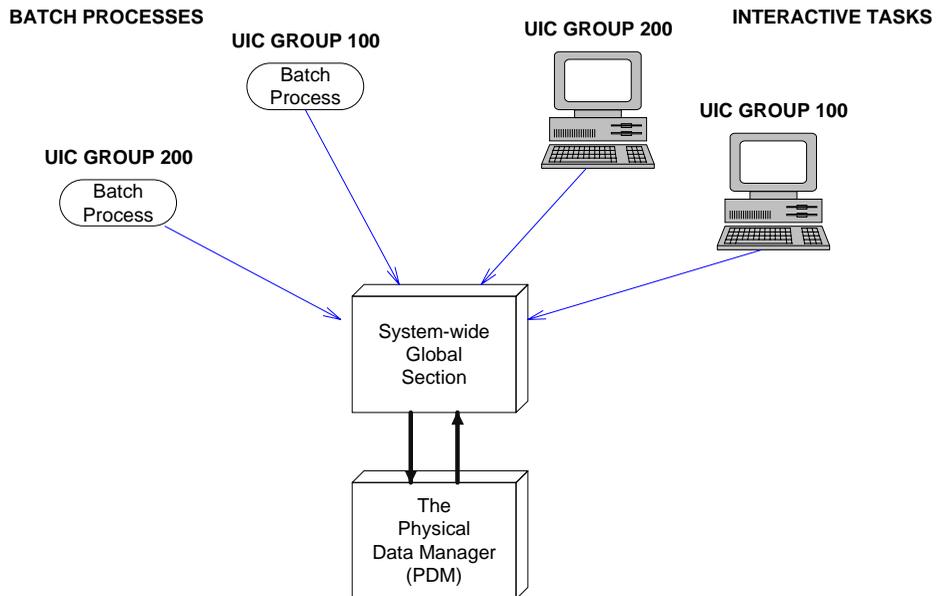
Each batch and interactive task has the logical CSI\_PDMID defined to the same value as a result of executing SUPRA\_COMS:PDM\_LOGICALS\_000100.COM.

## Systemwide PDM

A systemwide PDM can be accessed by users in different VMS UIC groups. You use a systemwide PDM if users in different VMS UIC groups need to access the same data. The logicals for a systemwide PDM are defined in the system logical name table LNM\$SYSTEM.

A systemwide PDM can service groupwide and systemwide databases.

The following figure illustrates a systemwide PDM system with two different VMS UIC groups:



Logicals reside in the LNM\$SYSTEM table.

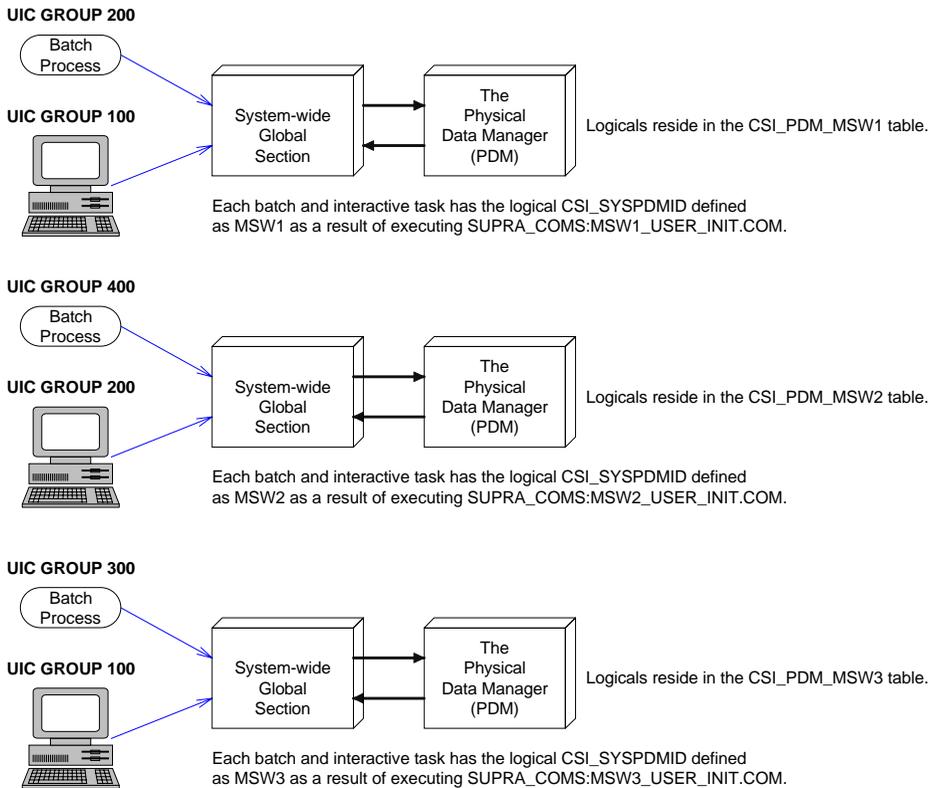
Each batch and interactive task has the logical CSI\_PDMID defined to 000000 as a result of executing PDM\_LOGICALS\_000000.COM.

## Multiple systemwide PDM

A multiple systemwide PDM environment allows you to have more than one systemwide PDM. Each VMS UIC group has access to several PDMs. The PDMs use systemwide VMS Global Sections for communication with the application tasks.

The logicals for the PDMs are in the special logical name table `CSI_PDM_*`. An additional logical, `CSI_SYSPDMID`, specifies the name of the PDM that is used. This logical is defined in the file `pdmname_USER_INIT.COM` (see “*pdmname\_USER\_INIT.COM*” on page 109). It can be defined in any logical name table available to the application task. For more information on logical name tables, see “*Defining logicals for your PDM environment*” on page 62.

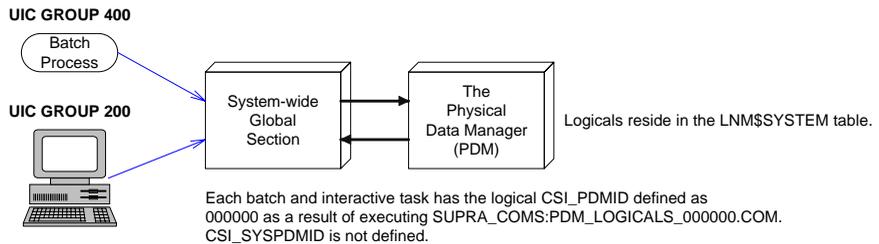
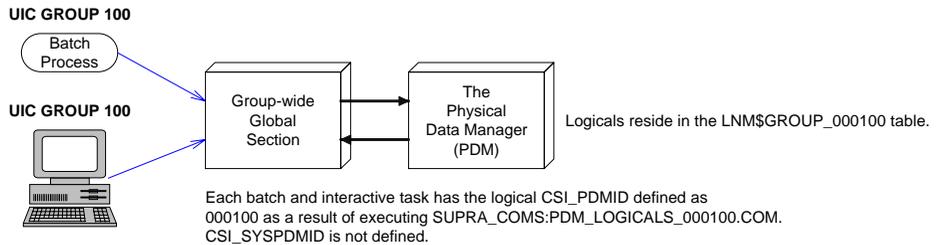
The following figure illustrates a multiple systemwide PDM configuration with four separate VMS UIC groups and three PDMs:



## Systemwide and groupwide PDM

You can use both systemwide and groupwide PDMs on the same VMS system. In such a configuration, the users using the groupwide PDM cannot access a systemwide PDM. For example, the VMS UIC group 000100 in the following figure can only use the related groupwide PDM.

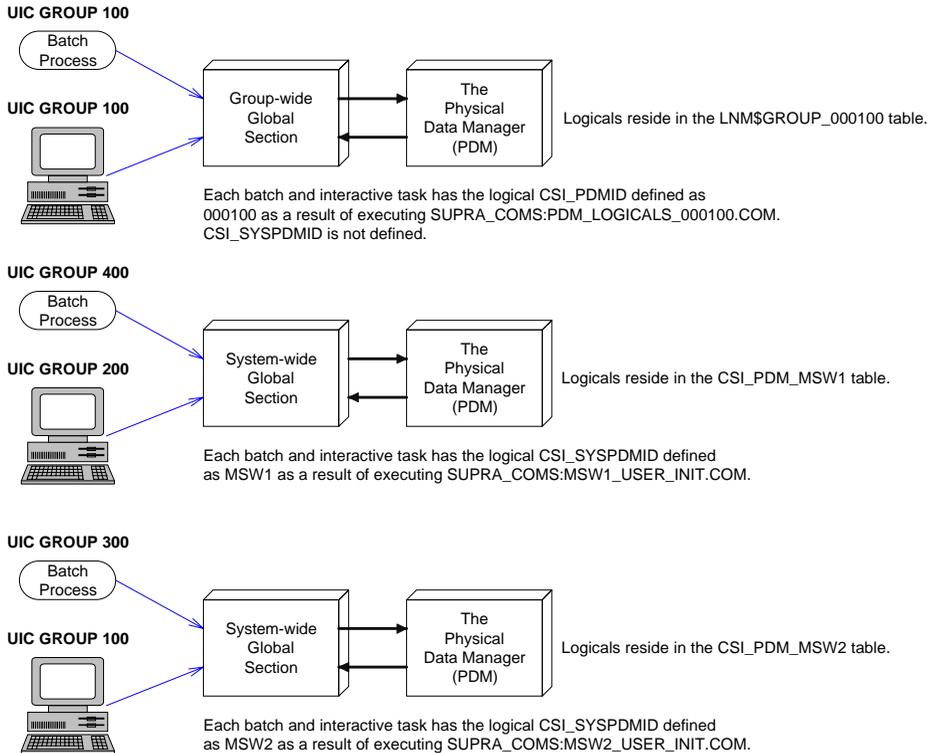
The following figure illustrates two VMS UIC groups connected to a systemwide PDM, and one VMS UIC group connected to a groupwide PDM:



## Multiple systemwide and groupwide PDM

You can use both multiple systemwide and groupwide PDMs on the same VMS system.

The following figure illustrates four VMS UIC groups with two multiple systemwide PDM configurations, and one groupwide PDM configuration:



## Initializing the PDM

PDM initiation methods allow you to control whether you initiate your PDM automatically or manually. If you initiate the PDM automatically, the first attempt to use a database initiates the PDM. If you initiate the PDM manually, you must initiate the PDM at the command level before any users attempt to sign on to any database serviced by the PDM. If you are using manual initiation and the PDM has not been initiated before a user attempts to use a database, the user will receive an error message and will not be able to access the database.

The logical definition `CSI_AUTOSTART` tells SUPRA Server whether to initiate the PDM automatically or manually. When you set `CSI_AUTOSTART` to YES, the PDM initiates automatically. When you set `CSI_AUTOSTART` to NO, the PDM must be manually initiated.

You choose your PDM initiation method when you use `SUPRA_MENU.COM` to create your PDM environment. After you set `CSI_AUTOSTART` to either YES or NO, it will be defined in the procedure `PDM_LOGICALS_*.COM` for your PDM. For details on logical definitions, `CSI_AUTOSTART`, and `PDM_LOGICALS_*.COM`, see [“Defining your operating environment”](#) on page 57.

If you have defined your original environment with `CSI_AUTOSTART` set to NO and wish to change it's value to YES, you will also need to define a logical name in the `SUPRA_LIBRARY:PDM_LOGICALS_*.COM` file. It needs to be created at the same level and in the same table as the other logical names defined in this procedure. The logical name is `CSI_*` where the \* is the 6-digit group UIC or the 000000 system UIC or the translation of the `CSI_SYSPDMID` logical name. This logical name points to the `SUPRA_LIBRARY:PDM_START_*.COM` file. For example:

```
$ DEFINE/NOLOG/TABLE=LNM$GROUP_000100 CSI_000100
DKAO:[CINCOM.SUPRA.PDM_24.LIBRARY]PDM_START_000100.COM
```

## Manual PDM initiation

If you set the logical CSI\_AUTOSTART to NO, you must initiate the PDM manually before you can access your PDM or any database your PDM services. To initiate the PDM manually, enter the following command at the DCL prompt:

```
$ @CSI_*
```

where \* represents the 6-digit UIC group number for a groupwide PDM, 000000 for a systemwide PDM, or the 1- to 8-character name of your multiple systemwide PDM (see “[Defining your operating environment](#)” on page 57).

Always initiate the PDM manually to test your setup before you rely on automatic initiation. During this test, it is best to use the PDM input parameter `CONSOLE=Y`. This causes the PDM to send error messages to the VMS operator console (see “[Entering parameters for the PDM input file](#)” on page 125). You can also check the CSIPDMLOG file to see any error messages.

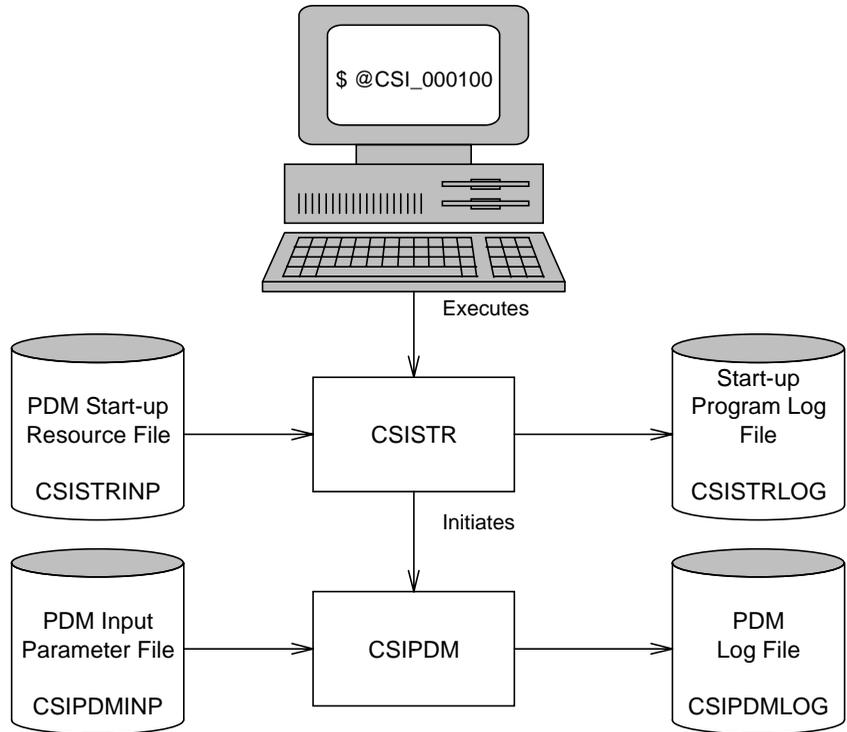


---

If an attempted automatic PDM start-up fails repeatedly, it will return an error status of NMAC and will not attempt any further restarts.

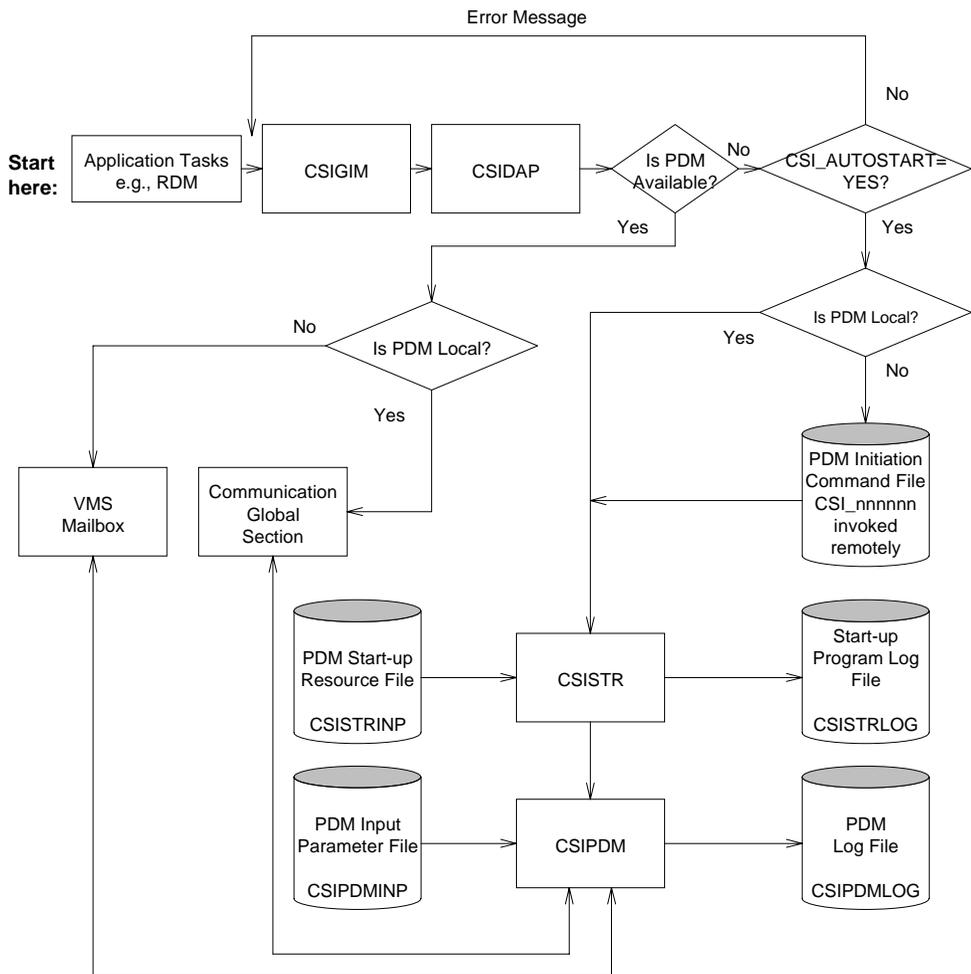
---

The following figure illustrates the process of manual PDM initiation using a groupwide PDM for UIC group 100:



## Automatic PDM initiation

During automatic PDM initiation (which you choose by setting the logical CSI\_AUTOSTART to YES), the first task to attempt database access is granted a PDM initiation lock. This prevents other tasks from starting the PDM on another machine. The task then communicates with the PDM by using CSIDAP either through the global section (if the task is on the same machine as the PDM) or through DECnet (if the task and PDM are running on different machines). When the PDM has started on one of the machines on the preferred machine list, the PDM initiation lock is released to allow other tasks to communicate with the PDM. The following figure illustrates the process of automatic PDM initiation:



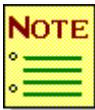
Automatic PDM initiation starts a PDM without any user intervention. It can be used on the first PDM start-up, and it can start the PDM after a failure. It needs no user intervention because it is initiated and controlled by the first application task that tries to access a database serviced by the PDM.

If the PDM fails, any attempt to use the failed PDM restarts the PDM. The PDM could fail for the following reasons:

- ◆ Hardware fault such as a machine check
- ◆ Software fault such as a PDM abort

The initiating task creates a VMS lock for the PDM, thus preventing other tasks from restarting the PDM for the same database. Automatic PDM initiation uses the preferred machine list to identify alternative machines on which the PDM can run. The preferred machine list is identified by the logical name `dbname_CSI_PDM_MACS`, which is defined in the `PDM_LOGICALS_*.COM` procedure (see “`PDM_LOGICALS_*.COM`” on page 87).

Once the PDM has started or restarted, the PDM initiation lock is released. After the PDM is restarted, all tasks that were signed on to the database are reconnected, and their current logical units of work are reset via the automatic database warm start. When the warm start is complete, the PDM returns a status of DRST (dynamic reset) to each task. The tasks must then reapply any modifications made since the last successful COMMIT point.



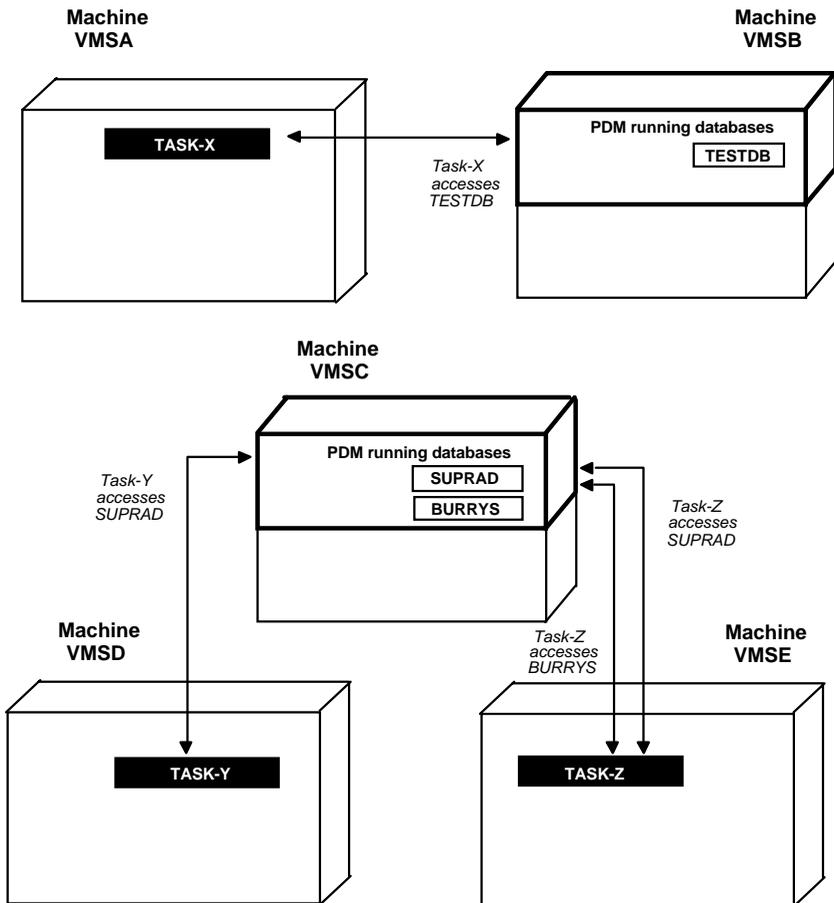
---

Once you have successfully connected to a PDM, CSIDAP will only communicate with that PDM. The preferred machine list will not be translated again.

---

**Examples of automatic PDM initiation.** The following figure illustrates an example cluster/network configuration consisting of five machines, VMSA through VMSE. Three databases, SUPRAD, TESTDB and BURRYS, are active. TESTDB is loaded in one copy of the PDM on machine VMSB, SUPRAD and BURRYS are loaded in another copy of the PDM on machine VMSC. The following tasks access these databases:

- ◆ TASK-X on machine VMSA
- ◆ TASK-Y on machine VMSD
- ◆ TASK-Z on machine VMSE



Both copies of the PDM share the same global section name. The preferred machine list logical definitions for the three databases are as follows:

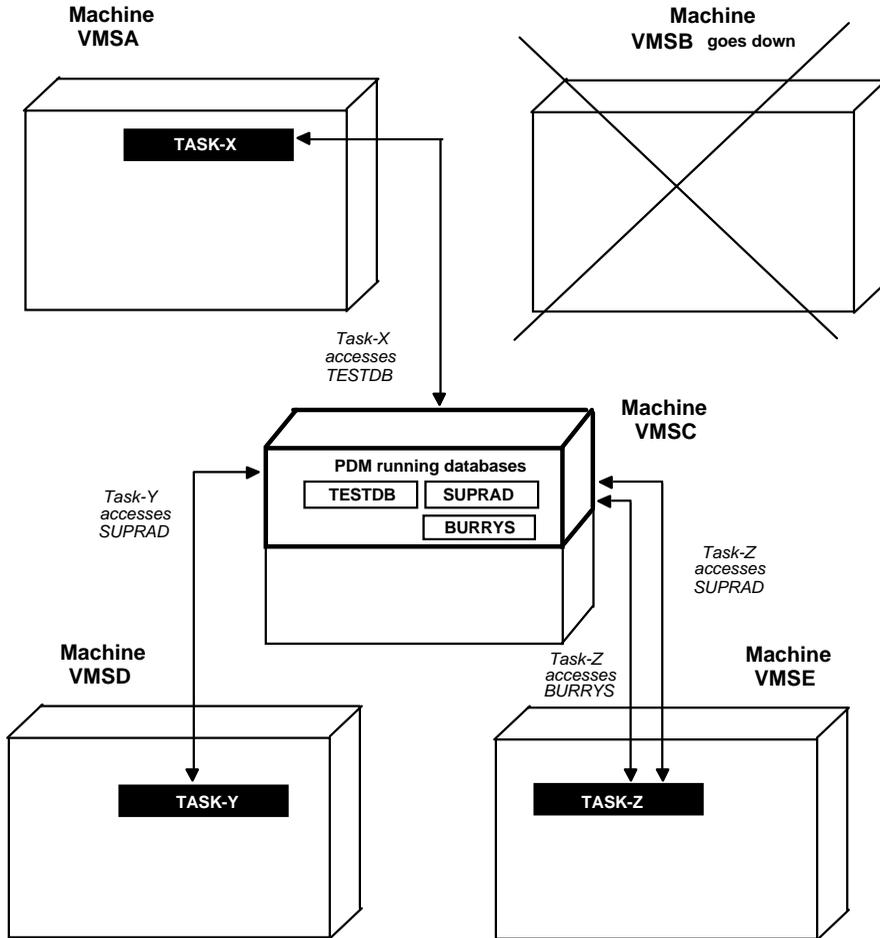
```
$ DEFINE TESTDB_CSI_PDM_MACS VMSB, VMSC
$ DEFINE SUPRAD_CSI_PDM_MACS VMSC, VMSA, VMSB, VMSD, VMSE
$ DEFINE BURRYS_CSI_PDM_MACS VMSC, VMSA, VMSB, VMSD, VMSE
```

All three databases have been initiated on the first machine on their preferred machine list. See “[PDM\\_LOGICALS\\_\\*.COM](#)” on page 87 for a description of how to set up a preferred machine list for each database.

Assume machine VMSB fails. The database TESTDB, running on machine VMSB, is no longer available. TASK-X, which is using database TESTDB from machine VMSA, will receive a message to say that its PDM has gone down. The first database access after the PDM failure will:

1. Look up the first machine on the machine list identified by the logical name TESTDB\_CSI\_PDM\_MACS. This machine is VMSB.
2. Discover that VMSB is unavailable.
3. Look up the next machine on the machine list identified by the logical name TESTDB\_CSI\_PDM\_MACS. This machine is VMSC.
4. Determine whether a PDM is running on VMSC.
5. Find the PDM that is already running on VMSC (servicing databases SUPRAD and BURRYS).
6. Set up communication with the PDM on VMSC through DECnet and start up database TESTDB.

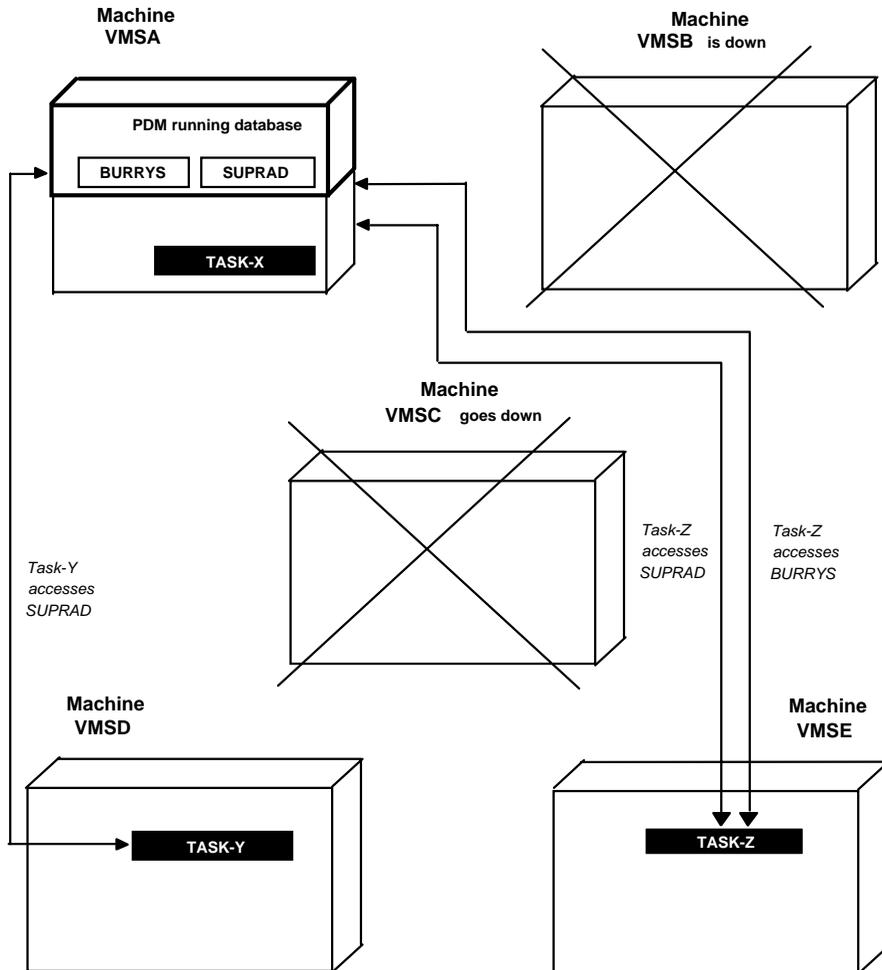
The following figure illustrates the cluster/network configuration after the PDM completes Step 6:



If machine VMSC now fails, all tasks connected to it are disconnected. The next task to access SUPRAD starts the PDM on VMSA (the next machine in the preferred machine list for SUPRAD). The PDM is then local for Task-X, also running on VMSA, but Task-X has no access to the PDM. Task-Y on VMSD and Task-Z on VMSE will connect to VMSA when they next access SUPRAD.

Any application task that tries to use TESTDB receives a NMAC status code because none of the machines on the preferred machine list are available. However, if either VMSB or VMSC becomes available before a task attempts to access TESTDB, the PDM for that database restarts there.

The next task to access BURRYS starts it in machine VMSA, the next machine on the BURRYS preferred machine list. The following figure illustrates this new machine configuration:



## Initiating the PDM on a network

Before you try to access the PDM from a network, you must determine whether the PDM is already running. You should run `CSI_FINDPDM` whenever you know the machine you are using is likely to be used as a PDM machine and might service networked tasks. SUPRA Server uses the network image `CSI_FINDPDM` to determine whether the PDM is already active on the local machine.

You can set `CSI_FINDPDM` to be activated automatically by the first task that needs to use the PDM. Alternatively, you can start `CSI_FINDPDM` as a detached process before you run any SUPRA Server application or at system initiation time. `CSI_FINDPDM` uses very few resources when it is not servicing any network requests.

The image `CSI_FINDPDM` must be installed with privileges when you invoke it directly or indirectly from a nonprivileged account. The command procedure `INSTALL_SUPRA24_PDM_IMAGES.COM` installs this image with the appropriate privileges.

`CSI_FINDPDM` has one parameter, `RQST_BUFQUO`, which determines the number of remote requests VMS can buffer in the Network Mailbox. The default value for this parameter is 12. This means a maximum of 12 remote network tasks can queue requests to `CSI_FINDPDM` simultaneously.

To manually invoke `CSI_FINDPDM` with the `RQST_BUFQUO` parameter, you need to create an input parameter file containing the `RQST_BUFQUO` parameter and a command procedure containing the run command and qualifiers.

A sample input parameter file is:

```
!
! Sample Input parameter file for the
! CSI_FINDPDM program :
!     File spec, is e.g., SUPRA_EXE:CSI_FINDPDM.INP
!
RQST_BUFQUO=6
!
!
```

A sample command file for starting up CSI\_FINDPDM as a detached process is:

```
$!
$!
$!     Sample Command file to start up CSI_FINDPDM
$!     File spec is, e.g., SUPRA_EXE:FINDPDM.COM
$!
$RUN/DETACHED -
  /INPUT  = SUPRA_EXE:CSI_FINDPDM.INP -
  /OUTPUT = SUPRA_EXE:CSI_FINDPDM.LOG -
  SUPRA_EXE:CSI_FINDPDM_2400.EXE
```

## Specifying a database

Once you have defined your PDM environment, you use the logical definition CSI\_SCHEMA to specify to the RDM which database to use. CSI\_SCHEMA is described below.

---

### CSI\_SCHEMA *dbname*

**Description**     *Required.* Database name used by all RDM applications.

**Format**            6-character database name

**Logical name table**  
                        Any

**Consideration** This logical is used in conjunction with CSI\_PREFIX.

## Using a database prefix

A database prefix allows you to distinguish between databases of the same name that are serviced by the same PDM. `CSI_PREFIX` must be defined only for those users who need to access that specific database. The definitions for your prefixed databases can be included in the `PDM_LOGICALS_*.COM` procedure for your PDM.

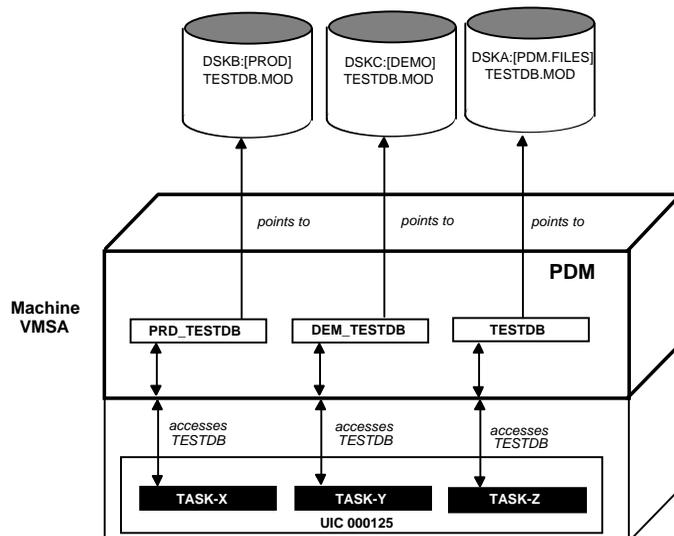


This section describes how to use a database prefix for groupwide databases serviced by a groupwide PDM. To apply a database prefix to a database serviced by a systemwide or multiple systemwide PDM, substitute `/SYSTEM` or `/TABLE=CSI_PDM_*` (where `*` is the name of your multiple systemwide PDM) for `/GROUP` in the prefixed logical definitions described below.

**Example of database prefixing.** Assume you have three databases called TESTDB, all used by tasks in UIC group 000125 on machine VM5A. Each database uses a different compiled database description (.MOD file). You can have only one group-level logical name TESTDB pointing to one compiled database description. The database prefix allows you to distinguish the other two by creating a group-level logical name `xxx_TESTDB` for each database, where `xxx` is a unique 1- to 3-character prefix identifier. For example:

```
$ DEFINE/GROUP xxx_TESTDB dev:[directory]dbname.MOD
```

The following figure illustrates this example configuration of three compiled database descriptions for the database TESTDB:



The logical definitions are given below:

```
$! Group-level logical names for the three TESTDB compiled
  database
$! descriptions.
$!
$DEFINE /GROUP   DIRECTORY   DSKA:[PDM.DIRECTORY]
$DEFINE /GROUP   DATAFILES  DSKA:[PDM.FILES]
$DEFINE /GROUP   PRD_DATAFILES DSKB:[PROD]
$DEFINE /GROUP   DEM_DATAFILES DSKC:[DEMO]
$!
$DEFINE /GROUP   TESTDB      DATAFILES:TESTDB.MOD
$DEFINE /GROUP   PRD_TESTDB  PRD_DATAFILES:TESTDB.MOD
$DEFINE /GROUP   DEM_TESTDB  DEM_DATAFILES:TESTDB.MOD
$!
$DEFINE /GROUP   TESTDB_CSI_PDM_MACS   VMSA,VMSB,VMSC
$DEFINE /GROUP   PRD_TESTDB_CSI_PDM_MACS VMSB,VMSA,VMSC
$DEFINE /GROUP   DEM_TESTDB_CSI_PDM_MACS VMSC,VMSA,VMSB
$!
```



---

You can use the 1- to 3-character prefix with the logical name `xxx_dbname_CSI_PDM_MACS` to define a different preferred machine list for each prefixed database.

---

TASK-X, TASK-Y, and TASK-Z in the preceding figure run in UIC group 000125 on machine VMSA. They all sign on to a TESTDB database; however, each task uses a different compiled database description (TESTDB.MOD). To identify the compiled database description they are accessing, tasks may or may not define the process-level logical name:

```
$ DEFINE CSI_PREFIX xxx
```

where `xxx` is the 1- to 3-character identifier prefixing the logical database name (as you defined in the `PDM_LOGICALS_*.COM` command procedure).

The values of CSI\_PREFIX for TASK-X, TASK-Y, and TASK-Z are defined as:

- ◆ TASK-X      \$DEFINE CSI\_PREFIX PRD
- ◆ TASK-Y      \$DEFINE CSI\_PREFIX DEM
- ◆ TASK-Z      Needs no logical definition for CSI\_PREFIX because TESTDB points to the default compiled database description.

Each task can define only one value for CSI\_PREFIX, and can redefine it to access a different compiled database description from its next application. Any task that omits the logical definition CSI\_PREFIX accesses the default database identified by the unprefix logical name TESTDB.

When a task that has the logical CSI\_PREFIX defined accesses a database, the PDM first attempts to translate the prefixed logical database name. Then, if no prefixed version of the database name logical exists, the PDM attempts to translate the unprefix database name to find the default compiled database description.



---

The PDM will attempt to apply the same prefix rules to other logicals used by the database to locate physical dataset files, log files, and index files.

---

### Considerations for using database prefixes

- ◆ If you use any logical names in the directory specification for the compiled database description file, make sure you add the prefix to the appropriate logical name. For example:

```
DEFINE/GROUP DATAFILES   DSKA:[PDM.FILES]
DEFINE/GROUP TESTDB      DATAFILES:TESTDB.MOD
DEFINE/GROUP DEM_TESTDB  DATAFILES:TESTDB.MOD
```

This accesses the same compiled database description for both the prefixed and the unprefixed versions of TESTDB. To ensure that the prefixed database accesses the appropriate file in this example, you should make the following logical definitions:

```
DEFINE/GROUP DATAFILES   DSKA:[PDM.FILES]
DEFINE/GROUP TESTDB      DATAFILES:TESTDB.MOD
DEFINE/GROUP DEM_DATAFILES DSKC:[DEMO]
DEFINE/GROUP DEM_TESTDB  DEM_DATAFILES:TESTDB.MOD
```

DATAFILES still points to DSKA:[PDM.FILES], but the prefixed database now uses the compiled database description held in DSKC:[DEMO].

- ◆ If one task loads a database without a prefix, a subsequent task can define a prefix for that database and sign on using the prefix. If this happens, the database is associated with the prefix and subsequent tasks must use the prefix when signing on. The first task, however, can still use the loaded database and sign off without problems.
- ◆ Once a database has been associated with a prefix, the prefix cannot be changed or removed without first unloading the database. See “[Unloading a database \(UNLOAD\)](#)” on page 169.

## Writing SUPRA Server PDM user exits

In addition to standard PDM operation, SUPRA Server provides user exits as entry points into the PDM. These entry points are a way to have the PDM run your own user-written, custom-designed code before and/or after a PDM function is executed. User exits are implemented as a single shareable image.

Implement user exit modules by following these steps:

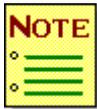
1. Create a sharable image with the following characteristics:

Entry point	Entry point name	Point of initiation
#1	CSD_UPDM_USEREX1	Immediately before the start of PDML function processing.
#2	CSD_UPDM_USEREX2	Immediately after completion of PDML function processing.

2. Before you execute your SUPRA Server PDM application program, define the logical name CSI\_USEREX to point to the image. For example, the following definition creates the logical CSI\_USEREX in the process logical name table:

```
DEFINE CSI_USEREX dev:[dir]:MY_USEREX.EXE
```

You can include this logical assignment in the logical name table according to the scope you wish to give the user exits, and in the appropriate command files (SUPRA\_COMS:PDM\_LOGICALS\_\*.COM).



If you do not define the logical CSI\_USEREX, the user exits are not called.

Each user exit supplies an identical parameter list consisting of the SUPRA Server PDML function call parameters. You can examine these parameters before and after PDM function processing. The call mechanism for each parameter is call-by-reference.

“[Example user exits](#)” on page 259 contains example user exits in COBOL and FORTRAN with examples of command files to compile and link the exits. The COBOL example uses both before and after exits. The FORTRAN example uses only the after exit.

A sample user exit is provided in SUPRA\_EXAMPLES. A vested version of the image is provided for use in Alpha environments.



---

For all Alpha versions and all versions 2.3 or higher on VAX, you must include a new additional parameter, Nargs or p0. See “[COBOL user exits](#)” on page 259 and “[FORTRAN user exit](#)” on page 267. This new parameter is a placeholder containing the number of arguments passed. It is unused but must be present.

---

# 3

## Defining your operating environment

SUPRA Server comes with a full set of components that help you configure your system in VMS. This chapter provides you with complete details of these components and describes the logicals and symbols used to define your operating environment.

When you install SUPRA Server, you install a set of command procedures that helps you define your operating environment and administer your SUPRA Server system. The following list explains the functionality of these command procedures:

- ◆ SUPRA\_MENU.COM: Performs many installation functions that help you administer your SUPRA Server environment. See “[Creating a PDM environment](#)” on page 58.
- ◆ INSTALL\_SUPRA 24\_PDM\_IMAGES.COM: Installs the necessary SUPRA Server images. This command procedure must be invoked before you attempt to access SUPRA Server.
- ◆ SUPRA\_SYMBOL.COM: Defines VMS symbols to facilitate using SUPRA Server components and administration utilities from the DCL command level.
- ◆ LOGICALS.COM: Creates general SUPRA Server logicals.

## Creating a PDM environment

Before you can use SUPRA Server, you must first define your operating environment. When you install SUPRA Server on your VMS system, one of the first steps you take is to create a PDM environment using the administration utility SUPRA\_MENU.COM. You continue to use SUPRA\_MENU.COM to define each PDM environment you need. SUPRA\_MENU.COM and similar utilities for other Cincom products use a common set of command procedures. This common set of command procedures uses a CLB file, which is placed in SYS\$COMMON:[CINCOM] at the time of your installation.



---

You can change the location of the common utilities from the default of SYS\$COMMON:[CINCOM] by defining the logical CSI\_PRODUCT\_FILE to point to the directory location of these common utilities.

---

When you create or maintain a PDM environment, SUPRA\_MENU.COM creates and maintains the following file in the directory identified by SUPRA\_COMS:

- ◆ SUPRA\_SYSTEM.COM: Designed for use at VMS system start-up to define all PDM environments.

SUPRA\_MENU.COM also creates and maintains the following files in the directory identified by SUPRA\_LIBRARY:

- ◆ PDM\_LOGICALS\_\*.COM: Defines all logicals that are specific to a particular PDM.
- ◆ PDM\_OPTIONS\_\*.INP: The input parameters for the specified PDM.
- ◆ PDM\_START\_\*.COM: The PDM start-up procedure.
- ◆ *pdmname*\_USER\_INIT.COM: Only created for a multiple systemwide PDM. Used to define the logical CSI\_SYSPDMID for the specified multiple systemwide PDM.

The wild card character \* represents the 6-character UIC group number for a groupwide PDM, 000000 for a systemwide PDM, or the 1- to 8-character name of a multiple systemwide PDM.

The SUPRA\_MENU.COM functions help you configure your SUPRA Server system during installation and administer your SUPRA Server system after installation. When you invoke SUPRA\_MENU.COM, you can:

- ◆ Move portions of the SUPRA Server system to different VMS directories (CUSTOM.COM).
- ◆ Deinstall SUPRA Server Known images (DEINSTALL\_SUPRA24\_PDM\_IMAGES.COM).
- ◆ Browse the introduction and tutorial of the SUPRA\_MENU.COM routines (EXPLAIN.COM).
- ◆ Install SUPRA Server Known images (INSTALL\_SUPRA24\_PDM\_IMAGES.COM).
- ◆ Initialize and configure a PDM environment (MAKE\_CLASS.COM).
- ◆ Upgrade an existing SUPRA Server 1.1.1 system to SUPRA Server 2.4 (UPGRADE.COM).



---

You can run the command procedures called by SUPRA\_MENU.COM stand-alone.

---

The procedure `INSTALL_SUPRA24_PDM_IMAGES.COM`, which is located in the directory identified by the logical definition `SUPRA_COMS`, should be included in your `SYSTARTUP` or executed before you attempt to access any PDM. This procedure installs various SUPRA Server images with the appropriate qualifiers and privileges. Some of the SUPRA Server images require privileges as shown in the following table:

Image name	Privilege	Reason needed
CSIPDM	DETACH	Used in creating other VMS processes, such as during System Log dumping.
	SYSLCK	Used to allocate systemwide VMS locks.
	SYSGBL	Allows the PDM to create systemwide global sections.
	WORLD	Used for multiprocess interoperation, such as Client/Server synchronization.
	EXQUOTA	Guarantee that critical information can always be logged.
	SYSNAM	Used to create systemwide logical definitions.
	SYSPRV	Access resources as if the process has a system UIC.
	ALTPRI	Allows the PDM to change its base execution priority as specified by the PDM Input Parameter File (CSIPDMINP) option <code>PRIORITY</code> .
	BYPASS	Allows access to any file. Enables the PDM to control Client task access to data based on VMS security and the PDM Input Parameter File (CSIPDMINP) parameters <code>UICCHECK</code> and <code>ACLCHECK</code> .
	CMEXEC	Allows the PDM to perform executive mode functions, such as manipulating RMS locks.

Image name	Privilege	Reason needed
CSISTR	CMKRNL	Allows the PDM Startup program to perform kernel mode functions, such as setting the detached PDM process UIC.
	DETACH	Used to create the detached PDM Server process.
	SYSPRV	Access resources as if the process has a system UIC.
	WORLD	To control the execution of the PDM Server owned by other UIC groups during PDM Initiation.
	GROUP	To control the execution of the PDM Server in the same UIC group during PDM Initiation.
	TMPMBX	Allows creation of temporary mailbox devices.
	NETMBX	Allows creation of network devices.
CSI_FINDPDM	SYSLCK	Used to allocate systemwide VMS locks.
	SYSNAM	Used to create systemwide logical definitions.
	SYSPRV	Access resources as if the process has a system UIC.
	WORLD	Used for multiprocess interoperation, such as Client/Server initiation.
	TMPMBX	Allows creation of temporary mailbox devices.
	NETMBX	Allows creation of network devices.

When you define your first PDM system, a procedure named SUPRA\_SYSTEM.COM is created in the directory identified by the logical definition SUPRA\_COMS. Subsequent PDM definitions will update this procedure. SUPRA\_SYSTEM.COM should also be included in your SYSTARTUP, or executed before you attempt to access any PDM.

SUPRA\_SYSTEM.COM will contain two lines for each active PDM environment. The first line executes LOGICALS.COM for the PDM (see [“LOGICALS.COM”](#) on page 63). The second line executes PDM\_LOGICALS\_\*.COM (see [“PDM\\_LOGICALS\\_\\*.COM”](#) on page 87).



You can use comments that are inserted each time you create or maintain a PDM environment to track changes to an existing PDM system. These comments indicate the level of the PDM (groupwide, systemwide, or multiple systemwide), the name of the process updating the file, and the date and time of the change.

## Defining logicals for your PDM environment

The PDM runs as a detached process. This detached process accesses the logical definitions through logical name tables. Each logical definition needed by SUPRA Server must be assigned to one of several types of logical name tables. Some logicals must reside in a specific table, while some can be defined in any table.

The administration utilities that are described throughout this chapter automatically define logicals at the level that is appropriate for your PDM. The common levels of logical name tables are described below.

- ◆ If your PDM is groupwide, the logical name table where your PDM logical definitions reside is the group table.
- ◆ If your PDM is systemwide, the logical name table where your PDM logical definitions reside is the system table.
- ◆ If your PDM is multiple systemwide, the logical name table where your PDM logical definitions reside is a specially defined shareable table. This table is named `CSI_PDM_pdmname`.

The scope of a logical name is defined by the logical name table(s) in which it appears. A logical name can appear in more than one logical name table.

A logical name that occurs in more than one logical name table can be associated with a different equivalence name in each logical name table. To avoid confusion, the system searches logical name tables in a particular order and uses the first definition found, regardless of any other occurrences. The default logical name table search sequence is `PROCESS`, `JOB`, `GROUP`, and `SYSTEM`. However, you can change this order to suit your requirements. See your VMS documentation for details.



---

Some logicals must be defined in the system table when using either a systemwide or a multiple systemwide PDM (see “LOGICALS.COM” below and “[Optional SUPRA Server logicals](#)” on page 279).

---

## LOGICALS.COM

The procedure LOGICALS.COM is located in the directory identified by the logical SUPRA\_COMS. This procedure defines general SUPRA Server logicals in a specified logical name table. The appropriate parameter specifying the logical name table is passed to LOGICALS.COM when it is executed by the SUPRA\_SYSTEM.COM procedure. If you choose to execute the LOGICALS.COM procedure without using SUPRA\_SYSTEM.COM, you must pass the appropriate parameter.

LOGICALS.COM can be called in one of the following ways:

- ◆ \$ @LOGICALS GROUP. Defines the logicals in the UIC group logical name table of the current process.
- ◆ \$ @LOGICALS SYSTEM. Defines the logicals in the system logical name table.
- ◆ \$ @LOGICALS LNM\$GROUP\_\*. Defines the logicals in the UIC group logical name table identified by \*.
- ◆ \$ @LOGICALS CSI\_PDM\_*pdmname*. Defines logicals in the appropriate logical name table used by a multiple systemwide PDM, and creates that logical name table if needed. Certain logicals are also defined in the system logical name table.

Some components of SUPRA Server will not function properly in a multiple systemwide PDM environment unless certain logicals are defined in the system logical name table. LOGICALS.COM will define these special logicals in the system logical name table instead of the CSI\_PDM\_ *pdmname* table. The reasons for this requirement are described below.

When CSI\_SYSPDMID is defined, CSISTR and CSIPDM obtain access to the CSI\_PDM\_ *pdmname* logical name table. However, the images identified by the logicals CSISTR and CSIPDM cannot be found in a multiple systemwide environment unless the definitions SUPRA\_EXE, CSISTR, and CSIPDM are in the system logical name table.

CSTUDSLF, the System Log Dump program, which also runs as a detached process, must also have access to the CSI\_PDM\_ *pdmname* table. To activate, the shareable images with which CSTUDSLF is linked must be available. These images are CSIGIM (through the logical SUPRAPDM) and CSIDAP. These logicals must also be placed in the system logical name table in a multiple systemwide PDM environment. This is done automatically by the LOGICALS.COM routine.

For systemwide and multiple systemwide PDMs, certain logicals may need to be defined with the /EXECUTIVE\_MODE option. This requirement comes from the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL. Both MANTIS and SPECTRA use this routine. To avoid problems with this RTL routine, LOGICALS.COM defines the affected logicals in /EXECUTIVE\_MODE for systemwide and multiple systemwide PDM environments.

The following table lists in alphabetical order the logicals that are created by the procedure LOGICALS.COM. Additional information on these logicals follows the table. Note that the variable *svclvl* in equivalence names is replaced with the service level of SUPRA Server.

Logical name	Equivalence name	Description
BGRN	SUPRA_EXE: CSUBGRN_ <i>svclvl</i> .EXE	Provides background DBA utilities.
CSDUSSERV	SUPRAPDM	Used by users who upgraded from ULTRA.
CSIDAP	SUPRA_EXE: CSIDAP_ <i>svclvl</i> .EXE	PDM interface module.
CSIDAP_DEB	SUPRA_EXE: CSIDAP_DEB_ <i>svclvl</i> .EXE	Used for testing by Cincom Support.
CSIDBA	SUPRA_EXE: CSIDBA_ <i>svclvl</i> .EXE	Provides DBA facilities.
CSI_DBA	SUPRA_REPORT	Directory containing Batch Directory Maintenance and Directory Views.
CSIDBAUTL	SUPRA_EXE: CSIDBAUTL_ <i>svclvl</i> .EXE	Provides online DBA utilities.
CSIDBPDM	SUPRA_EXE: CSIDBPDM_ <i>svclvl</i> .EXE	Database testing tool.
CSIDBVER	SUPRA_EXE: CSIDBVER_ <i>svclvl</i> .EXE	Database integrity verification utility.
CSIDDLLOAD	SUPRA_EXE: CSIDDLLOAD_ <i>svclvl</i> .EXE	Loads TOTAL- compatible DDL.
CSIDMPANL	SUPRA_EXE: CSIDMPANL_ <i>svclvl</i> .EXE	Provides a PDM dump analysis.
CSI_EXEC_DISPATCH	SUPRA_EXE:CSI_EXEC_ DISPATCH_ <i>svclvl</i> .EXE	Executive mode dispatcher.
CSI_EXEC_DISPATCH_DEB	SUPRA_EXE:CSI_EXEC_ DISPATCH_DEB_ <i>svclvl</i> .EXE	Used for testing by Cincom Support.

Logical name	Equivalence name	Description
CSIGIM	SUPRA_EXE: CSIGIM_svc/vl.EXE	General Interface Module.
CSIGIM_DEB	SUPRA_EXE: CSIGIM_DEB_svc/vl.EXE	Used for testing by Cincom Support.
CSIINDEX	SUPRA_COMS: CSIINDEX.CLD	Command definition file for the Index utility.
CSI_KERNEL_DISPATCH	SUPRA_EXE:CSI_KERNEL_DISPATCH_svc/vl.EXE	Kernel mode dispatcher.
CSI_KERNEL_DISPATCH_DEB	SUPRA_EXE:CSI_KERNEL_DISPATCH_DEB_svc/vl.EXE	Used for testing by Cincom Support.
CSILOCKS	SUPRA_AUXIL: CSILOCKS_svc/vl.EXE	Reports on VMS locks granted to a process.
CSIOAUTH	SUPRA_EXE: CSIOAUTH_svc/vl.EXE	CSIOPCOM command authorization program.
CSIOPCOM	SUPRA_EXE: CSIOPCOM_svc/vl.EXE	Alternative to VMS OPCOM Utility.
CSIPDM	SUPRA_EXE: CSIPDM_svc/vl.EXE	Detached PDM image.
CSIPDM_DEB	SUPRA_EXE: CSIPDM_DEB_svc/vl.EXE	Used for testing by Cincom Support.
CSIPDM_PATCH	<i>(security patch contents)</i>	Security patch logical for the PDM.
CSI_SMENU	SUPRA_EXE: CSI_SMENU_svc/vl.EXE	Menu for SUPRA Server components.
CSISTR	SUPRA_EXE: CSISTR_svc/vl.EXE	PDM startup routine.
CSMCHANGEDB	SUPRA_EXE: CSMCHANGEDB_svc/vl.EXE	Fast Utilities program.
CSMCOMBAT	SUPRA_EXE: CSMCOMBAT_svc/vl.EXE	Batch validate, compile, and print program.
CSTUDSLF	SUPRA_EXE: CSTUDSLF_svc/vl.EXE	System log dump program.
CSTUFMT	SUPRA_EXE: CSTUFMT_svc/vl.EXE	Stand-alone format program.

Logical name	Equivalence name	Description
CSTUFMTSHR	SUPRA_EXE: CSTUFMTSHR_svc/vl.EXE	Shareable format image.
CSTUIDX	SUPRA_EXE: CSTUIDX_svc/vl.EXE	Stand-alone index maintenance utility.
CSTUIDXSHR	SUPRA_EXE: CSTUIDXSHR_svc/vl.EXE	Shareable index maintenance image.
CSTURCV	SUPRA_EXE: CSTURCV_svc/vl.EXE	Stand-alone recovery program.
CSTURCVSHR	SUPRA_EXE: CSTURCVSHR_svc/vl.EXE	Shareable recovery image.
CSVBASIC	SUPRA_EXE: CSVBASIC_svc/vl.EXE	BASIC RDML preprocessor.
CSVCOBOL	SUPRA_EXE: CSVCOBOL_svc/vl.EXE	COBOL RDML preprocessor.
CSVDBAID	SUPRA_EXE: CSVDBAID_svc/vl.EXE	DBAID utility.
CSVFORTRA	SUPRA_EXE: CSVFORTRA_svc/vl.EXE	FORTRAN RDML preprocessor.
CSVGLOBAL	SUPRA_EXE: CSVGLOBAL_svc/vl.EXE	Provides global view creation.
CSVIPLVS	SUPRA_EXE: CSVIPLVS_svc/vl.EXE	SUPRA Server Relational Data Manager.
CSVIPLVS_DEB	SUPRA_EXE: CSVIPLVS_DEB_svc/vl.EXE	Used for testing by Cincom Support.
CSXSCREEN	SUPRA_EXE: CSXSCREEN_svc/vl.EXE	Screen handler.
DBAEDT	SUPRA_EXE:DBAEDT.EDT	Customized DBA EDT environment.
DBVER	SUPRA_COMS: CSIDBVER.CLD	Command verb for CSIDBVER utility.
RUNBASIC	SUPRA_COMS: RUNBASIC.COM	Command file to run the BASIC RDML preprocessor.

Logical name	Equivalence name	Description
RUNCOBOL	SUPRA_COMS:RUNCOBOL.COM	Command file to run the COBOL RDML preprocessor.
RUNDBAID	SUPRA_COMS:RUNDBAID.COM	Command file to run the DBAID utility.
RUNDIRM	SUPRA_COMS:RUNDIRM.COM	Command file to run the DIRM utility.
RUNFORTRA	SUPRA_COMS:RUNFORTRA.COM	Command file to run the FORTRAN RDML preprocessor.
SUPRA	SUPRA_EXE:CSI_SMENU_svc/vl.EXE	SUPRA Server main menu.
SUPRA_AUXIL	dev:[directory.SUPRA.PDM_24.AUXIL]	Directory containing unsupported auxiliary images.
SUPRA_BASE	dev:[directory.SUPRA.PDM_24]	Base SUPRA Server directory.
SUPRA_BURRYS	dev:[directory.SUPRA.PDM_24.BURRYS]	Directory containing the BURRYS database.
SUPRA_CLEAN_DICT	dev:[directory.SUPRA.PDM_24.CLEAN_DICT]	Directory containing an original copy of the Directory database SUPRAD.
SUPRA_CLEAN_EXE	dev:[directory.SUPRA.PDM_24.CLEAN_EXE]	(VAX environments only.) Directory containing original copies of SUPRA Server images. Used in applying maintenance.
SUPRA_COMS	dev:[directory.SUPRA.PDM_24.COMS]	Directory containing SUPRA Server command procedures.
SUPRA_DICT	dev:[directory.SUPRA.PDM_24.DICT]	Directory available for your use to hold a Directory database (SUPRAD).

Logical name	Equivalence name	Description
SUPRA_EXAMPLES	<i>dev:[directory.SUPRA.PDM_24.EXAMPLES]</i>	Directory containing example procedures and information.
SUPRA_EXE	<i>dev:[directory.SUPRA.PDM_24.EXE]</i>	Directory containing SUPRA Server images.
SUPRA_HELP	SUPRA_HELP: SUPRA_HELP.HLB	Identifies the help text library used by the DBA utility.
SUPRA_HELP	<i>dev:[directory.SUPRA.PDM_24.HELP]</i>	Directory containing the online help text library for the DBA utility.
SUPRA_LIBRARY	<i>dev:[directory.SUPRA.PDM_24.LIBRARY]</i>	Directory containing PDM setup files.
SUPRA_PATCH_WORK	<i>dev:[directory.SUPRA.PATCH_WORK]</i>	Directory used to apply maintenance.
SUPRAPDM	CSIGIM	Shareable PDM image linked with applications.
SUPRA_PDM_PATCHES	<i>dev:[directory.SUPRA.PDM_24.PATCHES]</i>	Directory containing security codes (VAX and Alpha).
SUPRA_REPORT	<i>dev:[directory.SUPRA.PDM_24.REPORT]</i>	Directory containing Batch Directory Maintenance and Directory Views.
SUPRA_TEST_EXE	<i>dev:[directory.SUPRA.PDM_24.TEST_EXE]</i>	Directory for test versions of the SUPRA Server images.
SYS\$ULTRA	CSI_DIRDB	For users who upgraded from ULTRA.
ULTRAPDM	SUPRAPDM	For users who upgraded from ULTRA.

---

## BGRN SUPRA\_EXE:CSUBGRN\_svc/vl.EXE

**Description**     *Required.* Provides background DBA utilities.

**Logical name table**  
Any

---

## CSDUSSERV SUPRAPDM

**Description**     *Optional.* Used by users who upgraded from ULTRA.

**Logical name table**  
Group, System

### Considerations

- ◆ For systemwide and multiple systemwide PDM environments, this logical is defined in the system logical name table. See the considerations for [CSTUDSLF](#).
  - ◆ For systemwide and multiple systemwide PDM environments, CSDUSSERV is defined in the system logical name table in executive mode. This prevents errors when you access SUPRA Server by using the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL while the calling image (such as MANTIS and SPECTRA) is installed with privileges.
- 

## CSIDAP SUPRA\_EXE:CSIDAP\_svc/vl.EXE

**Description**     *Required.* PDM interface module.

**Logical name table**  
Group, System

### Considerations

- ◆ For systemwide and multiple systemwide PDM environments, this logical is defined in the system logical name table. See the considerations for [CSTUDSLF](#).
- ◆ For systemwide and multiple systemwide PDM environments, CSIDAP is defined in the system logical name table in executive mode. This prevents errors when you access SUPRA Server by using the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL while the calling image (such as MANTIS and SPECTRA) is installed with privileges.

---

**CSIDAP\_DEB SUPRA\_EXE:CSIDAP\_DEB\_svc/vl.EXE**

**Description** *Optional.* Used for testing by Cincom Support.

**Logical name table**  
Any

**Consideration** This is a version of the CSIDAP image that was linked with debug. This debug image is used by Cincom Support for testing.

---

**CSIDBA SUPRA\_EXE:CSIDBA\_svc/vl.EXE**

**Description** *Required.* Provides DBA facilities.

**Logical name table**  
Any

**Consideration** The symbol DBA, defined in SUPRA\_COMS:SUPRA\_SYMBOL.COM, executes this image.

---

**CSI\_DBA SUPRA\_REPORT**

**Description** *Optional.* Specifies the directory for Batch Directory Maintenance and Directory Views.

**Default** SUPRA\_REPORT

**Format** VMS directory specification or logical pointing to a directory

**Logical name table**  
Any

---

**CSIDBAUTL SUPRA\_EXE:CSIDBAUTL\_svc/vl.EXE**

**Description** *Optional.* Provides online DBA utilities.

**Logical name table**  
Any

---

**CSIDBPDM SUPRA\_EXE: CSIDBPDM\_svc/vl.EXE**

**Description** *Optional.* Database testing tool.

**Logical name table**  
Any

**Consideration** Used to access databases directly using PDML.

---

### CSIDBVER SUPRA\_EXE:CSIDBVER\_svc/vl.EXE

**Description** *Optional.* Database integrity verification utility.

**Logical name table**  
Any

#### Considerations

- ◆ Verifies data set integrity (physical record locations and record pointers).
- ◆ Gathers physical statistics about data sets.
- ◆ Verifies data set physical characteristics with the information in the compiled database description file and the information on the SUPRAD Directory.
- ◆ For details of use, refer to the *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220.

---

### CSIDDLLOAD SUPRA\_EXE:CSIDDLLOAD\_svc/vl.EXE

**Description** *Optional.* Loads TOTAL-compatible DDL.

**Logical name table**  
Any

---

### CSIDMPANL SUPRA\_EXE:CSIDMPANL\_svc/vl.EXE

**Description** *Optional.* Provides a PDM dump analysis.

**Logical name table**  
Any

**Consideration** This utility can be used by Cincom Support to analyze the dump file produced by a PDM failure or by a PRINT OPCOM command.

---

**CSI\_EXEC\_DISPATCH SUPRA\_EXE:CSI\_EXEC\_DISPATCH\_svc/vl.EXE**

**Description**     *Required.* Executive mode dispatcher.

**Logical name table**

Group, System

**Considerations**

- ◆ For systemwide and multiple systemwide PDM environments, this logical is defined in the system logical name table. See the considerations for [CSTUDSLF](#).
- ◆ For systemwide and multiple systemwide PDM environments, CSI\_EXEC\_DISPATCH is defined in the system logical name table in executive mode. This prevents errors when you access SUPRA Server by using the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL while the calling image (such as MANTIS) is installed with privileges.

---

**CSI\_EXEC\_DISPATCH\_DEB SUPRA\_EXE:CSI\_EXEC\_DISPATCH\_DEB\_svc/vl.EXE**

**Description**     *Optional.* Used for testing by Cincom Support.

**Logical name table**

Any

**Consideration** This is a version of the CSIGIM image that was linked with debug. This debug image is used by Cincom Support for testing.

---

**CSIGIM SUPRA\_EXE:CSIGIM\_svc/vl.EXE**

**Description**     *Required.* General Interface Module.

**Logical name table**

Group, System

**Considerations**

- ◆ For systemwide and multiple systemwide PDM environments, this logical is defined in the system logical name table. See the considerations for [CSTUDSLF](#).
- ◆ For systemwide and multiple systemwide PDM environments, CSIGIM is defined in the system logical name table in executive mode. This prevents errors when you access SUPRA Server by using the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL while the calling image (such as MANTIS and SPECTRA) is installed with privileges.

---

### CSIGIM\_DEB SUPRA\_EXE:CSIGIM\_DEB\_svc/vl.EXE

**Description** *Optional.* Used for testing by Cincom Support.

**Logical name table**  
Any

**Consideration** This is a version of the CSIGIM image that was linked with debug. This debug image is used by Cincom Support for testing.

---

### CSIINDEX SUPRA\_COMS:CSIINDEX.CLD

**Description** *Optional.* Command definition file for the Index utility.

**Logical name table**  
Any

---

### CSI\_KERNEL\_DISPATCH SUPRA\_EXE:CSI\_KERNEL\_DISPATCH\_svc/vl.EXE

**Description** *Required.* Kernel mode dispatcher.

**Logical name table**  
Group, System

**Considerations**

- ◆ For systemwide and multiple systemwide PDM environments, this logical is defined in the system logical name table. See the considerations for [CSTUDSLF](#).
  - ◆ For systemwide and multiple systemwide PDM environments, CSI\_KERNEL\_DISPATCH is defined in the system logical name table in executive mode. This prevents errors when you access SUPRA Server by using the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL while the calling image (such as MANTIS) is installed with privileges.
- 

### CSI\_KERNEL\_DISPATCH\_DEB SUPRA\_EXE:CSI\_KERNEL\_DISPATCH\_DEB\_svc/vl.EXE

**Description** *Optional.* Used for testing by Cincom Support.

**Logical name table**  
Any

**Consideration** This is a version of the CSIGIM image that was linked with debug. This debug image is used by Cincom Support for testing.

---

---

**CSILOCKS SUPRA\_AUXIL:CSILOCKS\_svc/vl.EXE**

**Description** *Optional.* Reports on VMS locks granted to a process.

**Logical name table**  
Any

---

**CSIOAUTH SUPRA\_EXE:CSIOAUTH\_svc/vl.EXE**

**Description** *Required.* CSIOPCOM command authorization program.

**Logical name table**  
Any

---

**CSIOPCOM SUPRA\_EXE:CSIOPCOM\_svc/vl.EXE**

**Description** *Required.* Utility provided as an alternative to the VMS OPCOM Utility.

**Logical name table**  
Any

**Consideration** For details on using CSIOPCOM, see [“Communicating with the SUPRA Server PDM through CSIOPCOM”](#) on page 174.

---

**CSIPDM SUPRA\_EXE:CSIPDM\_svc/vl.EXE**

**Description** *Required.* PDM image executed by the detached PDM process.

**Logical name table**  
Group, System

**Consideration** For systemwide and multiple systemwide PDM environments, this logical is defined in the system logical name table. See the considerations for [CSTUDSLF](#).

---

**CSIPDM\_DEB SUPRA\_EXE:CSIPDM\_DEB\_svc/vl.EXE**

**Description** *Optional.* Used for testing by Cincom Support.

**Logical name table**  
Any

**Consideration** This is a version of the CSIPDM image that was linked with debug. This debug image is used by Cincom Support for testing.

---

### CSIPDM\_PATCH (*security patch contents*)

**Description** *Required.* Security patch for Alpha PDM.

**Logical name table**

Group, System, or CSI\_PDM\_pdmname

**Consideration** This logical is required for Alpha environments only.

---

### CSI\_SMENU SUPRA\_EXE: CSI\_SMENU\_svc/vl.EXE

**Description** *Optional.* Menu for SUPRA Server components.

**Logical name table**

Any

**Considerations**

- ◆ You might not have access to all of the components on the menu (see “[Selecting a SUPRA Server facility](#)” on page 27).
  - ◆ The symbol SUPRA, defined in SUPRA\_COMS:SUPRA\_SYMBOL.COM, runs this image.
- 

### CSISTR SUPRA\_EXE: CSISTR\_svc/vl.EXE

**Description** *Required.* Starts up the detached PDM process.

**Logical name table**

Group or System

**Consideration** In a multiple systemwide PDM environment, CSISTR gives itself access to the CSI\_PDM\_pdmname logical name table. However, in order for any process to automatically initiate the PDM (the logical CSI\_AUTOSTART is set to Y), or for various automatically initiated SUPRA Server utilities to initiate the PDM, this logical must be defined in the system logical name table in a multiple systemwide PDM environment.

---

### CSMCHANGEDB SUPRA\_EXE:CSMCHANGEDB\_svc/vl.EXE

**Description** *Optional.* Fast Utilities program.

**Logical name table**

Any

**Consideration** You must first set the CHANGEDB command using SUPRA\_EXE:CHANGEDB.CLD.

---

---

**CSMCOMBAT SUPRA\_EXE:CSMCOMBAT\_svc/vl.EXE**

**Description** *Required.* Batch validate, compile, and print program.

**Logical name table**

Group, System, or CSI\_PDM\_pdmname

**Consideration** You must first set the COMBAT command using SUPRA\_EXE:COMBAT.CLD.

---

**CSTUDSLF SUPRA\_EXE:CSTUDSLF\_svc/vl.EXE**

**Description** *Required.* System log dump program.

**Logical name table**

Group, System, or CSI\_PDM\_pdmname

**Considerations**

- ◆ This image is executed as a detached process.
- ◆ In a multiple systemwide PDM environment, CSTUDSLF grants itself access to the multiple systemwide shareable logical name table CSI\_PDM\_pdmname. However, for the CSTUDSLF image to activate, it must be able to find the shareable images with which it is linked. Therefore, LOGICALS.COM puts the definitions for these images, SUPRAPDM, CSIGIM, CSIDAP, and SUPRA\_EXE, in the system logical name table for systemwide and multiple systemwide PDM environments.

---

**CSTUFMT SUPRA\_EXE:CSTUFMT\_svc/vl.EXE**

**Description** *Required.* Stand-alone format program.

**Logical name table**

Any

---

**CSTUFMTSHR SUPRA\_EXE:CSTUFMTSHR\_svc/vl.EXE**

**Description** *Required.* Shareable format image.

**Logical name table**

Any

---

### **CSTUIDX SUPRA\_EXE: CSTUIDX\_svc/vl.EXE**

**Description** *Required.* Stand-alone index maintenance utility.

**Logical name table**  
Any

**Consideration** Provides access to the index utilities without using the DBA utility.

---

### **CSTUIDXSHR SUPRA\_EXE: CSTUIDXSHR\_svc/vl.EXE**

**Description** *Required.* Shareable index maintenance image.

**Logical name table**  
Any

**Consideration** Index utility used by the DBA utility and **CSTUIDX**.

---

### **CSTURCV SUPRA\_EXE: CSTURCV\_svc/vl.EXE**

**Description** *Required.* Stand-alone recovery program.

**Logical name table**  
Any

---

### **CSTURCVSHR SUPRA\_EXE: CSTURCVSHR\_svc/vl. EXE**

**Description** *Required.* Shareable recovery image.

**Logical name table**  
Any

---

### **CSVBASIC SUPRA\_EXE: CSVBASIC\_svc/vl.EXE**

**Description** *Optional.* The BASIC RDML preprocessor.

**Logical name table**  
Any

---

### **CSVCOBOL SUPRA\_EXE: CSVCOBOL\_svc/vl.EXE**

**Description** *Optional.* The COBOL RDML preprocessor.

**Logical name table**  
Any

---

**CSVDBAID SUPRA\_EXE:CSVDBAID\_svc/vl.EXE**

**Description**     *Required.* The DBAID utility.

**Logical name table**  
Any

---

**CSVFORTRA SUPRA\_EXE:CSVFORTRA\_svc/vl.EXE**

**Description**     *Optional.* The FORTRAN RDML preprocessor.

**Logical name table**  
Any

---

**CSVGLOBAL SUPRA\_EXE:CSVGLOBAL\_svc/vl.EXE**

**Description**     *Required.* The global view creation utility.

**Logical name table**  
Any

---

**CSVIPLVS SUPRA\_EXE:CSVIPLVS\_svc/vl.EXE**

**Description**     *Required.* The SUPRA Server Relational Data Manager.

**Logical name table**  
Group, System

**Consideration** For systemwide and multiple systemwide PDM environments, CSVIPLVS is defined in the system logical name table in executive mode. This prevents errors when you access SUPRA Server by using the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL while the calling image (such as MANTIS and SPECTRA) is installed with privileges.

---

**CSVIPLVS\_DEB SUPRA\_EXE:CSVIPLVS\_DEB\_svc/vl.EXE**

**Description**     *Optional.* Used for testing by Cincom Support.

**Logical name table**  
Any

**Consideration** This is a version of CSVIPLVS image that was linked with debug. This debug image is used by Cincom Support for testing.

---

### CSXSCREEN SUPRA\_EXE:CSXSCREEN\_svc/vl.EXE

**Description**     *Required.* Screen handler.

**Logical name table**  
Any

---

### DBAEDT SUPRA\_EXE:DBAEDT.EDT

**Description**     *Optional.* Customized DBA EDT environment for Logical View Maintenance.

**Logical name table**  
Any

**Considerations**

- ◆ You must have access to the EDT user environment to do Logical View Maintenance.
  - ◆ If you do not already use an EDT start-up file, you can point the logical EDTINI to SUPRA\_EXE:DBAEDT.EDT instead of using the logical DBAEDT.
  - ◆ If you already use the logical EDTINI to point to an EDT start-up file, use the DBAEDT logical to point to SUPRA\_EXE:DBAEDT.EDT or place the DBAEDT.EDT file in the VMS directory from which you run DBA.
  - ◆ You can change the name of the DBAEDT.EDT file as long as you use either the EDTINI or DBAEDT logical definition to point to the file.
- 

### DBVER SUPRA\_COMS:CSIDBVER.CLD

**Description**     *Optional.* Command verb for the **CSIDBVER** utility.

**Logical name table**  
Any

---

### RUNBASIC SUPRA\_COMS:RUNBASIC.COM

**Description**     *Optional.* Command file to run the optional BASIC RDML preprocessor.

**Logical name table**  
Any

**Consideration** Use this command procedure to generate standard BASIC code from an RDML BASIC program.

---

**RUNCOBOL SUPRA\_COMS:RUNCOBOL.COM**

**Description** *Optional.* Command file to run the optional COBOL RDML preprocessor.

**Logical name table**

Any

**Consideration** Use this command procedure to generate standard COBOL code from an RDML COBOL program.

---

**RUNDBAID SUPRA\_COMS:RUNDBAID.COM**

**Description** *Optional.* Command file to run DBAID.

**Logical name table**

Any

**Consideration** This command procedure defines logicals and then executes the DBAID utility.

---

**RUNDIRM SUPRA\_COMS:RUNDIRM.COM**

**Description** *Required.* Command file to run the DIRM utility.

**Logical name table**

Any

---

**RUNFORTRA SUPRA\_COMS:RUNFORTRA.COM**

**Description** *Optional.* Command file to run the optional FORTRAN RDML preprocessor.

**Logical name table**

Any

**Consideration** Use this command procedure to generate standard FORTRAN code from an RDML FORTRAN program.

---

**SUPRA SUPRA\_EXE:CSI\_SMENU\_svc/vl.EXE**

**Description** *Required.* SUPRA Server main menu.

**Default** SUPRA\_EXE:CSI\_SMENU\_svc/vl.EXE

**Logical name table**

Any

---

**SUPRA\_AUXIL dev:[*directory*.SUPRA.PDM\_24.AUXIL]**

**Description** *Optional.* Directory containing unsupported utilities used by Cincom Support.

**Logical name table**  
Any

---

**SUPRA\_BASE dev:[*directory*.SUPRA.PDM\_24]**

**Description** *Optional.* Base SUPRA Server directory.

**Logical name table**  
Any

---

**SUPRA\_BURRYS dev:[*directory*.SUPRA.PDM\_24.BURRYS]**

**Description** *Optional.* Directory containing the BURRYS database used in learning about SUPRA Server.

**Logical name table**  
Any

---

**SUPRA\_CLEAN\_DICT dev:[*directory*.SUPRA.PDM\_24.CLEAN\_DICT]**

**Description** *Required.* Directory containing an unused Directory database (SUPRAD).

**Logical name table**  
Any

**Consideration** Used when you create or maintain a PDM environment with SUPRA\_MENU.COM.

---

**SUPRA\_CLEAN\_EXE dev:[*directory*.SUPRA.PDM\_24.CLEAN\_EXE]**

**Description** *Required.* Directory containing original, unpatched SUPRA Server images.

**Logical name table**  
Any

**Consideration** Used in applying patch maintenance in VAX environments only.

---

**SUPRA\_COMS dev:[directory.SUPRA.PDM\_24.COMS]**

**Description** *Required.* Directory containing SUPRA Server command procedures.

**Logical name table**

Any

**Consideration** This directory contains utility procedures and PDM environment definition files.

---

**SUPRA\_DICT dev:[directory.SUPRA.PDM\_24.DICT]**

**Description** *Optional.* Provided for your use as a SUPRA Directory database (SUPRAD) location.

**Logical name table**

Any

---

**SUPRA\_EXAMPLES dev:[directory.SUPRA.PDM\_24.EXAMPLES]**

**Description** *Optional.* Directory containing example command procedures and information.

**Logical name table**

Any

---

**SUPRA\_EXE dev:[directory.SUPRA.PDM\_24.EXE]**

**Description** *Required.* Directory containing SUPRA Server images.

**Default** *dev:[directory.SUPRA.PDM\_24.EXE]*

**Format** VMS directory specification

**Logical name table**

Group or System

**Considerations**

- ◆ For systemwide and multiple systemwide PDM environments, this logical is defined in the system logical name table. See the considerations for **CSIGIM**, **CSISTR**, and **CSTUDSLF**.
- ◆ For systemwide and multiple systemwide PDM environments, SUPRA\_EXE is defined in the system logical name table in executive mode. This prevents errors when you access SUPRA Server by using the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL while the calling image (such as MANTIS and SPECTRA) is installed with privileges.

---

### SUPRA-HELP SUPRA\_HELP:SUPRA-HELP.HLB

**Description**     *Required.* Help library for SUPRA Server.

**Default**            SUPRA\_HELP:SUPRA-HELP.HLB

**Logical name table**  
                  Any

---

### SUPRA\_PATCH\_WORK dev:[*directory*.SUPRA.PATCH\_WORK]

**Description**     *Required.* Area provided for applying maintenance.

**Logical name table**  
                  Any

---

### SUPRAPDM CSIGIM

**Description**     *Required.* Used to link the SUPRA Server shareable image with applications.

**Logical name table**  
                  Group, System

#### Considerations

- ◆ For systemwide and multiple systemwide PDM environments, this logical is defined in the system logical name table. See the considerations for [CSTUDSLF](#).
  - ◆ For systemwide and multiple systemwide PDM environments, SUPRAPDM is defined in the system logical name table in executive mode. This prevents errors when you access SUPRA Server by using the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL while the calling image (such as MANTIS and SPECTRA) is installed with privileges.
- 

### SUPRA\_PDM\_PATCHES dev:[*directory*.SUPRA.PDM\_24.PATCHES]

**Description**     *Required.* Directory containing security codes (VAX and Alpha).

**Logical name table**  
                  Any

---

**SUPRA\_REPORT dev:[directory.SUPRA.PDM\_24.REPORT]**

**Description** *Optional.* Directory containing Batch Directory Maintenance and Directory Views.

**Logical name table**  
Any

---

**SUPRA\_TEST\_EXE dev:[directory.SUPRA.PDM\_24.TEST\_EXE]**

**Description** *Optional.* Points to the directory provided for testing SUPRA Server images.

**Format** VMS directory specification

**Logical name table**  
Group, System

**Considerations**

- ◆ For systemwide and multiple systemwide PDM environments, this logical is defined in the system logical name table. See the considerations for [CSTUDSLF](#).
- ◆ For systemwide and multiple systemwide PDM environments, SUPRA\_TEXT\_EXE is defined in the system logical name table in executive mode. This prevents errors when you access SUPRA Server by using the VMS Run Time Library routine LIB\$FIND\_IMAGE\_SYMBOL while the calling image (such as MANTIS and SPECTRA) is installed with privileges.

---

**SY\$ULTRA CSI\_DIRDB**

**Description** *Optional.* Provided for users who upgraded from ULTRA.

**Logical name table**  
Group, System, or CSI\_PDM\_pdmname

---

**ULTRAPDM SUPRAPDM**

**Description** *Optional.* Provided for users who upgraded from ULTRA.

**Logical name table**  
Group, System

**Consideration** Note the considerations for SUPRAPDM.

The LOGICALS.COM procedure does not automatically create the logicals listed in the following table. You can add these logicals manually. The requirements for entering each logical follows the table.

Logical name	Equivalence name	Description
CHANGEDB	SUPRA_COMS :CHANGEDB.CLD	Command definition file for Fast Utilities.
COMBAT	SUPRA_COMS :COMBAT.CLD	Command definition file for validate, compile, and print program.
CSVLINK	SUPRA_COMS :CSVLINK.COM	Command file to link RDM application programs.
RUNCSV	SUPRA_COMS :RUNCSV.COM	Command file to run an RDM application.
ULTRADBMS	SUPRAPDM	For users who upgraded from ULTRA 1.4.

---

**CHANGEDB SUPRA\_COMS :CHANGEDB.CLD**

**Description** *Optional.* Specifies a command definition file to run Fast Utilities.

**Logical name table** Any

---

**COMBAT SUPRA\_COMS :COMBAT.CLD**

**Description** *Optional.* Command definition file to run validate, compile, and print programs.

**Logical name table** Any

---

**CSVLINK SUPRA\_COMS :CSVLINK.COM**

**Description** *Optional.* Command file to link RDM application programs.

**Logical name table** Any

---

**RUNCSV SUPRA\_COMS :RUNCSV.COM**

**Description** *Optional.* Command file to run an RDM application.

**Logical name table** Any

---

**ULTRADBMS SUPRAPDM**

**Description** *Optional.* For users upgrading from ULTRA 1.4.

**Logical name table** Group or System

**Consideration** Note the considerations for [SUPRAPDM](#).

## **PDM\_LOGICALS\_\*.COM**

The procedure PDM\_LOGICALS\_\*.COM (where \* is the 6-character UIC group number for a groupwide PDM, 000000 for a systemwide PDM, or the 1- to 8-character name of a multiple systemwide PDM) defines logicals that are specific to a PDM environment.

PDM\_LOGICALS\_\*.COM can be invoked automatically from the SUPRA\_SYSTEM.COM procedure.

The logicals that can be defined in PDM\_LOGICALS\_\*.COM are broken into two groups: automatically included and manually added. The automatically included logicals are created for you when you create or maintain your PDM environment using SUPRA\_MENU.COM. The manually added logicals are those you can add to PDM\_LOGICALS\_\*.COM to fit your processing needs.

**Logical names automatically included in PDM\_LOGICALS\_\*.COM.**

The logicals listed in the following table are automatically included in PDM\_LOGICALS\_\*.COM. Some of the logicals are included as comments. To define these logicals, remove the comment character (!).

Additional information on the logicals in this table follows the table:

Logical name	Equivalence name	Description
CSI_*	<i>dev:[directory]</i> PDM_START_*.COM	PDM start-up command procedure.
CSI_AUTOSTART	YES or NO	Enable/disable automatic PDM start-up.
CSI_CONSOLE	OPER <i>n</i>	Operator console to which CSIDAP messages are sent.
[ <i>xxx</i> ]CSI_DIRDB	<i>dev:[directory]</i>	Location of Directory database (SUPRAD).
CSI_DMPANL	<i>dev:[directory]</i> CSI_DMP.ANL	Location of PDM crash dump file.
CSI_MRELAY	TRUE	Sends CSIDAP to a mailbox to be read by a user-written program.
CSIOPCOM_AUTH	<i>dev:[directory]</i> CSIOPCOM_AUTH.LIS	CSIOPCOM command authorization file.
CSIOPCOM_SNAPS	<i>dev:[directory]</i> CSIOPCOM_SNAPS.LIS	File for dumped CSIOPCOM screens.
*CSI_PDMID	<i>pdmname</i>	Identifies the name of the PDM for group and systemwide PDMs.
CSIPDMINP	<i>dev:[directory]</i> PDM_OPTIONS_*.INP	PDM input parameter file.
CSIPDMLOG	<i>dev:[directory]</i> CSIPDM.LOG	Log file for PDM messages.
CSISTRLOG	<i>dev:[directory]</i> CSISTR.LOG	Output file for PDM start-up program.
[ <i>xxx</i> ]dbname	<i>dev:[directory]</i> dbname.MOD	Points to compiled database description file(s).
[ <i>xxx</i> ]dbname_CSI_PDM_MACS	list of machines	Preferred machine list for a database.
[ <i>xxx</i> ]SUPRAD	<i>dev:[directory]</i> SUPRAD.MOD	Directory compiled database description.
[ <i>xxx</i> ]SUPRAD_CSI_PDM_MACS	mac1[,mac2...mac <i>n</i> ]	Preferred machine list.

\* Refers to the 6-digit group number for a groupwide PDM, 000000 for a systemwide PDM, or the translation of the logical CSI\_SYSPDMID (the PDM name) for a multiple systemwide PDM.

\* Only included for groupwide or systemwide PDM environments.

---

**CSI\_\* dev:[directory]PDM\_START\_\*.COM**

**Description**     *Required.* Points to the PDM start-up procedure, where \* is the 6-character UIC group number for a groupwide PDM, 6 zeros for a systemwide PDM, or the 1- to 8-character name of your multiple systemwide PDM.

**Default**             *dev:[directory]PDM\_START\_\*.COM*, where *dev:[directory]* is identified by the logical SUPRA\_LIBRARY.

**Format**                VMS file specification

**Logical name table**  
Group or System

---

**CSI\_AUTOSTART**

YES
NO

**Description**     *Optional.* Enables or disables the automatic PDM initiation.

**Default**             YES

**Logical name table**  
Any

**Considerations**

- ◆ You set this logical when you create or maintain your PDM environment.
- ◆ If automatic PDM initiation is disabled, you must manually start the PDM before any databases serviced by that PDM can be accessed on that machine.
- ◆ If you have originally defined this logical name as NO and wish to set it's value to YES you will also need to define the logical name CSI\_\* in the SUPRA\_LIBRARY:PDM\_LOGICALS\_\*.COM file. The logical name must be at the same level and in the same table as the rest of the logical names in this procedure. The logical CSI\_\* points to the SUPRA\_LIBRARY:PDM\_START\_\*.COM file. ( \* is the 6-digit group UIC, the 000000 system UIC, or the translation of the CSI\_SYSPDMID logical name.)

---

## CSI\_CONSOLE OPER*n*

**Description** *Optional.* Specifies the one operator console to which CSIDAP sends messages.

**Format** Any valid operator console number (OPER1 - OPER12)

**Logical name table**  
Group, System, or CSI\_PDM\_*pdmname*

---

## [*xxx\_*]CSI\_DIRDB *dev*:**[*directory*]**

**Description** *Required.* Location of the Directory database SUPRAD.

**Format** *xxx\_* 1–3 alphanumeric characters followed by “\_” (the optional database prefix)

*dev*:**[*directory*]** VMS directory specification

**Logical name table**  
Group, System, or CSI\_PDM\_*pdmname*

---

## CSI\_DMPANL *dev*:**[*directory*]**CSI\_DMP.ANL

**Description** *Optional.* Location of PDM crash dump file.

**Default** *dev*:**[*directory*]**CSI\_DMP.ANL, where *dev*:**[*directory*]** is identified by the logical CSI\_DIRDB.

**Format** VMS file specification

**Logical name table**  
Group, System, or CSI\_PDM\_*pdmname*

### Considerations

- ◆ If you do not define this logical, a file with the name CSI\_DMPANL will be created when the PDM fails. It will be created in the VMS directory in which the PDM was started.
- ◆ The SUPRA Server PDM operator command PRINT will also use this logical when creating a dump of an active database.
- ◆ This file may be analyzed by Cincom Support personnel.

---

**CSI\_MRELAY TRUE**

**Description** *Optional.* Sends PDM messages to a mailbox to be read by a user-written program.

**Logical name table**

Any

**Considerations**

- ◆ This logical is defined at your request when you create a PDM environment.
- ◆ See “[Automating operator communication](#)” on page 183 for information about creating a mailbox-reading program.
- ◆ This logical definition also appears in the manually added logicals section of PDM\_LOGICALS\_\*.COM.

---

**CSIOPCOM\_AUTH dev:[*directory*]CSIOPCOM\_AUTH.LIS**

**Description** *Optional.* CSIOPCOM command authorization file.

**Default** *dev:[directory]CSIOPCOM\_AUTH.LIS*, where *dev:[directory]* is identified by the logical CSI\_DIRDB.

**Format** VMS file specification

**Logical name table**

Any

**Consideration** By default, users with SYSPRV and OPER privileges have access to all PDM operator commands through CSIOPCOM if there is no logical definition for CSIOPCOM\_AUTH. Users with lower privileges without a CSIOPCOM command authorization file can only use the DISPLAY command.

---

**CSIOPCOM\_SNAPS dev:[*directory*]CSIOPCOM\_SNAPS.LIS**

**Description** *Optional.* File for dumped CSIOPCOM screens.

**Format** VMS file specification

**Logical name table**

Any

**Consideration** See “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174 for information on CSIOPCOM.

## CSI\_PDMID *pdmname*

**Description** *Required* for groupwide or systemwide PDMs. Specify the logical in the group or system table, depending on whether the PDM should run groupwide or systemwide, respectively.

**Format** *pdmname* is the 1–8 character name identifying the PDM.

### Logical name table

Group, System

### Considerations

- ◆ If this logical is in both the group and the system tables, users in that group may only access the groupwide PDM or any available multiple systemwide PDM, but not a systemwide PDM. See [“Understanding the Physical Data Manager \(PDM\)”](#) on page 31.
- ◆ CSI\_SYSPDMID supersedes CSI\_PDMID. See [“LOGICALS.COM”](#) on page 63.

---

## CSIPDMINP *dev:[directory]PDM\_OPTIONS\_\*.INP*

**Description** *Optional*. PDM input file.

**Default** *dev:[directory]PDM\_OPTIONS\_\*.INP*, where *dev:[directory]* is identified by the logical SUPRA\_LIBRARY.

**Format** VMS file specification

### Logical name table

Group, System, or CSI\_PDM\_*pdmname*

**Consideration** Describes the PDM input parameters (see [“Entering parameters for the PDM input file”](#) on page 125).

---

**CSIPDMLOG dev:[directory]CSIPDM.LOG**

**Description** *Optional.* Identifies the output file to which the PDM will send its messages.

**Format** VMS file specification

**Logical name table**

Group, System, or CSI\_PDM\_pdmname

**Consideration** If you do not define the logical name CSIPDMLOG, the PDM will create a file called CSIPDMLOG in the VMS directory in which the PDM was started.

---

**CSISTRLOG dev:[directory]CSISTR.LOG**

**Description** *Required.* Identifies the output file to which the PDM start-up image CSISTR will send its messages.

**Default** dev:[directory]CSISTR.LOG, where dev:[directory] is identified by the logical CSI\_DIRDB.

**Format** VMS file specification

**Logical name table**

Group, System, or CSI\_PDM\_pdmname

**Considerations**

- ◆ The database access program CSIDAP uses this log file on a local start-up. Therefore, the logical name CSISTRLOG must point to a valid file specification.
- ◆ The file this logical name points to should be unique for each PDM you wish to start.

**[xxx\_]dbname dev:[directory]dbname.MOD**

<b>Description</b>	<i>Required.</i> Points to a compiled database description file.	
<b>Format</b>	<i>xxx_</i>	1–3 alphanumeric characters followed by “_” (the optional database prefix)
	<i>dbname</i>	6 alphanumeric characters
	<i>dev:[directory]dbname.MOD</i>	File specification for compiled database description file

**Logical name table**

Group, System, or CSI\_PDM\_ *pdmname*

**Considerations**

- ◆ Define a separate logical name for each database you want to load.
- ◆ You must define this logical name before you can format the physical files for the database.
- ◆ This logical must exist before you can use the database.  
SUPRA\_COMS: SUPRA\_SYSTEM.COM can be executed at system start-up to set up the environment for all selected PDMs.
- ◆ The 1 to 3-character prefix allows you to differentiate between databases of the same name that are serviced by the same PDM. See “[Specifying a database](#)” on page 50 for a description of how to use a database prefix.

**[xxx\_]dbname\_CSI\_PDM\_MACS mac1[,mac2...macn]**

<b>Description</b>	<i>Required.</i> A list of machines, in descending order of preference, on which the specified PDM for this database can run (the preferred machine list).	
<b>Format</b>	<i>xxx_</i>	1–3 alphanumeric characters followed by “_” (the optional database prefix)
	<i>dbname</i>	6 alphanumeric characters
	<i>mac1</i>	1–6 alphanumeric character node name
	<i>[,mac2...macn]</i>	List of 1–6 alphanumeric character node names

**Logical name table**

Any

**Considerations**

- ◆ *mac1* is the node name of the first choice machine on which the PDM can run.
- ◆ *[,mac2...macn]* are the node names of any other machines on which the PDM can run, in descending order of preference.
- ◆ Create a preferred machine list for each database.
- ◆ If the preferred machine list contains more than one machine, duplicate the logical assignments on each machine that might execute an application.
- ◆ The logical name *[xxx\_]dbname\_CSI\_PDM\_MACS* must be accessible to all applications that can use the specified database. By placing this logical name in the group logical name table, you restrict access to applications in the same group as the initiating task. By placing this logical name in the system logical name table, you allow access to all applications on the system. By placing this logical name in the *CSI\_PDM\_pdmname* logical name table, you allow access to all applications on the system where *CSI\_SYSPDMID* is the same as *pdmname*.
- ◆ This logical must exist before you can use the database. SUPRA\_COMS: SUPRA\_SYSTEM.COM can be executed at system start-up to set up the environment for all selected PDMs.
- ◆ The PDM can load a database on an alternative machine if the physical files it uses are accessible from that machine, for example, if they reside on clustered disks, dual-ported disks, or disks local to the active machine.
- ◆ The 1 to 3-character prefix allows you to specify a different preferred machine list for each prefixed database. See “Using a database prefix” on page 51 for a description of how to use a database prefix.

**[xxx\_]SUPRAD dev:[directory]SUPRAD.MOD**

- Description**    *Required.* The Directory compiled database description logical.
- Default**        *dev:[directory]*    The same as that identified by the logical CSI\_DIRDB
- Format**        *xxx\_* 1–3 alphanumeric characters followed by “\_” (the optional database prefix)
- dev:[directory]*    VMS directory specification

**Logical name table**

Group, System, or CSI\_PDM\_ *pdmname*

**[xxx\_]SUPRAD\_CSI\_PDM\_MACS mac1[,mac2...macn]**

**Description**     *Required.* A list of machines, in descending order of preference, on which the specified PDM for this database can be run (the preferred machine list).

**Format**

<i>xxx_</i>	1–3 alphanumeric characters followed by “_” (the optional database prefix)
<i>mac1</i>	1-6 alphanumeric character node name
<i>[,mac2...macn]</i>	List of 1-6 alphanumeric character node names

**Logical name table**

Any

**Considerations**

- ◆ mac1 is the node name of the first choice machine on which the PDM can run.
- ◆ [,mac2...macn] are the node names of any other machines on which the PDM can run, in descending order of preference.
- ◆ Create a preferred machine list for each SUPRAD database.
- ◆ If the preferred machine list contains more than one machine, duplicate the logical assignments on each machine that might execute an application.
- ◆ The logical name [xxx\_]SUPRAD\_CSI\_PDM\_MACS must be accessible to all applications that can use the specified SUPRAD database. By placing this logical name in the group logical name table, you restrict access to applications in the same group as the initiating task. By placing this logical name in the system logical name table, you allow access to all applications on the system. By placing this logical name in the CSI\_PDM\_pdmname logical name table, you allow access to all applications on the system where CSI\_SYSPDMID is the same as pdmname.
- ◆ This logical must exist before you can use the database. SUPRA\_COMS:SUPRA\_SYSTEM.COM can be executed at system start-up to set up the environment for all selected PDMs.
- ◆ The PDM can load a database on an alternative machine if the physical files it uses are accessible from that machine, for example, if they reside on clustered disks, dual-ported disks, or disks local to the active machine.
- ◆ The 1 to 3-character prefix allows you to specify a different preferred machine list for each prefixed SUPRAD database. See "Using a database prefix" on page 54 for a description of how to use a database prefix.

**Logical names manually added to PDM\_LOGICALS\_\*.COM.**

Depending on your requirements, you should include the logical definitions shown in the following table in the PDM\_LOGICALS\_\*.COM procedure for your PDM, in LOGICALS.COM, or elsewhere in your system start-up routines. Additional information on these logicals follows the table.



You can define CSI\_MRELAY in PDM\_LOGICALS\_\*.COM by making that selection when you define your PDM system through SUPRA\_MENU.COM.

Logical name	Equivalence name	Description
BATCH_GLOBAL_INPUT	<i>dev.[directory] filename.ext</i>	Input text file used to create a global view file.
CSI_ALLOW_DUP_RECORD_CODE	TRUE	Allow duplicate record code in element list.
CSI_FINDPDM	<i>dev.[directory] CSI_FINDPDM.COM</i>	Command file to enable automatic CSI_FINDPDM start-up.
CSI_MRELAY2	TRUE	Sends CSIDAP messages to a mailbox to be read by a user-written program.
CSI_NODIRECTORY	TRUE	Prevents RDM from signing on to the SUPRA Server Directory.
CSI_PREFIX	1 to 3-character prefix	Specifies the prefix used to distinguish databases of the same name used in the same group or system.
CSI_REINIT_ON_SINON	TRUE	Retranslate <i>csi_prefix</i> at each <i>sinon</i> .
CSI_RMS_RU_ON	TRUE	Enables RMS Journaling.
CSISTRINP	<i>dev.[directory] CSISTR.INP</i>	PDM start-up resource file.
CSI_USEREX	<i>filename.EXE</i>	PDM user exit.
CSI_VAL_EXIT	<i>dev.[directory] image-name.EXE</i>	Identifies the Shareable image you write to do RDM Domain Validation.
CSI_WILD_EN	x	Equal or next wild card character.
CSI_WILD_EQ	x	Equal only wild card character.
CSUBGRN_CONTINUE_ON_ERROR	TRUE	Allow unload/reload to continue regardless of error count.
DBAID_HELP_NOSPAWN	TRUE	Allow <i>dbaid</i> to use <i>lbr\$output_help</i> .
DUMPSLF_ [xxx_]dbname	<i>dev.[directory]filename .ext</i>	Location of the system log dump input file.
GVSHEMA	<i>filename.GBL</i>	Specifies the global view file.
GVSHEMA_SYS	TRUE	Indicates that the global view file is to be loaded systemwide.

---

**BATCH\_GLOBAL\_INPUT dev:[directory]filename.ext**

**Description**     *Optional.* Input text file used to create a global view file.

**Format**            VMS file specification

**Logical name table**  
Any

**Consideration** Refer to the *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220, for information on how to create global view files.

---

**CSI\_ALLOW\_DUP\_RECORD\_CODE TRUE**

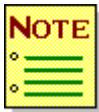
**Description**     *Optional.* Permit a record code to be repeated in an element list as was needed in ULTRA.

**Default**            FALSE

**Logical name table**  
Group logical name table for a groupwide database, or system logical name table for a systemwide database.

**Considerations**

- ◆ This feature on VAX was previously an optional patch and is now implemented as an optional logical name. It is intended for use in a special situation and is not intended for general use.



---

Cincom does not recommend the use of this logical name. If you are uncertain whether you need this feature or not, then you probably do not need it.

---

- ◆ CSI\_ALLOW\_DUP\_RECORD\_CODE replaces the previous logical name CSI\_DUPLICATE\_CODE which was used in FC 950516.
- ◆ This logical name is \*only\* checked at pdm startup initialization and therefore must be defined prior to pdm startup. Defining this logical name after the pdm server process has started will have no effect.
- ◆ Replaces patches 907753, 927548, 950516, and 950641 used in previous VAX releases.

---

## CSI\_FINDPDM dev:[directory]CSI\_FINDPDM.COM

**Description** *Optional.* Command file used to automatically start a task that determines if a PDM is running on a networked node.

### Logical name table

Group, System

### Considerations

- ◆ Even if this logical is defined, this is only used for remote PDM access in a networked environment.
- ◆ With this logical defined, the command procedure it points to is activated automatically the first time an attempt is made to access a remote networked PDM.
- ◆ See “[Initiating the PDM on a network](#)” on page 48 for details on the contents of the CSI\_FINDPDM.COM file.
- ◆ In a networked environment, this logical should be defined on any node in which the PDM may be run.

---

## CSI\_MRELAY TRUE

**Description** *Optional.* Sends PDM messages to a mailbox to be read by a user-written program.

### Logical name table

Any

### Considerations

- ◆ This logical can be defined at your request when you create a PDM environment.
- ◆ This logical definition also appears with the automatically included logicals section of PDM\_LOGICALS\_\*.COM.
- ◆ See “[Automating operator communication](#)” on page 183 and “[Example mailbox-reading program](#)” on page 275 for information about creating a mailbox-reading program.

---

**CSI\_NODIRECTORY TRUE**

**Description**     *Optional.* Prevents RDM from signing on to the SUPRA Server Directory.

**Default**            FALSE

**Logical name table**  
Any

**Considerations**

- ◆ Any value other than TRUE is considered FALSE.
- ◆ To use your database through the RDM without the SUPRA Server Directory, you must use a Global View file identified by the logical **GVSHEMA**. The Global View file then provides the user access authority, passwords, and preopened copies of the views.
- ◆ Any view that is not included in the Global View file will not be available for use when CSI\_NODIRECTORY is defined to TRUE.
- ◆ Refer to the *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220, for information on how to create global view files.
- ◆ When CSI\_NODIRECTORY is defined to TRUE, you cannot precompile any RDML applications.

## CSI\_PREFIX xxx

**Description** *Optional.* Prefix used to distinguish databases of the same name serviced by the same PDM.

**Format** 1–3 characters

**Logical name table**  
Any

**Consideration** All logicals for the database should be prefixed. If the PDM does not find prefixed database logicals, any unprefixed database logicals are used. See “[Specifying a database](#)” on page 50.

---

## CSI\_REINIT\_ON\_SINON TRUE

**Description** *Optional.* Permit user to change CSI\_PREFIX after SINOF but before image termination.

**Default** FALSE

**Logical name table**  
Any

### Considerations

- ◆ Logical name CSI\_REINIT\_ON\_SINON can be defined as TRUE, forcing the logical name CSI\_PREFIX to be translated at each SINON. This allows an application to SINOF, redefine [CSI\\_PREFIX](#), then SINON again, using a different database.
- ◆ Replaces patches 927252 and 932856 used in previous VAX releases.

---

## CSI\_RMS\_RU\_ON TRUE

**Description**     *Optional.* Enables RMS Recovery Unit (RU) Journaling on RDM RMS data sets.

### Logical name table

Any

### Considerations

- ◆ RMS data sets are maintained by the RDM, not the PDM. RMS Recovery Unit Journaling records all updates to RMS data sets in a way similar to task logging for PDM data sets.
- ◆ After a program or system failure, RU Journaling automatically recovers each RMS file to the last successful commit point.
- ◆ RU Journaling does not support network operations. Any attempt to access an RMS file marked for RU Journaling from a remote node is rejected.
- ◆ For details on how to implement Recovery Unit Journaling, refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260.

---

## CSISTRINP *dev:[directory]CSISTR.INP*

**Description**     *Required.* Identifies the start-up resource input file.

### Logical name table

Group, System, or CSI\_PDM\_ *pdmname*

### Considerations

- ◆ A template of this file is supplied for you with the name CSISTR.CSI in the directory identified by the logical SUPRA\_LIBRARY. You can also create the input file using a VMS editor.
- ◆ A comment line with a default definition is provided in PDM\_LOGICALS\_\*.COM as *dev:[directory]CSISTR.INP*, where *dev:[directory]* is the same as identified by the logical CSI\_DIRDB.
- ◆ The file to which this logical name points should be unique for each PDM you wish to start.
- ◆ You can specify the PDM UIC owner and VMS process quotas and limits in the start-up resource file (see “[Entering input parameters](#)” on page 113).

---

### CSI\_USEREX filename.EXE

**Description** *Optional.* PDM user exit image.

**Format** VMS file specification

**Logical name table**  
Any

**Consideration** See “[Writing SUPRA Server PDM user exits](#)” on page 55 and “[Example user exits](#)” on page 259 for more information on PDM user exits.

---

### CSI\_VAL\_EXIT

**Description** *Optional.* RDM Domain Validation image.

**Format** VMS file specification

**Logical name table**  
Any

**Consideration** Refer to the [SUPRA Server PDM RDM Administration Guide \(VMS\)](#), P25-8220, for more information.

---

### CSI\_WILD\_EN x

**Description** *Optional.* Wild card character used to specify that an equal or next RDM generic read return the value.

**Default** \*

**Format** Any character

**Logical name table**  
Any

**Consideration** The RDM uses the value of this logical as the wild card character. This wild card character specifies an equal or next match during generic reads by using a secondary key on character data. For additional information, refer to the [SUPRA Server PDM RDM Administration Guide \(VMS\)](#), P25-8220.

---

**CSI\_WILD\_EQ x**

**Description** *Optional.* Equal only wild card character for RDM generic reads.

**Default** =

**Format** Any character

**Logical name table**

Any

**Consideration** When doing generic reads using a secondary key on character data, the RDM will use the value of this logical to specify the wild card character for an equal or next match. For additional information, refer to the [SUPRA Server PDM RDM Administration Guide \(VMS\)](#), P25-8220.

---

**CSUBGRN\_CONTINUE\_ON\_ERROR TRUE**

**Description** *Optional.* Allow unlimited errors with utilities.

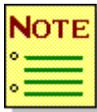
**Default** FALSE

**Logical name table**

Any

**Considerations**

- ◆ Logical name CSUBGRN\_CONTINUE\_ON\_ERROR can be defined as TRUE to allow CSUBGRN to continue executing regardless of the number of errors encountered.



---

This could result in an infinite loop.

---

- ◆ Replaces patches 898554, 898572, 927202, 928549, and 930391 used in previous VAX releases.

## DBAID\_HELP\_NOSPAWN TRUE

**Description**     *Optional.* DBAID help doesn't need to spawn.

**Default**            FALSE

### Logical name table

Any

### Considerations

- ◆ The HELP command within CSVDBAID spawns a subprocess to do DCL \$HELP. This can be a performance problem on some systems. This implements an optional logical name to change the HELP command to call LBR\$OUTPUT\_HELP routine which runs in the same VMS process as CSVDBAID. This occurs when the logical name DBAID\_HELP\_NOSPAWN is defined as TRUE.
- ◆ Advantage—The HELP command gives better performance because the overhead of spawning a subprocess is eliminated.
- ◆ Disadvantage—The behavior of the HELP command is slightly modified to no longer prompt “Press RETURN to continue...” between pages and to require a <cr> after a ? is entered.



---

This new behavior is the same as the other DEC layered products internal help.

---

---

**DUMPSLF\_[xxx\_]dbname dev:[directory]filename.ext**

**Description**     *Optional.* Identifies the location of the system log dump input file.

**Format**            *xxx\_*                    1–3 alphanumeric characters followed by “\_” (the optional database prefix)

*dbname*                6-character name of your database

*dev:[directory]*    VMS directory specification

*filename.ext*        VMS file specification

**Logical name table**

Group logical name table for a groupwide database, or system logical name table for a systemwide database

**Considerations**

- ◆ This logical is required when system logging is activated. Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260, for details on system log recovery.
- ◆ For a systemwide database, this logical must be in the VMS system logical name table even if the systemwide database is serviced by a multiple systemwide PDM. See the considerations for the logical CSTUDSLF in “[LOGICALS.COM](#)” on page 63.

---

**GVSHEMA filename.GBL**

**Description**     *Optional.* Global view file containing selected views in a preopened state.

**Format**            VMS file specification

**Logical name table**

Any

**Considerations**

- ◆ This is only used for RDM access to your database.
- ◆ Use of this logical with the logical [CSI\\_NODIRECTORY](#) set to TRUE allows you to bypass use of the SUPRA Server Directory database (SUPRAD).
- ◆ Refer to the *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220, for information on the creation and proper use of the global view file.

## **GVSHEMA\_SYS TRUE**

**Description**     *Optional.* Specifies that the global view file should be loaded into a systemwide global section instead of a groupwide global section.

**Logical name table**

Any

**Considerations**

- ◆ If this logical is not defined, or if it is defined as any value other than TRUE, the global view file is loaded in a groupwide global section.
- ◆ Refer to the *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220, for more information on global view files.

## ***pdmname\_USER\_INIT.COM***

The procedure *pdmname\_USER\_INIT.COM* defines the logical CSI\_SYSPDMID for a multiple systemwide PDM. The procedure *pdmname\_USER\_INIT.COM* is only generated when you create a multiple systemwide PDM environment. This procedure should be invoked by each application task prior to attempting to use any database serviced by this multiple systemwide PDM.

*pdmname\_USER\_INIT.COM* defines the logical CSI\_SYSPDMID in the process logical name table, unless you call the procedure with the logical name table as the optional parameter. For example:

```
$ @SUPRA_LIBRARY:TESTPDM_USER_INIT.COM
$ @SUPRA_LIBRARY:TESTPDM_USER_INIT.COM LNM$GROUP_000100
```

where TESTPDM is the name of the multiple systemwide PDM.

---

### **CSI\_SYSPDMID *pdmname***

**Description**     *Required* for multiple systemwide PDMs.

**Format**            *pdmname* 1–8 character name identifying the PDM.

**Logical name table**  
Any

#### **Considerations**

- ◆ CSI\_SYSPDMID supersedes CSI\_PDMID. See “PDM\_LOGICALS\_\*.COM” on page 87.
- ◆ If you pass a parameter into the *pdmname\_USER\_INIT.COM* file that table must be included in the table search list logical LNM\$FILE\_DEV that is defined in each user process or one of the default tables for each process. Otherwise the user process will not see the logical defined by the *pdmname\_USER\_INIT.COM*.

## SUPRA\_SYMBOL.COM

The procedure SUPRA\_SYMBOL.COM is located in the directory identified by the logical SUPRA\_COMS . It defines the symbols shown in the following table to access various utilities from the DCL command line.

Symbol	Function
DBA	Initiates the DBA Facility.
DBAID	Runs the DBAID utility.
DIRM	Initiates the Batch Directory Maintenance utility.
FORMAT	Formats data set files and/or log files from the command line.
GLOBAL	Creates a Global View File.
REPORT	Generates DBA Reports.
RUNBASIC	Precompiles an RDML program written in BASIC.
RUNCOBOL	Precompiles an RDML program written in COBOL.
RUNFORTRA*N	Precompiles an RDML program written in FORTRAN.
SUPRA_MENU	Initiates the SUPRA Server Administration Utilities.

You can choose to execute SUPRA\_SYMBOL.COM from your SYLOGIN procedure. The symbols beginning with RUN are for optional components that you may not have.

---

## Modifying VMS system parameters

To run SUPRA Server on your VMS system, your system parameters must allow sufficient resources for your PDM(s) to operate. Use standard VMS utilities to maintain your system parameters.



---

We recommend using AUTOGEN. See your VMS documentation for details.

---

The parameters you may need to change for SUPRA Server are:

- ◆ The size of the system disk buffers
- ◆ The amount of memory any given process may take
- ◆ The amount of CPU time a process is given
- ◆ Number of global pages
- ◆ Number of global sections
- ◆ PQL parameters, which you can also control with the parameters in the CSISTRINP start-up resource file (see “[Creating a PDM start-up resource file](#)” on page 114).



---

Alterations made using AUTOGEN do not become effective until you reboot the machine.

---



# 4

## Entering input parameters



---

You do not need to recreate a new environment for your production and test PDMs. Rerun the SUPRAPDM\_SERVICE\_LEVEL.COM and LOGICALS.COM before bringing up a new PDM.

---

This chapter describes the input parameters for your PDM start-up resource file (identified by the logical CSISTRINP) and your PDM input file (identified by the logical CSIPDMINP). It also describes in more detail how you can use VMS file protection and Access Control Lists (ACLs) to control access to your data.

A template for your PDM start-up resource file is provided in SUPRA\_LIBRARY with the name CSISTR.CSI. The PDM input file is created for you based on your responses when creating your PDM environment using SUPRA\_MENU.COM.

## Creating a PDM start-up resource file

The detached PDM process uses a different set of resources than those used by the typical application. You specify these resources in the PDM start-up resource file, which is identified by the logical name CSISTRINP.

A template PDM start-up resource file is provided in SUPRA\_LIBRARY with the name CSISTR.CSI. Alternatively, you can create the file using a standard editor such as EDIT/EDT or EDIT/TPU. You include in this file all the resources and quotas you wish to apply to the PDM by specifying each parameter and its value. For example: SUBPROCESS\_LIMIT=10 allows the PDM to create up to ten subprocesses. Start any comment lines with an exclamation mark in column one. In addition to specifying resources and quotas in the PDM start-up resource file, you can force the PDM to start under a particular UIC name.

If you do not specify a start-up resource file, the PDM uses the UIC name of the initiating task and the SYSGEN PQL default values. If these defaults are lower than the minimum suggested to run the PDM, the PDM sends a warning message to its log file (CSIPDMLOG) and continues execution. If the PDM process exhausts its resources, your application or the PDM may fail.

You can use the start-up resource file to allocate specific quotas to the PDM in one of two ways:

- ◆ Force the PDM to start up under a specified UIC name using the PDM\_UIC\_OWNER=(*user-name*) parameter in the start-up resource file. You can then give the specified UIC all the quotas you wish to allocate to the PDM process.
- ◆ Set the PDM quotas at start-up using the start-up resource file parameters.

Both methods use the start-up image identified by the logical CSISTR, and a resource file identified by the logical CSISTRINP, which contains either a specification for the PDM UIC name or a set of PDM quotas or both.

## Assigning a UIC name to the PDM

To assign a particular UIC name to the PDM, include the PDM\_UIC\_OWNER parameter in the start-up resource file. If you include this parameter in the start-up resource file, the PDM will always start under the specified UIC name, no matter which application task initiates it.

You can specify parameters in uppercase or lowercase. The syntax of the UIC name parameter is:

---

### **PDM\_UIC\_OWNER=(*uic-name*)**

- |                      |  |
|----------------------|--|
| <b>Description</b>   | <i>Optional.</i> Specifies the owner of the process in which the PDM image is being run.   |
| <b>Default</b>       | Owner of the process initiating the PDM  |
| <b>Format</b>        | Valid VMS UIC name   |
| <b>Consideration</b> | If the UIC name specified is invalid, the PDM will send a warning; however, the start-up will continue with the PDM taking the same UIC name as the initiating task. |

## Specifying PDM quotas

You can specify quotas by entering parameters in the PDM start-up resource file. Type the parameters in uppercase or lowercase. You can define any or all of the parameters listed in the following table:

Parameter specified in start-up resource file	VMS SYSGEN equivalent
AST_LIMIT	PQL_DASTLM
BUFFER_LIMIT	PQL_DBYTLM
ENQUEUE_LIMIT	PQL_DENQLM
EXTENT	PQL_DWSEXTENT
FILE_LIMIT	PQL_DFILLM
IO_BUFFERED	PQL_DBIOLM
IO_DIRECT	PQL_DDIOLM
MAXIMUM_WORKING_SET	PQL_DWSQUOTA
PAGE_FILE	PQL_DPGFLQUOTA
QUEUE_LIMIT	PQL_DTQELM
SUBPROCESS_LIMIT	PQL_DPRCLM
WORKING_SET	PQL_DWSDEFAULT

Any values you specify in the PDM start-up resource file must be greater than the system-defined minimum for detached processes. The system minimum and default values can be the same. Refer to the appropriate VMS documentation for a list of minimum and default VMS values or use the following VMS command:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> SHOW/PQL
```

Minimum values are PQL\_M(*quota-name*).

Default values are PQL\_D(*quota-name*).

Once you have included PDM quotas in the PDM start-up resource file, you can change them if necessary. You modify the value in the resource file, shut down the PDM and restart it. The new PDM process will automatically obtain the new quotas.

All output from CSISTR goes only to the file you specify using the logical name CSISTRLOG. Therefore, you can check this log file to find out whether the PDM has started successfully. If the PDM fails repeatedly, an error status of NMAC is returned and the PDM does not try to restart.

MAXIMUM\_WORKING\_SET should always be at least the same amount and usually higher than WORKING\_SET, and less than or equal to WSEXTENT.

1. PGFLQUOTA—Page file quota, which is the limitation of pageable memory that may be used by the process.
2. WSEXTENT—Working set extent, which is the limitation on physical memory that may be used by the process.

If the sum of these two process quotas is greater than the SYSGEN parameter VIRTUALPAGECNT, then the task is permitted access to only the number of pages of memory specified by VIRTUALPAGECNT.

The PDM process normally uses less than the full amount of the page file quota—this is quite likely for normal loads as the whole of WSEXTENT may be available to the PDM process. When the machine is more heavily loaded then only WSQUOTA is guaranteed, which puts a correspondingly heavier load on the page file.

Details on each of the PDM quotas are on the following pages.

## AST\_LIMIT=(value)

**Description** *Optional.* Specifies the maximum number of Asynchronous System Traps (ASTs) the PDM can have outstanding.

**Default** 24

**Minimum** The optimum value depends on the number of pending ASTs. The recommended minimum value for SUPRA Server is 100.

**Consideration** If not specified, the AST\_LIMIT value is taken from the SYSGEN parameter PQL\_DASTLM.

**Calculation**  $A + B + C + 3$

where:

$A + B = \text{MAXTASKS}$

$C = \text{MAXTHREADS} * (\text{number of databases loaded in the PDM})$

---

**BUFFER\_LIMIT=(value)**

**Description** *Optional.* Specifies the maximum amount of memory, in bytes, the PDM can use for buffered I/O operations or temporary mailbox creation.

**Default** 8192

**Minimum** The recommended minimum value for SUPRA Server is 40000.

**Consideration** If not specified, the BUFFER\_LIMIT value is taken from the SYSGEN parameter PQL\_DBYTLM.

**Calculation**  $\text{BUFFER\_LIMIT} = (\text{amount to open PDM}) + (128 * \text{number of files to open}) + (\text{largest buffer operation})$

where:

- ◆ (amount to open to PDM) = (BUFFER\_LIMIT quota specified for PDM) minus (Buffered I/O Byte count quota after just running CSISTR without loading any dbmods)
- ◆ Buffered I/O Byte count quota = shown using the \$SHOW PROCESS/QUOTA/ID=*nnn* command
- ◆ *nnn* = the PID of the PDM process

Additionally, each file that is concurrently open may require 128 units of Byte Count Quota.

## ENQUEUE\_LIMIT=(value)

**Description** *Optional.* Specifies the number of lock requests the PDM can have outstanding at any one time.

**Default** 30

**Minimum** The recommended minimum value for SUPRA Server is 200.

**Consideration** If not specified, the ENQUEUE\_LIMIT value is taken from the SYSGEN parameter PQL\_DENQLM.

**Calculation** The parameter setting for RMS\_DFMBFIDX (multibuffer count for RMS index files) is:

```
SET RMS/BUFFER=xx/system
```

SHOW RMS command will indicate the current value of this parameter. It is worth noting that increasing the value for MULTIBUFFER count reduces I/O, thereby speeding up the application. If the multibuffer count is set up on a systemwide basis, then the number of locks held by the RMS will be:

```
number index files opened * multi-buffer-count
```

On top of locks used by SUPRA PDM itself, PDM requires additional lock quota for RMS to perform its own record locking. The number of locks required is:

```
number of secondary keys*multi-buffer-count
```

usually the system parameter RMS\_DFMBC.

---

**EXTENT=(value)**

**Description** *Optional.* Specifies the maximum size to which the PDM can increase its physical memory.

**Default** 200

**Minimum** The recommended minimum value for SUPRA Server is 4096.

**Considerations**

- ◆ If not specified, the EXTENT value is taken from the SYSGEN parameter PQL\_DWSEXTENT.
- ◆ The value cannot be greater than the value for the SYSGEN parameter WSMAX.

**Calculation** After the PDM has loaded the databases, check the peak virtual memory size of the PDM process by using one of the following:

- ◆ `$show process/accounting`
- ◆ `$show process/continuous`

and set the WSEXTENT as close to that value as is reasonable for your site. Make sure that WSEXTENT (EXTENT in the csipdminp file) does not exceed 133% of PGFLQUO (PAGE\_FILE in the csipdminp file).

---

**FILE\_LIMIT=(value)**

**Description** *Optional.* Specifies the maximum number of files the PDM can have open at any one time.

**Default** 16

**Minimum** The recommended minimum value for SUPRA Server is 512.

**Consideration** If not specified, the FILE\_LIMIT value is taken from the SYSGEN parameter PQL\_DFILLM.

**Calculation** Maximum number of files the PDM server processes will concurrently have open. This includes the dbmod file, log files, dataset files, index files, and DECnet remote task connections for all loaded databases including the SUPRAD database, datasets, and tasklog.

---

### IO\_BUFFERED=(value)

- Description** *Optional.* Specifies the maximum number of buffered I/O operations the PDM can have outstanding.
- Default** 18
- Minimum** The recommended minimum value for SUPRA Server is 50.
- Consideration** If not specified, the IO\_BUFFERED value is taken from the SYSGEN parameter PQL\_DBIOLM.
- Calculation** Determine by monitoring the PDM process and setting the limit slightly above the values reached during peak processing.
- 

### IO\_DIRECT=(value)

- Description** *Optional.* Specifies the maximum number of direct I/O operations the PDM can have outstanding.
- Default** 18
- Minimum** The recommended minimum value for SUPRA Server is 20.
- Consideration** If not specified, the IO\_DIRECT value is taken from the SYSGEN parameter PQL\_DDIOLM.
- Calculation** Determine by monitoring the PDM process and setting the limit slightly above the values reached during peak processing.
- 

### MAXIMUM\_WORKING\_SET=(value)

- Description** *Optional.* Specifies the amount of physical memory guaranteed to the PDM, assuming there are sufficient resources.
- Default** 200
- Minimum** The recommended minimum value for SUPRA Server is 1000.

**Considerations**

- ◆ If not specified, the MAXIMUM\_WORKING\_SET value is taken from the SYSGEN parameter PQL\_DWSQUOTA.
- ◆ The value cannot be greater than the value for the PDM quota EXTENT.

---

**PAGE\_FILE=(value)**

**Description** *Optional.* Specifies the maximum number of pages that can be allocated in the paging file for the PDM.

**Default** 2048

**Minimum** The recommended minimum value for SUPRA Server is 40000.

**Consideration** If not specified, the PAGE\_FILE value is taken from the SYSGEN parameter PQL\_DPGFLQUOTA.

**Calculation** 
$$\text{PAGE\_FILE} = 4000 \text{ (for Alpha or 1400 for VAX)} + 1.35 * (\text{all dbmod sizes} + \text{largest dbmod size}) / 512.$$

This equation factors in the largest dbmod size twice, thus allowing room for memory fragmentation. The dbmod sizes, as shown with the csiopcom command display/databases, are in bytes and thus are divided by 512 to convert to pagelets.

---

**QUEUE\_LIMIT=(value)**

**Description** *Optional.* Specifies the maximum number of timer queue entries the PDM can have outstanding at any one time.

**Default** 8

**Minimum** The recommended minimum value for SUPRA Server is 20.

**Consideration** If not specified, the QUEUE\_LIMIT value is taken from the SYSGEN parameter PQL\_DTQELM.

**Calculation** Determine by monitoring the PDM process and setting the limit slightly above the values reached during peak processing.

---

**SUBPROCESS\_LIMIT=(value)**

**Description** *Optional.* Specifies the maximum number of subprocesses the PDM can create.

**Default** 8

**Minimum** The recommended minimum value for SUPRA Server is 8.

**Consideration** If not specified, the SUBPROCESS\_LIMIT value is taken from the SYSGEN parameter PQL\_DPRCLM.

## **WORKING\_SET=(value)**

**Description** *Optional.* Specifies the limit on the number of pages in the initial working set for the PDM.

**Default** 100

**Minimum** This value must not be greater than the value specified in the MAXIMUM\_WORKING\_SET parameter. The recommended minimum value for SUPRA Server is 500.

**Consideration** If not specified, the WORKING\_SET value is taken from the SYSGEN parameter PQL\_DWSDEFAULT.

**Example** The following is an example PDM start-up resource file:

```
AST_LIMIT=100
BUFFER_LIMIT=40000
ENQUEUE_LIMIT=200
EXTENT=4096
FILE_LIMIT=512
IO_BUFFERED=50
IO_DIRECT=20
MAXIMUM_WORKING_SET=1000
PAGE_FILE=40000
QUEUE_LIMIT=20
SUBPROCESS_LIMIT=8
WORKING_SET=500
```

A template file is provided for you in SUPRA\_LIBRARY:CSISTR.CSI. See the description of the logical **CSISTRINP**.

---

## Entering parameters for the PDM input file

When you generate a PDM system using SUPRA\_MENU.COM, the answers you give to some of the questions result in the PDM input file, which is named SUPRA\_LIBRARY:PDM\_OPTIONS\_\*.INP (where \* is the 6-digit group number for a groupwide PDM, 000000 for a systemwide PDM, or the translation of CSI\_SYSPDMID for a multiple systemwide PDM). The PDM input file contains the input parameters for the PDM.

The PDM input file is identified by the logical name CSIPDMINP. This logical is defined in the PDM\_LOGICALS\_\*.COM procedure (see “PDM\_LOGICALS\_\*.COM” on page 87). The PDM translates the logical name CSIPDMINP to locate the PDM input file and uses the parameters it contains to initiate the PDM.

The parameters in the input file are:

ACLCHECK	MAXDATA	RETRY
CONSOLE	MAXTASKS	SINGLEUNLOAD
DYNSLOCK	MAXTHREADS	SLFDUMPPRI
IDXCNVERR	MRELAY	STATISTICS
IDXDUPERR	MULTIHOLD	SYSOPCOM
IDXTIMEOUT	OPERATOR	TIMEOUT
INTERVAL	PDMNAME	UICCHECK
LOGFLUSH	PRIORITY	WARMSTART_ DATASET_ERROR

Details on each of these parameters are on the following pages.

**ACLCHECK=**

Y
N

**Description** *Optional.* Specifies whether the PDM checks for any Access Control Lists (ACL) set on PDM data set files, task log files, or system log files before allowing a task to access them.

**Default** N

**Considerations**

- ◆ When ACLCHECK=Y, the PDM does not check for ACLs placed on the compiled database description (*database-name.MOD*) file or on index files.
- ◆ You can use this parameter with **UICCHECK**. See “[Setting up and using PDM file protection checking](#)” on page 141.

---

**CONSOLE=**

Y
N

**Description** *Optional.* Specifies whether the PDM will send informational and error messages to operator terminals.

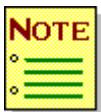
**Default** Y

- Options**
- Y Displays informational and error messages on operator terminals.
  - N Suppresses all PDM messages from being displayed on operator terminals except “Reply with a SUPRA PDM command,” which is controlled by the parameter **SYSOPCOM**.

**Consideration** The operator terminal that messages may be sent to is identified by the **OPERATOR** parameter.

DYNSLOCK=  $\begin{bmatrix} Y \\ N \end{bmatrix}$

**Description** *Optional.* Specifies whether you want the data sets in a database to remain locked if they have been updated by a task which fails to sign off normally when no task logging is in use ( when logical data corruption is likely).



**Warning:** Use of the DISABLE/DYNAMIC operator command overrides this option (if you issue DISABLE/DYNAMIC for the database, then the files will be unlocked regardless of the DYNSLOCK setting, and the database will be disabled when the data sets have been updated by a task which fails to sign off normally when no task logging is in use.) See “Disabling a database (DISABLE)” on page 151 for a complete description of DISABLE/ DYNAMIC.



**Warning:** If a task fails to sign off normally when data sets have been updated and there is no active task log, the updated data sets may be logically incorrect.

**Default** Y

**Options** Y Data sets remain locked if the task fails to sign off normally and updates have been made when no task log is active.

N Data sets do NOT remain locked if the task fails to sign off normally and updates have been made and no task log is active.

### Considerations

- ◆ Setting the parameter value to Y keeps file locking consistent with earlier releases of the PDM.
- ◆ If the PDM fails and there is no active task log, the updated data sets will remain locked.

## IDXCNVERR= $\begin{matrix} \text{Y} \\ \text{N} \end{matrix}$

- Description** *Optional.* Specifies the action you want the PDM to take when it encounters an error while converting data into a sortable format for the index file.
- Default** Y
- Options**
- Y The PDM backs out the updates to the PDM data set and returns a status of IDAT to the application. This causes the PDM to keep the index file consistent with its associated PDM data set.
  - N The PDM does not back out the updates to the PDM data set. Instead, the PDM deactivates the index file because the file is inconsistent with its associated PDM data set.
- 

## IDXDUPERR= $\begin{matrix} \text{Y} \\ \text{N} \end{matrix}$

- Description** *Optional.* Specifies the action you want the PDM to take when duplicate secondary keys are not allowed and the PDM attempts to insert into an index file a record that has the same secondary key value as another record in that index file.
- Default** Y
- Options**
- Y The PDM backs out the updates to the PDM data set and returns a status of DUPI to the application. This causes the PDM to keep the index file consistent with its associated PDM data set.
  - N The PDM does not back out the updates to the PDM data set. Instead, the PDM deactivates the index file because the file is inconsistent with its associated PDM data set.
- 

## IDXTIMEOUT=*nnnn*

- Description** *Optional.* Specifies the number of intervals (see [INTERVAL](#)) to wait for an index file's I/O request to complete before canceling that index file's I/O request.
- Default** 0
- Consideration** 0 indicates that the index files' I/O requests should never be canceled.

---

**INTERVAL=nnnn**

**Description** *Optional.* Specifies a period of time, measured in hundredths of a second.

**Default** 5

**Options** 1–1000

**Consideration** This time period is used by the **IDXTIMEOUT**, **TIMEOUT**, and **RETRY** parameters.

---

**LOGFLUSH=**

Y
N

**Description** *Optional.* Indicates whether you want CSIPDM messages flushed to the CSIPDMLOG file for viewing while the PDM is still up and running.

**Default** Y

**Consideration** There is a performance penalty associated with message flushing, especially when **STATISTICS** are enabled.

## MAXDATA=nnnnn

<b>Description</b>	<i>Optional.</i> Sets the maximum size of the message buffer used for communication between the PDM and applications, in number of bytes.
<b>Default</b>	4096
<b>Options</b>	0–32767

### Considerations

- ◆ The message area must be large enough to contain all the parameters passed to the PDM. These include the following:
  - User data area
  - Element list
  - Key field
  - Status function
  - End parameters
- ◆ The value of MAXDATA \* MAXTASKS determines the size of the global section used by the PDM for communication between the PDM and application tasks. The greater you make either of these, the greater the global page requirement.

---

**MAXTASKS=nnnn**

**Description** *Optional.* Specifies the maximum number of tasks allowed to concurrently access the PDM.

**Default** 50

**Options** 1–1000

**Considerations**

- ◆ You may wish to impose a maximum to ensure good performance for tasks that access the PDM.
- ◆ The value of **MAXDATA** \* **MAXTASKS** determines the size of the global section used by the PDM. The greater you make either of these, the greater the global page requirement.
- ◆ Each RDM application running with a SUPRA Server Directory requires up to two tasks each time it signs on to a database. See the description of the logical **CSI\_NODIRECTORY**.
- ◆ This is the limit for the PDM, which can impact the **MAX-TASKS** and **MAX-UPDATE-TASKS** for all databases serviced by this PDM.

## **MAXTHREADS=nnn**

<b>Description</b>	<i>Optional.</i> Specifies the maximum number of functions the PDM can process concurrently for each database it services.
<b>Default</b>	3
<b>Options</b>	1–100

### **Considerations**

- ◆ The optimum value for MAXTHREADS is the number of tasks + 2, up to a maximum of 100. The more threads you have, the more concurrent I/Os you can have. The optimum value will vary from site to site.
- ◆ Too low a value for MAXTHREADS could cause the PDM to hibernate while there is work to do. This will happen if each of the active threads is waiting on an event, such as the completion of a physical disk I/O.
- ◆ Too high a value can waste virtual memory.
- ◆ It is better to have too high a value than too low a value because PDM threads can increase database throughput dramatically.

---

**MRELAY=**

Y
N

**Description** *Optional.* Specifies whether to send console messages output by the PDM and the system log dump program, **CSTUDSLF**, to a VMS mailbox.

**Default** N

**Considerations**

- ◆ If you specify Y, users must write programs to pick up console messages from the mailbox. If the mailbox fills up, the PDM keeps a count of the number of messages lost and displays this number on the next successful send.
- ◆ You can use MRELAY in conjunction with the logical name CSI\_MRELAY to trap all messages from the Database Access program identified by the logical CSIDAP. See “Automating operator communication” on page 183 and “Example mailbox-reading program” on page 275 for more details about writing a mailbox-reading program to read the PDM messages.

---

**MULTIHOLD=**

Y
N

**Description** *Optional.* Specifies whether you want a task to be able to explicitly read and hold more than one record per file in a single logical unit of work.

**Default** Y

**Consideration** If you want your SUPRA Server applications to be more compatible with ULTRA, you should enable MULTIHOLD. If, however, you want your applications to be more compatible with IBM SUPRA Server, you should disable MULTIHOLD.

**OPERATOR=** **OPERnn**  
**OPER1**

**Description** *Optional.* Identifies the VMS operator number which may send commands to the PDM and receive messages from the PDM.

**Default** OPER1

**Options** OPER1–OPER12

**Considerations**

- ◆ You can specify only one operator number in the input file, although you can enable more than one terminal as that operator number.
- ◆ The PDM sends all messages to the OPER terminal(s) except those messages with severity level L (log only). These are sent to the PDM log file (**CSIPDMLOG**).
- ◆ When the PDM input parameter **SYSOPCOM** is enabled, VMS OPCOM prompts all specified operator terminals at regular intervals (5 or 10 minutes). Use the VMS REPLY command to respond to the operator prompt to enter a PDM operator command. See the VMS documentation for a description of the VMS REPLY command and its parameters and qualifiers (the repeated operator message is a VMS facility not controlled by the PDM).

---

**PDMNAME=***pdmname*

**Restriction** Only for a multiple systemwide PDM.

**Description** *Required* only if using a multiple systemwide PDM. Specifies the name of the multiple systemwide PDM.

**Format** 1–8 alphanumeric characters

**Consideration** *pdmname* must be the same as the equivalence string for the logical name CSI\_SYSPDMID.

---

**PRIORITY=nn**

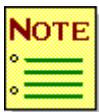
**Description**     *Optional.* Sets the VMS priority of the detached PDM process.

**Default**             Base priority of the invoking process

**Options**             1–15

**Considerations**

- ◆ You do not need the ALTPRI privilege to set the priority of the PDM as an input file parameter. Therefore, the initiating task can assign the PDM a priority higher than its own. This is because the CSIPDM image file is installed with ALTPRI.



---

Cincom recommends that you set the PDM server base priority to a value higher than that of the interactive and batch client tasks it will service. However, a base priority greater than 9 could result in competition with many VMS system processes which have base priorities in the 10–15 range. Since the PDM server should have a priority greater than that of the client tasks it will service, process preemption must be considered. The SYSGEN parameter `PRIORITY_OFFSET` is used to specify the priority offset for process preemption. Therefore, the PDM server priority should be set to the sum of the base priority for interactive tasks (generally 4) plus the value of `PRIORITY_OFFSET` (sometimes 3), plus 1 if you do not want the PDM server to be preempted by an application client task. Refer to your VMS documentation for a complete discussion of VMS process priority preemption and `PRIORITY_OFFSET`.

---

**RETRY=nnn**

**Description** *Optional.* Specifies the number of times the PDM attempts to obtain a held record before returning a HELD status.

**Default** 5

**Options** 1–100

**Consideration** The PDM waits for the period specified in numbers of intervals before retrying. For example, INTERVAL=5, RETRY=10 causes the PDM to retry ten times at 5/100 second intervals.

---

**SINGLEUNLOAD=**

Y
N

**Description** *Optional.* Indicates whether you want the CSIPDM server process to unload a dbmod from memory upon the signing off of a SINGLE mode job.

**Default** Y

**Consideration** Normally a dbmod should be unloaded after a SINGLE mode job completes in order that normal UPDATE processing may proceed. If you set this parameter to N, then you are responsible for manually unloading the dbmod (using the CSIOPCOM command UNLOAD).

---

**SLFDUMPPRI=nnnn**

**Description** *Optional.* Specifies the VMS process base priority of the dump of the system log files if the user need to change it from the same base priority as the PDM.

**Default** PDM priority value

**Options** 1–15

**Considerations**

- ◆ A lower priority value could result in a longer time period before completing a system log dump.
- ◆ A value at or higher than the PDM's own priority could result in more contention with processing that is already signed on and running.

---

**STATISTICS=**

Y
N

**Description** *Optional.* Indicates whether you want detailed PDM statistics. These statistics include physical I/O counts, the number of in-memory buffer hits, and many more items (see “[PDM statistics output](#)” on page 269).

**Default** N

**Considerations**

- ◆ Y sends the statistics to the following:
  - The PDM log file as identified by the logical name CSIPDMLOG
  - The message reading mailbox if MRELAY=Y
- ◆ Statistics are always gathered by the PDM, but are not automatically displayed when this parameter is set to N.
- ◆ You can use operator commands to display statistics even when this parameter is set to N. Use the commands ENABLE STATISTICS and DISPLAY /STATISTICS to do so.

## SYSOPCOM= | | |---| | Y | | N |

- Description** *Optional.* Specifies whether you can send commands to the SUPRA Server PDM using the VMS OPCOM utility.
- Default** Y
- Options** Y To communicate with the PDM via both the VMS REPLY command and the CSIOPCOM facility.
- N To communicate only through CSIOPCOM.
- Consideration** When SYSOPCOM is set to Y, VMS OPCOM prompts all specified operator terminals at regular intervals (5 or 10 minutes). Use the VMS REPLY command to respond to the operator prompt to enter a PDM operator command. See the VMS documentation for a description of the VMS REPLY command and its parameters and qualifiers (the operator message is a VMS facility not controlled by the PDM).
- When SYSOPCOM is set to N, and a database index has its INDEX-CORRUPT-ACTION set to O (Operator), the PDM server will still prompt the user as the operator console.

---

## TIMEOUT=nnnnn

- Description** *Optional.* Specifies the period, in numbers of intervals, before a task is dynamically reset and signed off if it remains inactive.
- Default** 0 (no dynamic sign off due to application task inactivity)
- Options** 0–10000

---

**UICCHECK=**

Y
N

**Description** *Optional.* Specifies whether the PDM checks the UIC-based protection set on PDM data set files, task log files, and system log files before allowing a task to access them.

**Default** N

### Considerations

- ◆ When UICCHECK=Y, the PDM does not check the protection set on the compiled database description (*database-name.MOD*) file or on index files.
- ◆ Use this parameter in conjunction with **ACLCHECK** to control PDM file protection checking. See “[Setting up and using PDM file protection checking](#)” on page 141 for a full description of how to set up and use PDM file protection checking.

---

**WARMSTART\_DATASET\_ERROR=**

A
C
O

**Description** *Optional.* Indicates how the PDM should process data set errors identified during warm start recovery.

**Default** C

### Options

- A Abort - Warm start recovery aborted.
- C Continue - Errors like file not found, or invalid data set size are ignored - This is the previous behavior.
- O Operator - Operator message is sent to the console asking the user whether warm start recovery should continue - Default response is No.

### Considerations

- ◆ When parameter is set to O and the error is found, startup will pause until an answer from the operator is received.
- ◆ Refer to the [SUPRA Server PDM Messages and Codes Reference Manual \(PDM/RDM Support for UNIX & VMS\)](#), P25-0022, for the accompanying codes this message can generate.

**Example**

The following example illustrates the contents of a PDM input file which specifies values for all possible parameters.

```
ACLCHECK=N
CONSOLE=Y
DYNLOCK=Y
IDXCNVERR=Y
IDXDUPEERR=Y
IDXTIMEOUT=0
INTERVAL=33
LOGFLUSH=Y
MAXDATA=4096
MAXTASKS=11
MAXTHREADS=13
MRELAY=N
MULTIHOLD=Y
OPERATOR=OPER9
PDMNAME=SYS1
PRIORITY=7
RETRY=10
SINGLEUNLOAD=Y
STATISTICS=N
SYSOPCOM=N
TIMEOUT=0
UICHECK=N
```

## Setting up and using PDM file protection checking

The SUPRA Server PDM image (identified by the logical CSIPDM) is installed with the BYPASS privilege. This privilege allows the PDM to access any file on your system. The two PDM input parameters **ACLCHECK** and **UICCHECK** allow you to control how the PDM checks protection on files before allowing a task to access them.

You can use the ACLCHECK=Y parameter in the PDM input file to cause the PDM to check the Access Control Lists (ACLs) on database files. ACLs allow you to define more selective database access rights than you can through UIC-based protection alone. You can define which tasks can access database files and what operations they can perform on those files.

In addition, you can set up the PDM to override the ACL and UIC-based protection. This means you can protect sensitive files against DCL access so users cannot dump a file from DCL. However, because the PDM can bypass the file protection, those users can access the files through the PDM.

The following shows the different combinations of the ACLCHECK and UICCHECK parameters and how the PDM functions based on those combinations:

**ACLCHECK=Y and UICCHECK=Y.** The PDM checks both the ACL and the UIC-based protection of a data file against the UIC of the accessing task as follows:

- ◆ If there is an Access Control Entry (ACE) which denies access, the PDM rejects the attempted operation with an error.
- ◆ If there is an ACE which allows access, the PDM proceeds with the operation.
- ◆ If there is no ACE for the accessing task, the PDM checks the UIC-based protection.

You might set up your PDM protection checking to check both ACLs and UIC-based protection if you want to specify the operations that each PDM task can perform without giving unrestricted access to all tasks or limiting them to the same level of access. For example, assume you have one important data file, PERSONNEL.INF, and five accessing tasks, BOSS, MANAGER, ADMIN, PLEB1, and PLEB2. You could set up the ACL for PERSONNEL.INF as follows:

```
( IDENTIFIER=[ BOSS ] , ACCESS=READ+WRITE )  
  ( IDENTIFIER=[ MANAGER ] , ACCESS=READ+WRITE )  
    ( IDENTIFIER=[ ADMIN ]+INTERACTIVE , ACCESS=READ+WRITE )  
      ( IDENTIFIER=[ PLEB1 ] , ACCESS=NONE )  
        ( IDENTIFIER=[ PLEB2 ] , ACCESS=NONE )
```

If BOSS, MANAGER, or ADMIN attempts to access PERSONNEL.INF, the PDM checks the ACL and allows the task to perform the operations specified in its ACE. PLEB1 and PLEB2 would have no access. If any other UIC attempted to access the file, the PDM would check the UIC-based protection. To make sure that no other users could access PERSONNEL.INF, you could include the following ACE last in the list:

```
( IDENTIFIER=[ * ] , ACCESS=NONE )
```

**ACLCHECK=N and UICCHECK=NB.** The PDM bypasses all protection checking, allowing tasks to use the PDM to access data files they may be prevented from accessing in any other way by the ACL and UIC-based protection.

This is particularly useful, for example, with a payroll database in which you want to allow tasks to access data only through the PDM, and not from DCL command level. This prevents users from dumping the contents of your payroll files or accessing them in any way other than through the PDM.

You would set up the file ACE to allow no access; for example:

```
( IDENTIFIER=[ * ] , ACCESS=NONE )
```

**ACLCHECK=N and UICCHECK Y.** The PDM checks the UIC-based file protection only, bypassing any ACLs.

**ACLCHECK=Y and UICCHECK=N.** The PDM checks the ACLs only, bypassing any UIC-based file protection. If there is no ACE for the specified task, access is denied.

### General considerations

- ◆ Use one of the following VMS commands to define ACLs:  

```
SET ACL /EDIT filename.type  
EDIT/ACL filename.type  
SET FILE/ACL filename.type
```
- ◆ For VMS Version 5 and above, DEC recommends you use SET ACL /EDIT instead of SET FILE/ACL.
- ◆ Refer to Digital's *VMS Access Control List (ACL) Editor Manual* for details on how to create ACLs.
- ◆ The PDM checks for ACLs set on data set, task and system log files, but does not check for ACLs placed on:
  - The compiled database description (*database-name.MOD*) file, because tasks need to access this file through CSIDAP before the PDM accesses the file
  - Index files, because users who can access the database files should also be able to access connected index files



---

SUPRA Server 2.4 does not support the system-defined identifiers DIALUP and REMOTE due to a VMS limitation.

---

# 5

## Communicating with the SUPRA Server PDM

The SUPRA Server PDM runs as a detached process. You can communicate with the PDM by using a set of PDM operator commands. You enter PDM operator commands in one of two ways:

- ◆ Through CSIOPCOM, a screen-based interface to the SUPRA Server PDM that offers comprehensive online help (see [“Communicating with the SUPRA Server PDM through CSIOPCOM”](#) on page 174)
- ◆ Through the VMS REPLY command (see [“Communicating with the PDM through the VMS REPLY command”](#) on page 189)

Eleven PDM operator commands are available; however, you may not want to give all users access to all operator commands. The user authorization program, which is identified by the logical CSIOAUTH, allows you to specify the commands available to each user. You run CSIOAUTH to create a user authorization file to which you assign the logical name CSIOPCOM\_AUTH. For details about CSIOAUTH, see [“Restricting use of PDM commands”](#) on page 185.



---

The user authorization file applies only to users communicating with the PDM through CSIOPCOM. Users of the VMS REPLY command can enter any PDM operator command; however, these users must have the OPER privilege.

---

You can also write your own interface to SUPRA Server PDM. [“Automating operator communication”](#) on page 183 describes how to direct the SUPRA Server PDM messages to a mailbox. You can read these messages through a user-written program. This section also contains an example mailbox-reading program in pseudocode. See [“Example mailbox-reading program”](#) on page 275 for an example mailbox-reading program written in COBOL.

You can use the PDML OPCOM command to send operator commands to the PDM from user-written programs. Refer to the [SUPRA Server PDM Programming Guide \(UNIX & VMS\)](#), P25-0240, for more information.

---

## Using the PDM operator commands

The SUPRA Server PDM operator commands control PDM usage. The following table lists the operator commands the PDM recognizes:

Command	Description
ACTIVATE	Starts logging index records on the specified index file(s).
DEACTIVATE	Stops logging index records on the specified index file(s).
DISABLE	Unloads one or all database(s) and prevents anyone else from reloading.
DISPLAY	Displays the status of loaded databases, the status of sign-on tasks, statistics, or held records.
DUMPSLF	Dumps a system log component manually.
ENABLE	Cancels the DISABLE restriction.
POPULATE	Populates one or more indexes by reading records from the PDM data set and writing corresponding index records to the index file.
READONLY	Sets one or all database(s) to read-only processing.
SHUTDOWN	Unloads all databases and stops the PDM.
UNLOAD	Unloads one or all databases.
UPDATE	Cancels the READONLY restriction.

## Activating an index (ACTIVATE)

ACTIVATE initiates index record logging. Each subsequent change to the data file is reflected by corresponding changes to the index file(s). Changes to the data file include updating, adding, or deleting records.

---

**ACTIVATE** *index* [*xxx\_*]*database-name* [*[uic - group - number]*]  
**[AT** *node-name*]

---

### *index*

**Description** *Required.* Specifies the index file to be activated for logging.

**Format** *dsetlXyy*

where

*dset* is the 4 alphanumeric character data set name

*IX* is entered as shown

*yy* is the 2 alphanumeric character index name

**Consideration** If the index has been deactivated for any length of time, run the check option of the index maintenance utility program identified by the logical **CSTUIDX**. This checks the index records and updates them where necessary, and automatically activates the index.

---

### **[*xxx\_*]*database-name***

**Description** *Required.* Identifies the database for which index logging is to take effect.

**Format** *xxx\_* 1–3 alphanumeric characters followed by “\_” (the optional database prefix)

*database-name* 6 alphanumeric characters

**[*uic - group - number*]**

- Description**     *Optional.* Identifies the group number if more than one database of the same name is loaded in different groups.
- Format**            1–6 digit UIC group number enclosed in brackets
- Consideration** You can omit any leading zeros; however, you must enter the brackets.
- 

**AT *node-name***

**Description**     *Optional.* Directs the command to the remote PDM running at the specified node.

**Format**            1–6 alphanumeric characters

**Considerations**

- ◆ If you omit this parameter, the command is directed to the local PDM.
- ◆ Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174.)

## Deactivating an index (DEACTIVATE)

DEACTIVATE stops logging index records to the specified index. This means the index file will no longer be updated with modifications to the data set file.

---

```
DEACTIVATE index [xxx_]database-name [[uic - group - number]]
          [AT node-name]
```

---

### *index*

**Description** *Required.* Specifies the index file for which logging is to be deactivated.

**Format** *dset**IX**yy*

where

*dset* is the 4 alphanumeric character data set name

*IX* is entered as shown

*yy* is the 2 alphanumeric character index name

**Consideration** Before reactivating the index, run the check option of the index maintenance utility program, **CSTUIDX**, to check the index records and update them where necessary. Check automatically activates the index.

---

### [*xxx\_*]*database-name*

**Description** *Required.* Identifies the database for which index logging is to take effect.

**Format** *xxx\_* 1–3 alphanumeric characters followed by “\_” (the optional database prefix)

*database-name* 6 alphanumeric characters

**[uic-group-number]**

- Description** *Optional.* Identifies the group number if more than one database of the same name is loaded in different groups.
- Format** 6-digit UIC group number enclosed in brackets
- Consideration** You can omit any leading zeros; however, you must enter the brackets.
- 

**AT node-name**

- Description** *Optional.* Passes the command to the remote PDM running at the specified node.
- Format** 1–6 alphanumeric characters

**Considerations**

- ◆ If you omit this parameter, the command is directed to the local PDM.
- ◆ Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174).

## Disabling a database (DISABLE)

DISABLE has three basic functions:

- ◆ Signing off each task using a database, unloading the database, then preventing the database from being reloaded until the restriction is lifted using the ENABLE command (“[Enabling a database \(ENABLE\)](#)” on page 160). See also the UNLOAD command (“[Unloading a database \(UNLOAD\)](#)” on page 169).
- ◆ Creating the condition where a dynamic sign off from a database will cause the database to be disabled, all other tasks to be signed off, and the database to be unloaded and prevented from reloading until the restriction is removed with the ENABLE command.
- ◆ Disabling the automatic display of PDM statistics.

---

### DISABLE

```

{
  {
    [/COMIT]
    [/SINOF]
    [/FORCE]
  } { [xxx_]database - name [[uic - group - number]] }
  { ALL }
}
/DYNAMIC [xxx_]database - name [[uic - group - number]]
STATISTICS

```

[AT node - name]

---

```

{
  {/COMIT }
  {/SINOF }
  {/FORCE }
  { [xxx_]database - name [[uic - group - number]] }
  ALL
}
/DYNAMIC [xxx_]database - name [[uic - group - number]]
STATISTICS

```

<b>Description</b>	<i>Required.</i> Specifies the object(s) of the DISABLE command.	
<b>Format</b>	<i>xxx_</i>	1–3 alphanumeric characters followed by “_” (the optional database prefix)
	<i>database-name</i>	6 alphanumeric characters
	<i>[uic-group-number]</i>	6-digit UIC group number enclosed in brackets
<b>Options</b>	<i>/COMIT</i>	Signs off each task from the specified database or all databases after the task’s next COMIT or RESET function.
	<i>/SINOF</i>	Waits until the last task has signed off the specified database or all databases.
	<i>/FORCE</i>	Resets any uncommitted updates and signs off each task immediately from the specified database or all databases.
	<i>/DYNAMIC</i>	Dynamically signs off all tasks, unloads and disables the database when that database is likely to contain logical data corruption, when a task which has updated a data set fails to sign off normally when there is no active task log.
	<i>STATISTICS</i>	Disables the automatic displaying of PDM statistics.



**Warning:** The DISABLE/DYNAMIC operator command overrides the PDM input parameter DYNLOCK ( if you issue DISABLE/DYNAMIC for the database, the files will be unlocked regardless of the DYNLOCK setting, and the database will be disabled when the data sets have been updated by a task which fails to sign off normally when no task logging is in use.)

## Considerations

- ◆ The *uic-group-number* parameter specifies a group number if more than one database of the same name is loaded in different groups. You can omit any leading zeros; however, you must enter the brackets.
- ◆ The ALL parameter dynamically signs off each task and disables and unloads all databases. The PDM then creates the following logical name:

```
DISABLED_<global-section-name> TRUE
```

The PDM places this logical assignment in either the group or system logical name table according to whether it refers to a groupwide or systemwide PDM global section. This logical assignment stops any attempt to load any database.

- ◆ Use the ENABLE command to cancel these restrictions.
- ◆ DISABLE/DYNAMIC is designed for CONTROL:Manufacturing sites. If a task gets dynamically signed off from a database for which you have entered DISABLE/DYNAMIC, all other tasks running on that database will be forcibly signed off and the database will be disabled.
- ◆ Under DISABLE/DYNAMIC, if the PDM fails and there is no active task log, the updated data sets will remain locked.

---

### AT *node-name*

<b>Restriction</b>	Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “ <a href="#">Communicating with the SUPRA Server PDM through CSIOPCOM</a> ” on page 174).
<b>Description</b>	<i>Optional.</i> Directs the command to the PDM running at the specified node.
<b>Default</b>	If you omit this parameter, the command is directed to the local PDM.
<b>Format</b>	1–6 alphanumeric characters

## Displaying a database (DISPLAY)

DISPLAY lists some or all of the following details on the screen according to the parameters you specify:

- ◆ Database name
- ◆ Whether the database is loaded in systemwide or groupwide global sections
- ◆ UIC group number
- ◆ The number of bytes in memory taken up by the database
- ◆ The state of the database ( Fail, Inactive, or Active)
- ◆ The number of active tasks
- ◆ The number of active functions
- ◆ The number of active threads
- ◆ Database statistics
- ◆ Database held records

---

```
DISPLAY { /DATABASES
          /TASKS
          /STATISTICS [/FILE = data - set]
          /HELDRECORDS [/FILE = data - set]
          /INDICES [/FILE = data - set] } { [xxx_]database - name[uic - group - number]}
          ALL }
```

[AT *node - name*]

---

/DATABASES /TASKS /STATISTICS [FILE = <i>data-set</i> ] /HELDRECORDS [FILE = <i>data-set</i> ] /INDICES [FILE = <i>data-set</i> ]	}	{ [ <i>xxx_</i> ] <i>database-name</i> [ <i>uic-group-number</i> ]} { ALL }
---	---	--

<b>Description</b>	<i>Required.</i> Specifies the object(s) of the DISPLAY function.										
<b>Format</b>	<table border="0" style="width: 100%;"> <tr> <td style="padding-right: 20px;"><i>xxx_</i></td> <td>1–3 alphanumeric characters followed by “_” (the optional database prefix)</td> </tr> <tr> <td><i>database-name</i></td> <td>6 alphanumeric characters</td> </tr> <tr> <td>[<i>uic-group-number</i>]</td> <td>1–6 digit number enclosed in brackets</td> </tr> <tr> <td><i>data-set</i></td> <td>4-character PDM data set name</td> </tr> </table>	<i>xxx_</i>	1–3 alphanumeric characters followed by “_” (the optional database prefix)	<i>database-name</i>	6 alphanumeric characters	[ <i>uic-group-number</i> ]	1–6 digit number enclosed in brackets	<i>data-set</i>	4-character PDM data set name		
<i>xxx_</i>	1–3 alphanumeric characters followed by “_” (the optional database prefix)										
<i>database-name</i>	6 alphanumeric characters										
[ <i>uic-group-number</i> ]	1–6 digit number enclosed in brackets										
<i>data-set</i>	4-character PDM data set name										
<b>Options</b>	<table border="0" style="width: 100%;"> <tr> <td style="padding-right: 20px;">/DATABASES</td> <td>Displays the status of one or all loaded databases.</td> </tr> <tr> <td>/TASKS</td> <td>Displays the status of signed-on tasks for one or all loaded databases.</td> </tr> <tr> <td>/STATISTICS</td> <td>Displays PDM statistics for one or all loaded databases or one data set.</td> </tr> <tr> <td>/HELDRECORDS</td> <td>Displays the RRN (relative record number) and operating system task (process) for records held in the PDM servers internal holding table.</td> </tr> <tr> <td>/INDICES</td> <td>Displays the status of the indices for one or all loaded databases of one data set.</td> </tr> </table>	/DATABASES	Displays the status of one or all loaded databases.	/TASKS	Displays the status of signed-on tasks for one or all loaded databases.	/STATISTICS	Displays PDM statistics for one or all loaded databases or one data set.	/HELDRECORDS	Displays the RRN (relative record number) and operating system task (process) for records held in the PDM servers internal holding table.	/INDICES	Displays the status of the indices for one or all loaded databases of one data set.
/DATABASES	Displays the status of one or all loaded databases.										
/TASKS	Displays the status of signed-on tasks for one or all loaded databases.										
/STATISTICS	Displays PDM statistics for one or all loaded databases or one data set.										
/HELDRECORDS	Displays the RRN (relative record number) and operating system task (process) for records held in the PDM servers internal holding table.										
/INDICES	Displays the status of the indices for one or all loaded databases of one data set.										

## Considerations

- ◆ The *uic-group-number* parameter specifies a UIC group number if more than one database of the same name is loaded in different groups. You can omit any leading zeros; however, you must enter the brackets.
- ◆ The */FILE* parameter specifies the data set for which statistics are requested.
- ◆ DISPLAY/STATISTICS may tie up your terminal for some time unless you use the CSIOPCOM command SET OUTPUT (*file-spec*) to direct output to a disk file (see “Using CSIOPCOM commands” on page 180).
- ◆ If you are using multiple physical databases, you can display statistics for all databases to which the specified data set is connected by entering:

```
DISPLAY /STATISTICS ALL /FILE=data-set
```
- ◆ See “PDM statistics output” on page 269 for a description of the PDM statistics output.
- ◆ DISPLAY/HELDRECORDS will display the entire record holding table (up to 65536 entries) for a database unless you specify */FILE=* to reduce output.



---

Cincom recommends the use of */FILE=* with */HELDRECORDS* as follows:

---

- ```
DISPLAY/HELDRECORDS (database) /FILE=(dataset)
```
- ◆ DISPLAY/INDICES output can be reduced by using the */FILE* option or the output can be redirected via the CSIOPCOM command SET OUTPUT (*file-spec*) to disk.
  - ◆ For information on the codes returned with the display commands, refer to the explanations for CSTI061R and CSTI064R in the *SUPRA Server PDM Messages and Codes Reference Manual (PDM/RDM Support for UNIX & VMS)*, P25-0022.

**AT node-name**

- Restriction** Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174).
- Description** *Optional.* Directs the command to the PDM running at the specified node.
- Default** If you omit this parameter, the command is directed to the local PDM.
- Format** 1–6 alphanumeric characters

## Dumping the contents of the System Log for a database (DUMPSLF)

The DUMPSLF command is used to manually request the PDM to dump the contents of System Log File components. The PDM must dump the contents of each System Log File component before it can be reused, and before you can run System Log Recovery.

The PDM can carry out this procedure automatically (see “[Understanding the Physical Data Manager \(PDM\)](#)” on page 31). However, after a system failure, one or both log files may contain data which has not yet been dumped. Alternatively, you may wish to dump remaining data to have a complete system log history. You may need this to run the recovery program because the PDM does not automatically dump the system log file if it is not full.

Use the DUMPSLF command in the following situations:

- ◆ To dump one or both System Log File components manually before starting System Level Recovery. Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260, for a description of System Level Recovery.
- ◆ To dump data from a system log which is not full.

---

**DUMPSLF [xxx\_]database-name [uic-group-number]  
[AT node-name]**

---

---

**[xxx\_]database-name**

**Description** *Required.* Specifies the database for which the System Log File is to be dumped.

**Format** *xxx\_* 1–3 alphanumeric characters followed by “\_” (the optional database prefix)

*database-name* 6 alphanumeric characters

**Consideration** The PDM does not allow you to dump an active System Log File if after-image records are still being logged to it. Use the UNLOAD command (“[Unloading a database \(UNLOAD\)](#)” on page 169) or the DISABLE command (“[Disabling a database \(DISABLE\)](#)” on page 151) to sign off these tasks.

---

**[uic-group-number]**

**Description** *Optional.* Specifies the UIC group number if more than one database of the same name is loaded in different groups.

**Format** 6-digit UIC group number enclosed in brackets

**Consideration** You can omit any leading zeros; however, you must enter the brackets.

---

**AT node-name**

**Description** *Optional.* Directs the command to the PDM running at the specified node.

**Format** 1–6 alphanumeric characters

**Considerations**

- ◆ If you omit this parameter, the command is directed to the local PDM.
- ◆ Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174).

## Enabling a database (ENABLE)

ENABLE cancels the restriction imposed by the DISABLE command (“Disabling a database (DISABLE)” on page 151), allowing tasks to reload a previously disabled database.

---

```
ENABLE { [xxx_]database - name [[uic - group - number]]
        { ALL
        { STATISTICS
```

```
[ AT node - name ]
```

---

```
{ [xxx_]database - name [[uic - group - number]]
  { ALL
  { STATISTICS
```

|                    |                                                              |                                                                                                                                                      |
|--------------------|--------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Description</b> | <i>Required.</i> Specifies the object of the ENABLE command. |                                                                                                                                                      |
| <b>Format</b>      | <i>xxx_</i>                                                  | 1–3 alphanumeric characters followed by “_” (the optional database prefix)                                                                           |
|                    | <i>database-name</i>                                         | 6 alphanumeric characters                                                                                                                            |
|                    | <i>[uic-group-number]</i>                                    | 1–6 digit number enclosed in brackets                                                                                                                |
| <b>Options</b>     | <i>[xxx_]database-name</i>                                   | Cancels the restriction imposed by the DISABLE command on the specified database.                                                                    |
|                    | <i>[[uic-group-number]]</i>                                  | UIC group number                                                                                                                                     |
|                    | ALL                                                          | Deassigns the logical name DISABLED_ <i>global_section-name</i> , allowing the PDM to load any database.                                             |
|                    | STATISTICS                                                   | Sends PDM statistics output to the PDM log file, any message reading mailbox, and the CSIOPCOM screen. Does not send output to the operator console. |

**Consideration** The *uic-group-number* parameter specifies a UIC group number if more than one database of the same name is loaded in different groups. You specify the UIC group number to identify the logical name table the PDM must search in order to locate the compiled database description file. You can omit any leading zeros; however, you must enter the brackets.

**AT node-name**

**Description** *Optional.* Directs the command to the PDM running at the specified node. Omit the AT keyword to pass the command to the local PDM.

**Format** 1–6 alphanumeric characters

**Consideration** Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174).

## Populating an index (POPULATE)

POPULATE populates and activates one or more index files by reading records from the data file and writing corresponding index records to the index file. Once you have run POPULATE, the index is ready for use.

---

**POPULATE *index* [xxx\_]database-name [[uic - group - number]]**  
**[AT node - name]**

---

---

### *index*

**Description**    *Required.* Identifies the index to be populated.

**Format**        *dsetlXyy*

where

*dset* is the 4 alphanumeric character data set name

*IX* is entered as shown

*yy* is the 2 alphanumeric character index name.

### **Considerations**

- ◆ POPULATE always formats a new index file before writing the index records. This minimizes the risk of inconsistencies between index files and their data files. If you defined a shadow index, POPULATE also formats a new shadow index file before writing the index records to it.
- ◆ If population of the main index file fails, both the main and the shadow index files are marked as invalid. If the population succeeds for the main index file, but fails for the shadow file, only the shadow file is marked as invalid and PDM indexing processing can proceed on the main index file alone.

---

**[xxx\_]database-name**

|                    |                                                                        |                                                                            |
|--------------------|------------------------------------------------------------------------|----------------------------------------------------------------------------|
| <b>Description</b> | <i>Required.</i> Specifies the database in which the index is defined. |                                                                            |
| <b>Format</b>      | <i>xxx_</i>                                                            | 1–3 alphanumeric characters followed by “_” (the optional database prefix) |
|                    | <i>database-name</i>                                                   | 6 alphanumeric characters                                                  |

---

**[uic-group-number]**

|                      |                                                                                                                           |  |
|----------------------|---------------------------------------------------------------------------------------------------------------------------|--|
| <b>Description</b>   | <i>Optional.</i> Specifies the UIC group number if more than one database of the same name is loaded in different groups. |  |
| <b>Format</b>        | 1–6 digit UIC group number enclosed in brackets                                                                           |  |
| <b>Consideration</b> | You can omit any leading zeros; however, you must enter the brackets.                                                     |  |

---

**AT node-name**

|                    |                                                                                |  |
|--------------------|--------------------------------------------------------------------------------|--|
| <b>Description</b> | <i>Optional.</i> Directs the command to the PDM running at the specified node. |  |
| <b>Format</b>      | 1–6 alphanumeric characters                                                    |  |

**Considerations**

- ◆ If you omit this parameter, the command is directed to the local PDM.
- ◆ Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see [“Communicating with the SUPRA Server PDM through CSIOPCOM”](#) on page 174).

## Specifying read-only access for a database (READONLY)

READONLY sets the specified database or all databases to read-only access, dynamically signs off all active update tasks in the database(s), and unloads the specified database(s). See also the UNLOAD command (“Unloading a database (UNLOAD)” on page 169) and the UPDATE command (“Specifying update access for a database (UPDATE)” on page 171).

---

```

READONLY { /COMIT
           /SINOF
           /FORCE }

          { [xxx_]database - name [uic - group - number] }
          { ALL }

          [AT node - name ]

```

---



---

```

{ /COMIT
  /SINOF
  /FORCE }

```

|                    |                                                                                                                                                                                                                                                                                                |        |                                                     |        |                                           |        |                                                                     |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|-----------------------------------------------------|--------|-------------------------------------------|--------|---------------------------------------------------------------------|
| <b>Description</b> | <i>Required.</i> Specifies how tasks are to be signed off.                                                                                                                                                                                                                                     |        |                                                     |        |                                           |        |                                                                     |
| <b>Options</b>     | <table> <tr> <td>/COMIT</td> <td>Signs off each task after its next COMMIT or RESET.</td> </tr> <tr> <td>/SINOF</td> <td>Waits until the last task has signed off.</td> </tr> <tr> <td>/FORCE</td> <td>Resets any uncommitted updates and signs off each task immediately.</td> </tr> </table> | /COMIT | Signs off each task after its next COMMIT or RESET. | /SINOF | Waits until the last task has signed off. | /FORCE | Resets any uncommitted updates and signs off each task immediately. |
| /COMIT             | Signs off each task after its next COMMIT or RESET.                                                                                                                                                                                                                                            |        |                                                     |        |                                           |        |                                                                     |
| /SINOF             | Waits until the last task has signed off.                                                                                                                                                                                                                                                      |        |                                                     |        |                                           |        |                                                                     |
| /FORCE             | Resets any uncommitted updates and signs off each task immediately.                                                                                                                                                                                                                            |        |                                                     |        |                                           |        |                                                                     |

---

```
{[xxx_]database - name [[uic - group - number]]}
{ALL}
```

|                    |                                                                                            |                                                                            |
|--------------------|--------------------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| <b>Description</b> | <i>Required.</i> Specifies whether one or all databases are to be set to read-only access. |                                                                            |
| <b>Format</b>      | <i>xxx_</i>                                                                                | 1–3 alphanumeric characters followed by “_” (the optional database prefix) |
|                    | <i>database-name</i>                                                                       | 6 alphanumeric characters                                                  |
|                    | <i>[[uic-group-number]]</i>                                                                | 1–6 digit number enclosed in brackets                                      |
| <b>Options</b>     | <i>[xxx_]database-name</i>                                                                 | Sets the specified database to read-only.                                  |
|                    | <i>[[uic-group-number]]</i>                                                                | UIC group number                                                           |
|                    | ALL                                                                                        | Sets all databases to read-only.                                           |

### Considerations

- ◆ The *uic-group-number* parameter specifies a UIC group number if more than one database of the same name is loaded in different groups. You can omit any leading zeros; however, you must enter the brackets.
- ◆ Use the UPDATE command to cancel the READONLY restriction.
- ◆ If you use the ALL parameter with the READONLY command, you must also specify the ALL parameter with the UPDATE command. The SUPRA Server PDM does not allow you to set all databases to read only access and then cancel the restriction for a specified database. Likewise, if you set a single database to READONLY, you cannot cancel that restriction with the UPDATE ALL command; the UPDATE command must specify the same single database as was specified in the READONLY command.

## **AT *node-name***

**Description**     *Optional.* Directs the command to the PDM running at the specified node.

**Format**            1–6 alphanumeric characters

### **Considerations**

- ◆ If you omit this parameter, the command is directed to the local PDM.
- ◆ Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174).

## Shutting down a database (SHUTDOWN)

SHUTDOWN signs off all tasks, unloads all loaded databases, and then terminates the PDM process. Any connected tasks get an ENDT status. SHUTDOWN implicitly disables automatic restart. However, automatic start-up will still occur if a new task attempts to access the PDM. You can start up the PDM manually if you wish. See “[Understanding the Physical Data Manager \(PDM\)](#)” on page 31 for a description of PDM initiation procedures.

---

```
SHUTDOWN { /COMIT
           /SINOF } pdmname [AT node - name]
           /FORCE
```

---

```
{ /COMIT
  /SINOF
  /FORCE }
```

|                    |                                                            |                                                                     |
|--------------------|------------------------------------------------------------|---------------------------------------------------------------------|
| <b>Description</b> | <i>Required.</i> Specifies how tasks are to be signed off. |                                                                     |
| <b>Options</b>     | /COMIT                                                     | Signs off each task after its next COMIT or RESET.                  |
|                    | /SINOF                                                     | Waits until the last task has signed off.                           |
|                    | /FORCE                                                     | Resets any uncommitted updates and signs off each task immediately. |

### Considerations

- ◆ Any task attempting a database sign-on will reinitialize the PDM provided automatic PDM initiation is enabled.
- ◆ The PDM name is the translation of CSI\_PDMID for groupwide or systemwide PDMs, or CSI\_SYSPDMID for multiple systemwide PDMs.

---

### *pdmname*

|                    |                                       |
|--------------------|---------------------------------------|
| <b>Description</b> | <i>Required.</i> The name of the PDM. |
| <b>Format</b>      | 1–8 alphanumeric characters           |

## **AT *node-name***

**Description**     *Optional.* Directs the command to the PDM running at the specified node.

**Format**            1–6 alphanumeric characters

### **Considerations**

- ◆ If you omit this parameter, the command is directed to the local PDM.
- ◆ Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174).

## Unloading a database (UNLOAD)

UNLOAD unloads the specified database after first signing off each task using it. The database can be reloaded by any subsequent task attempting a sign-on. If you wish to unload the database and prevent any new tasks from reloading it, use the DISABLE command ("[Disabling a database \(DISABLE\)](#)" on page 151).

---

```
UNLOAD { /COMIT
        /SINOF
        /FORCE }
      { [xxx_]database - name [uic - group - number] }
      ALL }
[AT node - name]
```

---

```
{ /COMIT
  /SINOF
  /FORCE }
```

|                    |                                                                                                                                                                                                                                                                                                |        |                                                     |        |                                           |        |                                                                     |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|-----------------------------------------------------|--------|-------------------------------------------|--------|---------------------------------------------------------------------|
| <b>Description</b> | <i>Required.</i> Specifies how tasks are to be signed off.                                                                                                                                                                                                                                     |        |                                                     |        |                                           |        |                                                                     |
| <b>Options</b>     | <table> <tr> <td>/COMIT</td> <td>Signs off each task after its next COMMIT or RESET.</td> </tr> <tr> <td>/SINOF</td> <td>Waits until the last task has signed off.</td> </tr> <tr> <td>/FORCE</td> <td>Resets any uncommitted updates and signs off each task immediately.</td> </tr> </table> | /COMIT | Signs off each task after its next COMMIT or RESET. | /SINOF | Waits until the last task has signed off. | /FORCE | Resets any uncommitted updates and signs off each task immediately. |
| /COMIT             | Signs off each task after its next COMMIT or RESET.                                                                                                                                                                                                                                            |        |                                                     |        |                                           |        |                                                                     |
| /SINOF             | Waits until the last task has signed off.                                                                                                                                                                                                                                                      |        |                                                     |        |                                           |        |                                                                     |
| /FORCE             | Resets any uncommitted updates and signs off each task immediately.                                                                                                                                                                                                                            |        |                                                     |        |                                           |        |                                                                     |

```
{[xxx_]database - name [[uic - group - number]]}
{ALL}
```

|                    |                                                                             |                                                                            |
|--------------------|-----------------------------------------------------------------------------|----------------------------------------------------------------------------|
| <b>Description</b> | <i>Required.</i> Specifies whether one or all databases are to be unloaded. |                                                                            |
| <b>Format</b>      | <i>xxx_</i>                                                                 | 1–3 alphanumeric characters followed by “_” (the optional database prefix) |
|                    | <i>database-name</i>                                                        | 6 alphanumeric characters                                                  |
|                    | <i>[[uic-group-number]]</i>                                                 | 1–6 digit number enclosed in brackets                                      |
| <b>Options</b>     | <i>[xxx_]database-name</i>                                                  | Unloads the specified database.                                            |
|                    | <i>[[uic-group-number]]</i>                                                 | UIC group number                                                           |
|                    | ALL                                                                         | Unloads all databases.                                                     |

**Consideration** The *uic-group-number* parameter specifies a UIC group number if more than one database of the same name is loaded in different groups. You can omit any leading zeros; however, you must enter the brackets.

---

### AT *node-name*

**Description** *Optional.* Directs the command to the PDM running at the specified node.

**Format** 1–6 alphanumeric characters

#### Considerations

- ◆ If you omit this parameter, the command is directed to the local PDM.
- ◆ Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174).

## Specifying update access for a database (UPDATE)

UPDATE cancels the READONLY command and sets databases to allow UPDATE access.

---

```
UPDATE { [xxx_]database - name [[uic - group - number]] }
      { ALL }
```

[ AT node - name ]

---



---

```
{ [xxx_]database - name [[uic - group - number]] }
{ ALL }
```

|                    |                                                                                         |                                                                            |
|--------------------|-----------------------------------------------------------------------------------------|----------------------------------------------------------------------------|
| <b>Description</b> | <i>Required.</i> Specifies whether one or all databases are to be set to UPDATE access. |                                                                            |
| <b>Format</b>      | <i>xxx_</i>                                                                             | 1–3 alphanumeric characters followed by “_” (the optional database prefix) |
|                    | <i>database-name</i>                                                                    | 6 alphanumeric characters                                                  |
|                    | <i>[[uic-group-number]]</i>                                                             | 1–6 digit number enclosed in brackets                                      |
| <b>Options</b>     | <i>[xxx_]database-name</i>                                                              | Allows update access to the specified database.                            |
|                    | <i>[[uic-group-number]]</i>                                                             | UIC group number                                                           |
|                    | ALL                                                                                     | Allows update access to all databases.                                     |

## Considerations

- ◆ The *uic-group-number* parameter specifies a UIC group number if more than one database of the same name is loaded in different groups. You can omit any leading zeros; however, you must enter the brackets.
- ◆ The UPDATE command does not load databases unloaded by the READONLY command.
- ◆ If you set all databases to read-only mode using READONLY ALL, you must use UPDATE ALL to cancel this restriction. If you set a specified database to read-only mode, you must explicitly specify that database with the UPDATE command to cancel the restriction. UPDATE ALL permits updates on only those databases affected by READONLY ALL.

For example, after the following sequence of commands through the CSIOPCOM interface, the database TESTDB remains in read-only mode:

```
==>READONLY/COMIT TESTDB[125]  
==>READONLY/FORCE ALL  
==>UPDATE ALL
```

After the UPDATE ALL, the PDM displays the following message:

```
CSTI156I All databases are now available for update 14
```

However, the database TESTDB remains in read-only mode. This is because the UPDATE ALL command affects only those databases unloaded by the READONLY ALL command. TESTDB, having already been set to READONLY, is affected by neither READONLY ALL nor UPDATE ALL. To cancel the read-only restriction on TESTDB, specify the database name with the UPDATE command as follows:

```
==>UPDATE TESTDB[125]
```

**AT node-name**

**Description**     *Optional.* Directs the command to the PDM running at the specified node.

**Format**            1–6 alphanumeric characters

**Considerations**

- ◆ If you omit this parameter, the command is directed to the local PDM.
- ◆ Use this parameter only if you are communicating with a remote PDM through the CSIOPCOM interface (see “[Communicating with the SUPRA Server PDM through CSIOPCOM](#)” on page 174).

## Communicating with the SUPRA Server PDM through CSIOPCOM

CSIOPCOM is a screen-based user interface through which you can enter PDM operator commands. In addition to context-related online help, pop-up menus from which to select PDM operator commands with a single keystroke, and function key support, CSIOPCOM offers:

- ◆ Two CSIOPCOM commands: LIST and SET
- ◆ A batch interface to the PDM

CSIOPCOM allows you to communicate with any PDM on the network. Therefore, you must always specify the [AT *node-name*] parameter to the PDM operator command to access a remote PDM.

No special privileges are needed to use CSIOPCOM. However, you should ensure that each user has access to a user authorization file through the logical name CSIOPCOM\_AUTH. You create user authorization files using CSIOAUTH (see “[Restricting use of PDM commands](#)” on page 185). Without a user authorization file, CSIOPCOM users with SYSRV and OPER privileges can access all PDM operator commands. CSIOPCOM users without these privileges can access only the DISPLAY command.

To run CSIOPCOM and issue SUPRA Server PDM operator commands, type

```
$RUN CSIOPCOM
```

The initial screen appears:

```
SUPRAPDM Operator Interface 2.4
```

```
-----  
                (C) Cincom Systems, Inc. 1992.  
                All Rights Reserved.
```

```
Use of this software is governed by a license  
agreement. This software contains confidential  
and proprietary information of Cincom Systems,  
Inc. which is protected by copyright, trade  
secret, and trademark law.
```

```
==>
```

```
<PF1>=Refresh <PF2>=Help <PF3>=List Cnds <PF4>=Screen Dump <CTRL/Z>=Exit
```

You can enter any authorized PDM operator command on the command line at the bottom of the screen. For example:

```
SHUTDOWN /COMIT TESTPDM AT VMS2
```

will shut down the PDM named TESTPDM on node VMS2 when the last task is signed off. Each task is signed off on the next COMIT.

The CSIOPCOM interface supports the function keys shown in the following table. The function keys are displayed on the bottom of the screen.

| Function key | Description                                                                       |
|--------------|-----------------------------------------------------------------------------------|
| 1            | Refreshes the screen display.                                                     |
| 2            | Displays related online help.                                                     |
| 3            | Lists the PDM commands you are authorized to use in a pop-up menu.                |
| 4            | Dumps the current screen to a file identified by the logical name CSIOPCOM_SNAPS. |

Press CTRL-Z to exit to the DCL command level.

**Refreshing the screen.** Press function key PF1 to refresh the screen and delete any characters from the command line.

**Displaying online help.** If you have defined the logical name SUPRA\_HELP (which is automatically defined by LOGICALS.COM), you can press function key PF2 at any stage during CSIOPCOM processing to display context-related help. For example, if you type DISPLAY at the command line and then press function key PF2, the following help screen displays:

```
SUPRAPDM Operator Interface 2.4
-----
CSIOPCOM

  DISPLAY

  Displays the current status of specified databases or tasks,
  listing the following details on the screen according to the
  parameters and qualifiers you specify:

      o Database name (6, 8, 9 or 10 characters)
      o If the database is loaded in system-wide (S) or group-wide
        (G) global sections
      o UIC group number (6 digits, leading zeros ignored)
      o The number of bytes in memory taken up by the database
      o The state of the database, Fail, Inactive or Active
      o The number of active tasks
      o The number of active functions

Press RETURN to continue...
==> DISPLAY
  <PF1>=Refresh <PF2>=Help <PF3>=List Cnds <PF4>=Screen Dump <CTRL/Z>=Exit
```

Use the online help if you are unsure of the syntax of a PDM command.

**Displaying a pop-up menu listing PDM commands.** The DBA can use CSIOAUTH to restrict the PDM commands available to certain users. Press function key PF3 to display the PDM commands that you have been authorized to use in a pop-up menu.

```
SUPRAPDM Operator Interface 2.4
-----
+-----+
| Valid SUPRAPDM          |
|   commands are :      |
|                         |
|   A - DISPLAY          |
|   B - UNLOAD           |
|   C - SHUTDOWN        |
|   D - READONLY        |
|   E - UPDATE           |
|   F - DISABLE         |
|   G - ENABLE          |
|   H - DUMPSLF         |
|   I - PRINT           |
|   J - ACTIVATE        |
|   K - DEACTIVATE      |
|   L - POPULATE        |
| Select option letter or |
| press RETURN to exit : |
+-----+

==>
<PF1>=Refresh <PF2>=Help <PF3>=List Ccmds <PF4>=Screen Dump <CTRL/Z>=Exit
```

The above example gives access to all the PDM operator commands. Type the key letter (A through L) to select a PDM command. Do not press RETURN. CSIOPCOM displays the command keyword (DISPLAY, UNLOAD, SHUTDOWN, etc.) at the command line ready for you to type any qualifiers and parameters. When you press RETURN after entering a PDM command, the screen clears, and the output from the command is shown at the top left of the display area.

**Dumping CSIOPCOM screens.** To enable the screen dump facility, define the logical name CSIOPCOM\_SNAPS to point to a valid VMS file specification:

```
$ DEFINE CSIOPCOM_SNAPS DUA3:[DOCS.PDM]CSIOPCOM_SNAPS.LIS
```

If you then press function key PF4, CSIOPCOM dumps the current screen to the file CSIOPCOM\_SNAPS.LIS in DUA3:[DOCS.PDM]. If you press function key PF4 without having first defined the logical name CSIOPCOM\_SNAPS, CSIOPCOM writes the screen image to a file called CSIOPCOM\_SNAPS.LIS in your default directory.

CSIOPCOM creates only one CSIOPCOM\_SNAPS file during a single session. If you press function key PF4 more than once, CSIOPCOM appends each screen to the end of the CSIOPCOM\_SNAPS file.

If you dump screens during several CSIOPCOM sessions without redefining CSIOPCOM\_SNAPS, CSIOPCOM creates a higher version of the CSIOPCOM\_SNAPS file for each session from which you dump one or more screens.



---

You can define CSIOPCOM\_SNAPS in the PDM\_LOGICALS\_\*.COM procedure. See "[PDM\\_LOGICALS\\_\\*.COM](#)" on page 87.

---

## Using CSIOPCOM commands

CSIOPCOM supports two additional commands: LIST and SET. These commands are *not* for communication with the PDM, although their syntax is similar to that of PDM commands.

---

### LIST

---

**Description** Displays all authorized SUPRA Server PDM commands in a window from which you can select a command.

### Considerations

- ◆ Do not use this command when running in batch.
- ◆ This command works only from the CSIOPCOM interface, not from VMS OPCOM. It is equivalent to pressing function key PF3.

The SET command sends input to and output from CSIOPCOM to both the terminal and a disk file, or to a disk file only.

---

SET {  
LOG *file - spec*  
NOLOG  
OUTPUT *file - spec*  
NOOUTPUT  
}

---

```

{ LOG file - spec
  NOLOG
  OUTPUT file - spec
  NOOUTPUT
}

```

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                      |                      |                                                                                         |       |                     |                         |                                                                                        |          |                                   |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-----------------------------------------------------------------------------------------|-------|---------------------|-------------------------|----------------------------------------------------------------------------------------|----------|-----------------------------------|
| <b>Description</b>      | <i>Required.</i> Specifies the disposition of CSIOPCOM logging.                                                                                                                                                                                                                                                                                                                                                                      |                      |                                                                                         |       |                     |                         |                                                                                        |          |                                   |
| <b>Format</b>           | <i>file-spec</i> VMS file specification                                                                                                                                                                                                                                                                                                                                                                                              |                      |                                                                                         |       |                     |                         |                                                                                        |          |                                   |
| <b>Options</b>          | <table> <tr> <td>LOG <i>file-spec</i></td> <td>Logs all input and output to the specified file and displays it on the terminal screen.</td> </tr> <tr> <td>NOLOG</td> <td>Terminates logging.</td> </tr> <tr> <td>OUTPUT <i>file-spec</i></td> <td>Sends all input and output to the specified file <u>only</u>, redefining SYS\$OUTPUT.</td> </tr> <tr> <td>NOOUTPUT</td> <td>Sends output back to SYS\$OUTPUT.</td> </tr> </table> | LOG <i>file-spec</i> | Logs all input and output to the specified file and displays it on the terminal screen. | NOLOG | Terminates logging. | OUTPUT <i>file-spec</i> | Sends all input and output to the specified file <u>only</u> , redefining SYS\$OUTPUT. | NOOUTPUT | Sends output back to SYS\$OUTPUT. |
| LOG <i>file-spec</i>    | Logs all input and output to the specified file and displays it on the terminal screen.                                                                                                                                                                                                                                                                                                                                              |                      |                                                                                         |       |                     |                         |                                                                                        |          |                                   |
| NOLOG                   | Terminates logging.                                                                                                                                                                                                                                                                                                                                                                                                                  |                      |                                                                                         |       |                     |                         |                                                                                        |          |                                   |
| OUTPUT <i>file-spec</i> | Sends all input and output to the specified file <u>only</u> , redefining SYS\$OUTPUT.                                                                                                                                                                                                                                                                                                                                               |                      |                                                                                         |       |                     |                         |                                                                                        |          |                                   |
| NOOUTPUT                | Sends output back to SYS\$OUTPUT.                                                                                                                                                                                                                                                                                                                                                                                                    |                      |                                                                                         |       |                     |                         |                                                                                        |          |                                   |

### Considerations

- ◆ You can enter the SET command only from the CSIOPCOM interface to the PDM. The VMS OPCOM interface does not support this function.
- ◆ The SET OUTPUT *file-spec* command is particularly useful with the DISPLAY STATISTICS PDM operator command. It allows you to send the statistics to a file without scrolling through the entire display on your terminal.

## Running CSIOPCOM in batch

To run CSIOPCOM in batch, create a command file containing the RUN command and any CSIOPCOM commands that you want to execute.

For example:

```
$!           Command File SHUTDOWN_PDMTEST.COM
$! Displays the status of all databases
$! currently running in the PDM, and then shuts down the PDM
$!
$ RUN CSIOPCOM
DISPLAY/DATABASES AT VMS780
SHUTDOWN /FORCE PDMTEST AT VMS780
$ EXIT
```

To execute the command file online, type:

```
$ @SHUTDOWN_PDMTEST.COM
```

## Automating operator communication

You can write applications to send operator commands to the PDM and to receive operator messages from various SUPRA Server components. These applications can reduce the need for manual operator intervention in administering your SUPRA Server system.

**Writing applications to send commands to the PDM.** You can send operator commands to the PDM by using OPCOM, a PDML function that allows an application to send SUPRA Server operator commands (see “Using the PDM operator commands” on page 146). For information about the use of the SUPRA Server PDML function OPCOM, refer to the *SUPRA Server PDM Programming Guide (UNIX & VMS)*, P25-0240.

**Writing applications to receive SUPRA Server messages.** SUPRA Server messages are output by the PDM, CSIDAP, and the system log dump utility (CSTUDSLF). To direct these messages to a mailbox that you can read from your program, define the following:

- ◆ The PDM input file parameter MRELAY=Y for PDM and System Log Dump utility (CSIUDSLF) messages
- ◆ The logical CSI\_MRELAY as TRUE for CSIDAP messages

The mailbox name is constructed by linking together the following:

- ◆ The translation of CSI\_PDMID for a groupwide or a systemwide PDM, or CSI\_SYSPDMID for a multiple systemwide PDM
- ◆ An underscore
- ◆ The 6-digit UIC group number for a groupwide PDM, or 000000 for a systemwide or a multiple systemwide PDM

For example, if the logical name CSI\_PDMID equates to TESTPDM, and the PDM is systemwide, the mailbox name will be TESTPDM\_000000. Alternatively, if the PDM is running in the group 000145, with CSI\_PDMID equating to QADBD, the mailbox name will be QADBD\_000145.

The number of messages SUPRA Server can send to the mailbox is determined by the SYSGEN parameters DEFMBXFUFQUO and DEFMBXMXMSG. If the PDM cannot send any more messages to the mailbox, either because there is no active mailbox-reading program, or because the program is not picking the messages up fast enough, it will keep a count of the number of messages that it was unable to send. On the next successful send, it will output the message:

```
MESSAGES LOST=nnnn
```

**Example message-reading program.** The following is a skeleton message-reading program in pseudocode. You should start your message-reading program before SUPRA Server (CSIDAP and/or CSIPDM) starts sending messages and fills up the mailbox buffer. See “[Example mailbox-reading program](#)” on page 275 for an example message-reading program written in COBOL.

```
Program Mailbox_read
Begin-code

    Construct the mailbox name

    Create a Mailbox using the name constructed, with a buffer
    quota size of 2048 and message size of 1024.

    While (ok to continue)
    begin-while

        Queue Asynchronous READ on the Mailbox by calling
        SYS$QIO and specify a completion AST - say mbx_ast.

        Hibernate or do something useful while call is being
        completed.

        Do something with the message received.
        Decide whether to continue with loop.

    End-while.

End-code.

Procedure mbx_ast

Begin-code

    ! This procedure will be executed when the call to the QIO
    ! system service has been completed, when a message
    ! is received.

    If main program is hibernating then
        Call SYS$WAKE to wake up main program.

End-code.
```

---

## Restricting use of PDM commands

SUPRA Server allows you to create files which authorize subsets of PDM OPCOM commands for one or more users. You create a user authorization file and make it available to a specified user or group of users in two stages:

1. Run CSIOAUTH and reply to the prompts to create an authorization file.
2. Define the logical name CSIOPCOM\_AUTH pointing to the authorization file created. See “[PDM\\_LOGICALS\\_\\*.COM](#)” on page 87 for details on CSIOPCOM\_AUTH.

By default, users with SYSPRV and OPER privileges have access to all PDM operator commands through CSIOPCOM if there is no logical definition for CSIOPCOM\_AUTH. Users with lower privileges can access only the DISPLAY operator command unless they have a user authorization file that gives access to other PDM operator commands.

To run the authorization file creation program, CSIOAUTH, you need both SYSPRV and OPER privileges. The following sequence of screens illustrates how to create an authorization file. Run the authorization program by entering:

```
$ RUN CSIOAUTH
```

This displays the authorization screen for the first PDM command, DISPLAY, shown below:

```
                SUPRA PDM Operator Command
                Authorization Program 2.4
-----
Permit SUPRA PDM Command <DISPLAY    > (Y/N/<CTRL>/Z) :

(c) Cincom Systems, Inc. 1992
Use of this software is governed by a license
agreement. This software contains confidential
and proprietary information of Cincom Systems,
Inc. which is protected by copyright, trade
secret, and trademark law.
```

Valid responses include:

- ◆ Y to allow a PDM command
- ◆ N to deny a PDM command
- ◆ CTRL-Z to skip to the prompt for the authorization file name or, if the authorization file name prompt is displayed, to abandon the selection.

Pressing CTRL-Z causes CSIOAUTH to deny all subsequent PDM commands. Thus, if you press CTRL-Z after having permitted the first two PDM commands, all other PDM commands will be denied.



Once you have permitted or denied the last SUPRA Server PDM command, CSIOAUTH prompts you to enter the file specification for the authorization file as follows:

```
SUPRA PDM Operator Command Authorization Program 2.4
-----
Enter File Specification for Authorization data:

----Commands permitted-----  -----Commands denied-----
DISPLAY		SHUTDOWN
UNLOAD		READONLY
		UPDATE
		DISABLE
		ENABLE
		DUMPSLF
		PRINT
-----
```

Enter a valid VMS file specification, including directory specification if necessary. After you have entered the file specification, CSIOAUTH exits to the DCL command level. See “[PDM\\_LOGICALS\\_\\*.COM](#)” on page 87 for details on defining the logical CSIOPCOM\_AUTH to point to the authorization file.

---

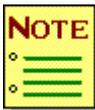
## Communicating with the PDM through the VMS REPLY command

You can use the VMS REPLY operator command to communicate with the PDM by following these steps:

1. Set the PDM input parameter SYSOPCOM=Y to display the VMS OPCOM prompt “Reply with a SUPRA PDM Command” at regular intervals at enabled PDM operator terminals.
2. Set the PDM input parameter OPERATOR=OPERn, where n is an operator number in the range of 1 to 12, to identify the PDM operator ID.
3. Enable one or more operator terminals by entering the following VMS command: REPLY/TEMPORARY/ENABLE=OPERn, where n is the operator number specified with the PDM input parameter OPERATOR=. The OPER privilege is required for this command.

The PDM then sends all messages to the operator console. It sends messages that do not have a severity of “L” to the PDM operator ID specified in the PDM input file. You can specify only one operator ID per PDM (for example, OPERATOR=OPER9); however, you can enable as many operator terminals per operator number as you wish.

At least one operator terminal should be enabled for the PDM at all times. Should all operator terminals be disabled, VMS OPCOM will cancel the outstanding Reply request, disabling communication with the detached PDM process. If this occurs, you can reestablish communication by causing the PDM to send another operator message. A database load, for example, will cause the PDM to reestablish operator communication with VMS OPCOM.



---

You can always communicate with the PDM using CSIOPCOM.

---

The Database Access Program, identified by the logical CSIDAP, also displays messages on the central operator console. To ensure that messages from the Database Access Program go to the same operator terminals as those from the central PDM image, assign the logical name CSI\_CONSOLE to the operator ID used for the PDM messages. The operator ID used for PDM messages is set using the PDM input parameter CONSOLE. See “PDM\_LOGICALS\_\*.COM” on page 87 for details on defining CSI\_CONSOLE. See “Entering parameters for the PDM input file” on page 125 for details on setting the PDM input parameter CONSOLE.

You can prevent the PDM operator messages from appearing on a terminal by entering REPLY/DISABLE=OPERn or setting the PDM input parameters CONSOLE and SYSOPCOM to N.

When the PDM input parameter SYSOPCOM=Y, VMS OPCOM issues operator prompts at enabled operator terminals at regular intervals (5 or 10 minutes). The following is an example operator prompt:

```

%%%%%%%%%% OPCOM 15-Nov-1996 11:32:46.43 %%%%%%%%%%
Request 143, from user CONTROLLER on VMS1
CSTI0060 QAPROC, Reply with a SUPRA PDM command.
    
```

Enter the PDM operator commands in response to the operator prompts using the following format:

---

REPLY {/PEND = } request-no "pdm-operator-command "

---



---

{/PEND = }  
{/TO = }

**Description** *Required.* Specifies the disposition of PDM operator command requests.

**Options** /PEND= Keeps the request in a wait state, thereby retaining the same message number.

/TO= Completes the request and causes a new number to be generated for the next message.

**Consideration** Refer to the VMS documentation for details of this command.

---

**request-no**

**Description**    *Required.* The message number to which you are replying.

**Format**        1–n digits

---

**"pdm-operator command "**

**Description**    *Optional.* A PDM operator command. (See [“Using the PDM operator commands”](#) on page 146 for descriptions of all PDM operator commands.)

**Format**        Must be enclosed in double quotation marks.

**General consideration**

Do not use the [AT *node-name*] parameter with PDM operator commands that are issued through VMS OPCOM, because the request number you specify uniquely identifies the PDM.

The following example illustrates a sample operator session in which the operator enters a selection of PDM operator commands. Bold type indicates user input, normal type indicates system response.

```

$
%%%%%%%%%%%% OPCOM  9-JUN-1996 16:29:45.55  %%%%%%%%%%%%%
Request 200, from user ROGERS on CORP1
CSTI0060 MARK22, Reply with a SUPRA PDM command.

$ REPLY/TO=200 "DISPLAY/DATABASES ALL"
DISPLAY/DATABASES ALL
16:35:46.53, request 200 was completed by operator _CORP1$TNA85:

%%%%%%%%%%%% OPCOM  9-JUN-1996 16:35:46.54  %%%%%%%%%%%%%
Message from user ROGERS on CORP1
CSTI064R      NAME  G/S  GROUP    SIZE  STATE  TASKS  FUNCTIONS
  THREADS
              MRPDBO G  000140  458716   A      1      0
0

%%%%%%%%%%%% OPCOM  9-JUN-1996 16:35:46.54  %%%%%%%%%%%%%
Request 201, from user ROGERS on CORP1
CSTI0060 MARKR22, Reply with a SUPRA PDM command.

```

**\$ REPLY/PEND=201 "DISPLAY/TASKS ALL"**

```
%%%%%%%%%% OPCOM 9-JUN-1996 16:36:05.07 %%%%%%%%%%
Message from user ROGERS on CORP1
CSTI061R MRPDBO(AU-000140) TASKNAME TASKID MODE
STATE
_TNA81: 0002 U I
```

**\$ REPLY/PEND=201 "SHUTDOWN/FORCE MARKR22"**

```
$
%%%%%%%%%% OPCOM 9-JUN-1996 16:36:21.98 %%%%%%%%%%
Message from user ROGERS on CORP1
CSTI022E MRPDBO[000140], _TNA81:, 0002, *DBTEST*,
task dynamically signed off - reason code is 14.
```

```
$
%%%%%%%%%% OPCOM 9-JUN-1996 16:36:22.42 %%%%%%%%%%
Message from user ROGERS on CORP1
CSTI004I MARKR22, Data Base MRPDBO[000140] unloaded.
```

```
$
%%%%%%%%%% OPCOM 9-JUN-1996 16:36:22.42 %%%%%%%%%%
Message from user ROGERS on CORP1
CSTI002I MARKR22, SUPRA PDM release 2.4 terminated.
```

```
$
```

# 6

## Setting up the SUPRA Server Directory database

The DBA accesses the SUPRA Server Directory database to define and maintain database descriptions using these facilities:

- ◆ **DBA functions.** Database administrators use the DBA functions to define and maintain:
  - The SUPRA Server Directory database itself (SUPRAD)
  - User database and data set descriptions stored on the SUPRA Server Directory database
  - Logical data items stored on the SUPRA Server Directory database
  - Logical views stored on the SUPRA Server Directory database

In addition, the Directory controls:

- Database users
- Access rights
- Enrolling of RDML programs

The DBA utility also provides the following tools:

- Recovery functions
- DBA utilities

Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260, for details on how to use the DBA functions. Refer to the *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220, for details on DBA utilities.

- ◆ **Fast utilities.** The DBA or system administrator uses Fast utilities to make physical changes to databases, including the Directory database SUPRAD. You can use Fast utilities both online and in batch. Refer to the *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220, for details on Fast utilities.
- ◆ **Batch validate, compile, and print.** The batch validate, compile, and print program, called COMBAT, is supplied with the DBA validate, compile, and print functions. The DBA can invoke COMBAT from within SUPRA DBA or from the command level.

COMBAT allows the DBA to submit a database validate, compile, and/or print function to the batch queue. Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260, for an explanation of the batch validate, compile, and print functions.

- ◆ **Format data sets.** The format program, called CSTUFMT, runs from SUPRA DBA or from the command level. Format creates and formats the physical files that will hold the data.
- ◆ **Format, populate, and check indices.** The index facilities program identified by the logical CSTUIDX runs both from DBA and from the command level. CSTUIDX formats the physical files that will hold the index records and populates these files from data already held on your data files. Once populated, you need only activate your index files to write an index record each time you write a data record. The check function verifies index files that have been deactivated for any length of time. The check function will correct any inaccurate index records.

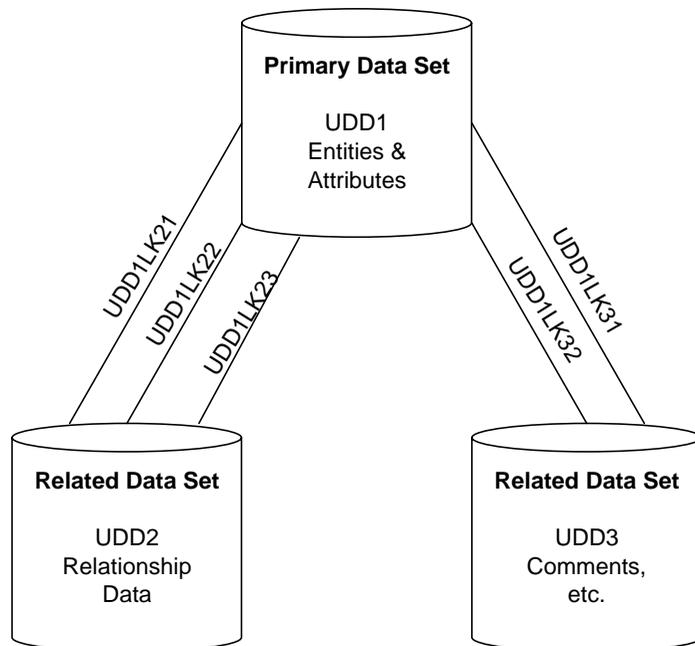
## The SUPRA Server Directory database

The SUPRA Server Directory is stored on the database SUPRAD. SUPRAD consists of the following three data sets and task log:

- ◆ UDD1—A primary data set holding entities and some attributes
- ◆ UDD2—A related data set holding relationships between entities
- ◆ UDD3—A related data set holding comment information, bound view data, and view access definitions
- ◆ UDDTLOG—Task log for roll back recovery holding before images

You can optionally define a system log for roll forward recovery for the SUPRA Server Directory.

The following figure shows the structure of the SUPRAD database in more detail. The lines in the figure represent linkpaths between the data sets. Specify the primary linkpaths, UDD1LK21 and UDD1LK31, when you run UNLOAD/RELOAD or when you run the Statistics utility from DBA utilities.



## Estimating the SUPRA Server Directory data set sizes

The Directory data sets, like your private data sets, operate more efficiently if they are sized properly. Use the following general guidelines to estimate the sizes of the Directory data sets for your system:

- ◆ Each entity requires two records on UDD1, one record on UDD2, and one record on UDD3.
- ◆ Each relationship requires one record on UDD2. Each entity has at least one relationship; many entities have more. You can estimate the number of relationships as the number of entities times four.
- ◆ UDD2 requires two and a half times as many records as UDD1.
- ◆ Each line of comment or navigation definition requires one record on UDD3.
- ◆ The Directory contains special information used by Directory maintenance and the definition of SUPRAD. Allow for this in your estimates as shown in the following table:

| Directory data set | Records storing directory maintenance information | Records storing SUPRAD definition | Total records required by data set | Total records available at install | Blocks required |
|--------------------|---------------------------------------------------|-----------------------------------|------------------------------------|------------------------------------|-----------------|
| UDD1               | 500                                               | 200                               | 700                                | 5000                               | 1200            |
| UDD2               | 1100                                              | 450                               | 1550                               | 10020                              | 3120            |
| UDD3               | 2500                                              | 300                               | 2800                               | 8000                               | 2048            |

## Setting up the SUPRA Server Directory user names

The Directory contains initial user names for your use. The following two user names, both of which have blank passwords, are included:

- ◆ **DATABASE-DESCRIPTIONS**—Accesses all the databases, views, and so on, you will define.
- ◆ **DATA-DICTIONARY**—Accesses only the definition of the Directory, SUPRAD.

Sign on to DBA with the user name DATABASE-DESCRIPTIONS to create less privileged user names for your staff, then change the passwords of the two initial user names. Note that certain utilities may require you to reset the password on either or both of the above user names to the original default of blank.

The Directory also contains the user name PUBLIC. When Relational Data Manager (RDM) programs are preprocessed, PUBLIC enrolls the programs onto the Directory. Program enrollment provides the DBA with a record of when the program was last modified and how many times it has been preprocessed. If you choose to remove the user name PUBLIC, you can still preprocess and run programs, but no record of them is made in the SUPRA Server Directory.

The user name UTILITIES is used only to execute the FORMAT, DBA utilities, or RECOVERY functions. This user name is not stored on the Directory. You must use the user name UTILITIES when running the FORMAT, DBA utilities, or RECOVERY functions against SUPRAD. You will learn the password at installation. Since this user name is not stored on the Directory, you cannot change the password.

In some cases the UTILITIES username cannot be used to format the TASKLOG for SUPRAD. In those cases, you must be sure SUPRAD.MOD is unloaded from the PDMs memory and then invoke format directly from the DCL prompt:

```
$ RUN CSIOPCOM
UNLOAD/FORCE SUPRAD
$ RUN CSTUFMT
```

## Changing the definition of the SUPRA Server Directory database

You can change certain parts of the definition of the Directory database, SUPRAD. This database is defined on the Directory and you must sign on to DBA with the user name DATA-DICTIONARY to modify it. You cannot alter the structure of SUPRAD, but you can change the following characteristics:

- ◆ Data set size(s)
- ◆ File specifications
- ◆ File shadowing
- ◆ System logging
- ◆ The number of buffers
- ◆ The maximum number of update tasks allowed

After you have been running the SUPRA Server Directory for some time, you may need to modify the Directory definition if:

- ◆ A LOAD status occurs when you sign off from DBA. This indicates that UDD2 and/or UDD3 have reached their load limit.
- ◆ A FULL status occurs. This indicates that at least one of the UDD files has no free space left.

## Creating a recovery point

A recovery point is a point from which a database can be recovered after changes have been made. Before making any changes to SUPRAD, you must create a good recovery point. Situations that require a recovery point include:

- ◆ Changing database metadata; for example, changing file size, adding data items, changing lengths of data items, deleting data items, and resetting load limits on data sets.
- ◆ Compiling a database description, such as SUPRAD.MOD.
- ◆ Moving your compiled database description to a different disk or directory location, if system logging is defined.
- ◆ Reorganizing the data using either DBA utilities or Fast utilities. Refer to the *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220.
- ◆ Running applications that do not do task logging.
- ◆ Establishing a base for recovery after a device failure.

To create a recovery point for your SUPRA Server Directory database, follow these steps:

1. Unload SUPRAD from the PDM.
2. Dump the SUPRAD System Log Files, if any.
3. Format the SUPRAD System Log Files, if any.
4. Use the VMS Backup utility to back up all SUPRAD files, including:
  - Compiled database description (SUPRAD.MOD)
  - Data sets (UDD1.CSI, UDD2.CSI, UDD3.CSI)
  - Task log (UDDTLOG.CSI)
  - System log dump files, if using system logging



---

The dumps from the system logs can be used to roll forward from the previous recovery point up to the current recovery point.

---

## Modifying the SUPRA Server Directory database

The Directory is central to the operation of SUPRA Server. Follow these steps *before* modifying your Directory:

1. Ensure that no user is signed on to the Directory running Directory reports, RDM applications, DBAID, SPECTRA (VAX only), or RDML language preprocessors.
2. Create a recovery point for the Directory database SUPRAD (see “[Creating a recovery point](#)” on page 199).

Keep a copy of a listing of the Directory database to be sure of its original details. This listing is contained in a file named SUPRAD.LIS.

To modify the Directory database (SUPRAD), take these steps:

1. Sign on as user name DATA-DICTIONARY.
2. Make your changes, referring to the table in this section for required actions.
3. Validate and compile SUPRAD.
4. Back up the new Directory-compiled database description, the UDD data sets, the UDDTLOG, and the System Log File if you are using one (create a new recovery point).



---

Compiling SUPRAD generates a new compiled database description file (SUPRAD.MOD unless you have changed the file specification). However, the SUPRA Server PDM continues to use the previous SUPRAD.MOD until it is unloaded from the PDM. When a user signs on, either implicitly (after using FORMAT) or explicitly (in an application), SUPRA Server uses the new SUPRAD.MOD.

---

If you have made changes to the definition of the task log for SUPRAD, you must format a new task log after compiling SUPRAD. This is essential if you change the task log size, the task log block size, or the value of MAXTASKS. The following table lists all possible changes to SUPRAD and the corresponding actions required after compilation.

If you have defined a system log for SUPRAD, you must format a new system log after making your Directory modifications. This is essential if you change the system log number of blocks or the system log block size. The following table lists all possible changes to SUPRAD and the corresponding actions required after compilation.

SUPRA DBA uses the Directory to operate. During maintenance, the Directory might not be valid. Therefore, to sign on to SUPRA DBA without signing on to the Directory, use the special user name UTILITIES. The user name UTILITIES gives access only to FORMAT, UTILITIES, and RECOVERY. Use these functions to re-create a valid Directory, then sign on as usual.

If you select FORMAT, UTILITIES, or RECOVERY from a user name other than UTILITIES, the system signs off for you regardless of whether you have actually run the utility. If you select another function, SUPRA Server signs on again using your original user name.

In some cases the UTILITIES username cannot be used to format the TASKLOG for SUPRAD. In those cases, you must be sure SUPRAD.MOD is unloaded from the PDMs memory and then invoke format directly from the DCL prompt:

```
$ RUN CSIOPCOM
UNLOAD/FORCE SUPRAD
$ RUN CSTUFMT
```

The following table lists the database details on the Directory that you can change and the actions you must take if you make the change:

| Parameter type     | Parameter name            | Action required as a result of the modification                                                                                                                                                                            |
|--------------------|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Database Details   | DATABASE-PASSWORD         | No further action.                                                                                                                                                                                                         |
|                    | MAX-HELD-RECORDS          | Do not reduce this value from 200.                                                                                                                                                                                         |
|                    | MAX-TASKS                 | Format a new task log. The size of the task log may change. The value it holds must match that in the compiled database description.                                                                                       |
|                    | MAX-UPDATE-TASKS          | Format a new task log.                                                                                                                                                                                                     |
|                    | SHADOW-OPTION             | If you changed this from N, make a copy of the relevant data sets, using the names used in the file specifications.                                                                                                        |
|                    | SINGLE-TASK               | If you change this to Y, SUPRA Server sets MAX-TASKS and MAX-UPDATE-TASKS to one. Therefore, format a new task log. If you changed it to N, no further action is required unless you change MAX-TASKS or MAX-UPDATE-TASKS. |
| Task Log Details   | TASK-LOG-BLOCK-SIZE       | Format a new task log.                                                                                                                                                                                                     |
|                    | TASK-LOG-NO-OF-BLOCKS     | Format a new task log.                                                                                                                                                                                                     |
|                    | TASK-LOG-FILE-SPEC        | Copy or rename the existing task log to the specified file or format a new task log.                                                                                                                                       |
|                    | TASK-LOG-SHADOW-FILE-SPEC | Copy or rename the existing task log to the specified file or format a new one.                                                                                                                                            |
| System Log Details | SYSTEM-LOG-BLOCK-SIZE     | Format a new system log.                                                                                                                                                                                                   |
|                    | SYSTEM-LOG-NO-OF-BLOCKS   | Format a new system log.                                                                                                                                                                                                   |
|                    | FILE-1-FILE-SPEC          | If just created, format a new system log. If changed, copy the existing system log or format a new one.                                                                                                                    |
|                    | FILE-2-FILE-SPEC          | If just created, format a new system log. If changed, copy the existing system log or format a new one.                                                                                                                    |

| <b>Parameter type</b>               | <b>Parameter name</b>                                                                                  | <b>Action required as a result of the modification</b> |
|-------------------------------------|--------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| System Log Details ( <i>cont.</i> ) | FILE-1-SHADOW-FILE-SPEC                                                                                | Copy or rename the system log or format a new one.     |
|                                     | FILE-2-SHADOW-FILE-SPEC                                                                                | Copy or rename the system log or format a new one.     |
| Data Set Buffers                    | NUMBER-OF-COPIES-OF-BUFFER                                                                             | No further action.                                     |
|                                     | Buffer use (which data sets share which buffers, including deleting buffers and creating new buffers). | No further action.                                     |
| Data Set Details                    | TOTAL-LOGICAL-RECORDS                                                                                  | Unload and reload data set.                            |
|                                     | LOGICAL-RECORDS-PER-BLOCK                                                                              | Unload and reload data set.                            |
|                                     | CONTROL-INTERVAL                                                                                       | Unload and reload data set.                            |
|                                     | LOAD-LIMIT                                                                                             | Unload and reload data set.                            |
|                                     | RECORD CODES                                                                                           | Cannot be changed.                                     |
|                                     | DATA ITEMS                                                                                             | Cannot be changed.                                     |
| Data Set File Specifications        | ALLOCATION - 1/2/3/4                                                                                   | Unload and reload data set.                            |
|                                     | FILE-SPEC - 1/2/3/4                                                                                    | Copy or rename data set.                               |
|                                     | SHADOW-FILE-SPEC - 1/2/3/4                                                                             | Copy or rename data set.                               |

## Modifying the SUPRA Server Directory data sets

You can increase the size or change the load limit of Directory data sets in UDD1, UDD2, and UDD3 using either:

- ◆ Fast utilities by changing the /TOTAL-RECORDS and /LOAD-LIMIT qualifiers (see “Using Fast utilities on UDD files” below).
- ◆ DBA utilities by using the unload and reload function (see “Using DBA utilities on UDD files” on page 206).

### Considerations

- ◆ You cannot use the Expand and Reset functions to modify Directory data sets UDD1, UDD2, and UDD3, because Expand and Reset update both the Directory and the data sets.
- ◆ Do not confuse the old SUPRAD.MOD and the new SUPRAD.MOD. If UDD1, UDD2, and UDD3 do not exactly match SUPRAD.MOD, do not sign on to DBA, explicitly or implicitly, except with the user name UTILITIES.

### Using Fast utilities on UDD files

You can use Fast utilities (CHANGEDB) to modify the UDD files.

Before you execute the Fast utilities against UDD files, create a recovery point for your Directory (see “Creating a recovery point” on page 199). To change the Directory data sets using Fast utilities, enter the command CHANGEDB at the command line with a list of qualifiers. You must specify the user name DATA-DICTIONARY to modify the Directory. For example, to increase the file size and reset the load limit for the related data set UDD2, enter the CHANGEDB command in the following format:

```
$ CHANGEDB /DATASET=UDD2 -  
-$ /RELATED /TOTAL_RECORDS=20000 -  
-$ /LOAD_LIMIT=90 -  
-$ SUPRAD -  
-$ /USERNAME=DATA-DICTIONARY
```

Alternatively, to alter more than one Directory data set at a time, use a change file containing a list of modifications. For example, the following change file modifies UDD1, UDD2, and UDD3:

```
! Change file UDDCHANGE.DAT
/DATASET=UDD1 /PRIMARY /TOTAL_RECORDS=10000
/DATASET=UDD2 /RELATED /TOTAL_RECORDS=20000 /LOAD_LIMIT=90
/DATASET=UDD3 /RELATED /TOTAL_RECORDS=10000
```

You can include only data set parameters in a change file. You cannot include the Directory Access parameters /USERNAME and /PASSWORD, or the database parameters /SIGNON\_DB\_NAME, /DB\_PASSWORD, and /OUTPUT. Specify these parameters on the command line.




---

For more information on using Fast utilities, refer to the *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220.

---

Invoke Fast utilities with a change file as follows:

```
$ CHANGEDB -
-$ /CHANGE_FILE=UDDCHANGE.DAT -
-$ SUPRAD -
-$ /USERNAME=DATA-DICTIONARY
```

If you wish to use more than one physical file for any UDD data set, remember that each UDD data set has a default file allocation of (1,0,0,0)—all the records are held in the first file specified and none are held in the other three. You must modify the file allocation value to reflect the number of physical files you define for each UDD data set. Fast utility usage, including details of file allocations, is described in more detail in the *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220.




---

Always create a recovery point for the Directory before you attempt any modifications. See “[Creating a recovery point](#)” on page 199.

---

## Using DBA utilities on UDD files

To unload/reload data sets using DBA utilities, perform the following steps:

1. Create a recovery point for the SUPRA Server Directory database SUPRAD (see [“Creating a recovery point”](#) on page 199).
2. Copy the existing SUPRAD.MOD file to another physical file. For example:  

```
$ COPY SUPRAD.MOD SUPRAD.SRC
```
3. Sign on to DBA with the user name DATA-DICTIONARY and perform the required changes. Change only one data set.
4. Validate and compile the database SUPRAD. This creates a new SUPRAD.MOD file.
5. Exit from DBA and sign on again with the user name UTILITIES.
6. Select the Utilities function from the main menu.
7. Select the unload/reload function from the utilities menu (function 1 for primary data sets or function 2 for related data sets). Specify SUPRAD.SRC as the source database file specification, and the new SUPRAD.MOD (created in Step 4) as the target. You must preserve the sequence of records on UDD2 and UDD3 when using unload/reload utilities. If you do not, then DBA can go haywire.
8. Repeat Steps 1 to 7 for each data set you want to change. You must invoke several utilities jobs, but you need to modify, validate, and compile the SUPRAD database only once for the primary data set UDD1, and once for the related data sets UDD2 and UDD3.

To unload and reload more than one Directory data set, first ensure that only one utilities job at a time runs against the Directory. When the utilities jobs finish, you can use the updated Directory, although taking a recovery point first is recommended. Refer to the [SUPRA Server PDM Utilities Reference Manual \(UNIX & VMS\)](#), P25-6220, for a detailed description of DBA utilities.

# 7

## Tuning your database

SUPRA Server is designed to provide optimum performance. When installed and implemented correctly, SUPRA Server can provide good response time and efficient work throughput. If your SUPRA Server performance deteriorates, that is response time increases, and batch jobs and application programs execute slowly, you can use these tuning guidelines to examine the way your system is running and look for ways to improve performance.



---

This chapter contains suggestions to help you tune your SUPRA Server system to improve overall performance. It is important to use the methods given in this chapter as guidelines only, not rules. You will find that some methods have more effect on performance than others; tuning is often a matter of trial and error.

---

Tuning can be broken down into the following general areas:

- ◆ System tuning
- ◆ Physical database tuning
- ◆ Logical database tuning
- ◆ Optimizing program design

This chapter discusses physical and logical database tuning, and efficient program design.

## Tuning your physical database

The first step in designing an efficient database is to normalize your organization's data. Data normalization involves reducing the transactions and data flow pathways to their simplest form. You can then use this framework as the basis for the physical database design. Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260, for details on how to normalize your data.

### Defining the file access method

You define the file access method for your database during database definition, selecting either QIO or RMS. A selection of QIO allows the PDM to choose the optimum access method on a file-by-file basis according to whether the file is local or remote. If the file is local, the PDM uses QIO access; if the file is remote, the PDM uses RMS access. QIO access to files can be multithreaded and provides optimum performance for local files. RMS access is single-threaded, which makes it slower than QIO. However, it is the only access method supported by DECnet. It is better to specify QIO for the file access method to allow the PDM to determine the best access method. It is also preferable to keep files on devices local to the PDM.

### Avoiding fragmented files

It is important to keep disk files contiguous. The SUPRA Server FORMAT utility attempts to create contiguous files; however, if this is not possible, FORMAT creates fragmented files and warns you if the file is in more than three fragments. You might also create fragmented files using the DBA utility unload/reload function. In this case, no warning is displayed.

Fragmented files cause performance to deteriorate. To eliminate fragmentation, you should perform a VMS BACKUP and RESTORE on each disk on a regular basis. Also, you can use a VMS COPY/CONTIGUOUS command.

## Using data sets

An important consideration for data set usage is the amount of wasted space in each block. For example, if records are 260 bytes long and blocks are 512 bytes long, each block holds one record plus 252 bytes of wasted space. You can avoid this wasted space by reducing the size of each record by four bytes, if possible, or by increasing the block size so a smaller proportion of the file is wasted. This is a major consideration when you use very large files.

## Evaluating redundant data items

Redundant data items are data items that appear in more than one data set. Redundant data items can improve the speed of data retrieval. However, they increase the processing time needed to update information and can cause database inconsistencies. Before you define redundant data items, consider the trade-off between speed of retrieval and speed of update.

## Using primary data sets

You should retain primary data sets for keyed access only. Non-keyed, repeating data can be moved from primary data sets to related data sets. This reduces the size of primary records, and allows more records-per-block and fewer out-of-block synonyms in the primary data sets. Transaction data contained in primary data sets can be converted to related data sets. For instance, if a primary data set contains a large amount of optional comment data, you can place this comment data in an associated related data set.

## Optimizing primary record retrieval

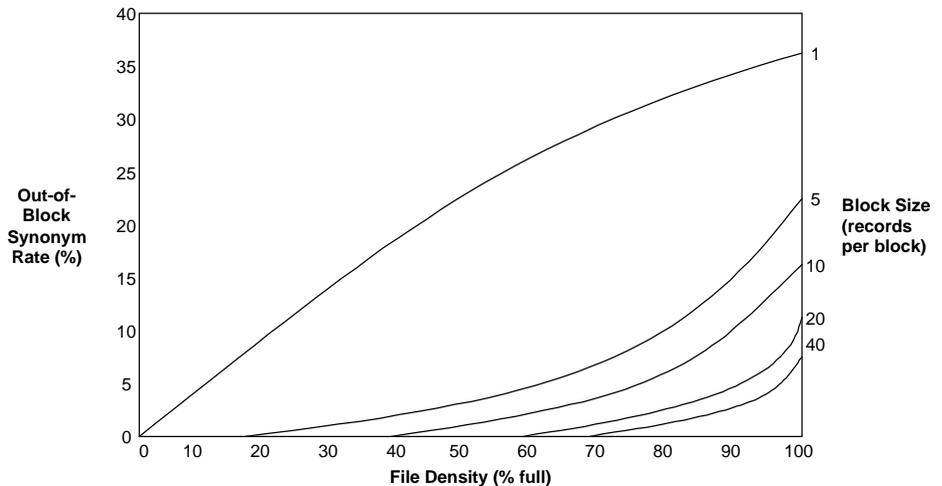
The PDM uses a hashed addressing technique, based on the value of the record key, to calculate a home address in the file for each primary record. The record key is randomized to give a relative record number (RRN) or home address. In some cases, two or more record keys randomize to the same RRN. These records are known as “synonyms.” When synonyms occur, SUPRA Server places the first record in the home address and chains all subsequent synonyms together (a “synonym chain”) as shown below:



A, B, and C are primary records whose record keys have all randomized to the same RRN. To retrieve record C (the last record in the chain), the PDM calculates the home address, retrieves A, finds that the record keys do not match, follows the chain to B, finds that the records keys still do not match, and then follows the chain to retrieve C.

The PDM attempts to place all synonyms in the same block so programs can retrieve any record in the synonym chain with only one physical I/O operation. The block in which a record's key value hashes to is called the home block for that record. If there is not enough room in the home block for a synonym record, the PDM places it in another block. Each block that the PDM has to search to retrieve a given record may require another physical I/O operation. The worst case, using the above example, would be if each synonym were in a different block. Three physical I/Os may be necessary to retrieve record C. If the file is fragmented so that the blocks are not contiguous, the above example could require even more physical I/Os to complete.

To optimize system performance, you should try to minimize the number of out-of-block synonyms. Since synonyms are bound to occur, you need to leave a certain percentage of file space free so most records can be placed in the home block. The following figure shows the out-of-block synonym rates measured at various packing densities and blocking factors.



The preceding figure describes the two factors that affect the out-of-block synonym rate:

- ◆ **File density**—Number of records in use as a percentage of file capacity, also known as the packing density
- ◆ **Block size**—Number of records per block

You can substantially reduce the out-of-block synonym rate by lowering file density and increasing block size. As a general rule, a density of 80% provides an acceptable level of out-of-block synonyms. The ideal block size depends on your type of application and the space you have available. However, it is possible to increase the block size to such an extent that it takes too long to read a list. Therefore, you must balance the benefits of reducing out-of-block synonyms against the data transfer time. A large data transfer time may give your task improved performance; however, it may degrade system efficiency.

## Using related data sets

The PDM stores related records in linked lists. Each related record can be associated with one or more primary records. The objective of blocking related data sets is to maintain a linked list of average length within one block. This means only one physical I/O is needed to perform successive actions on that single list (“chain”).

You specify the block size in the logical-records-per-block field when you define the file specifications for related data sets. Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260, for more information on specifying block size.

To work out the optimum block size for your application, establish the average length of a list of related records connected by a primary linkpath. You may need to return to the file specification screen several times to alter and test the effect of different block sizes. The values entered for control interval and load limit also affect whether a list of records overflows onto another block. The following sections discuss these considerations in more detail.

An important consideration for related data set usage is the amount of wasted space in each block. For example, if records are 260 bytes long and blocks are 512 bytes long, each block holds one record plus 252 bytes of wasted space. You can avoid this wasted space by reducing the size of each record by four bytes, if possible, or by increasing the block size so a smaller proportion of the file is wasted. This is a major consideration when you use very large files.

## Avoiding fragmented chains

The record chains of related data sets that are often updated may also become fragmented. Ideally, all the records for any given chain should be placed in the same control interval.

To avoid the deterioration in performance caused by fragmented chains, run the Fast utilities on volatile related data sets regularly. This may bring fragmented chains back into the same control interval for data sets that are below the LOAD level. Refer to the *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220, for a description of Fast utilities.

## Defining the control interval and load limit

You define values for the control interval and load limit at the data set file specification screen in DBA. Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260, for information on the DBA specification screens. The control interval specifies the number of related records whose allocation the PDM controls as a single unit. The load limit specifies the load capacity for the control interval expressed as a percentage. The control interval and load limit values affect the addition of records to a related data set in the following ways:

- ◆ If an application program adds a record to an existing list of related records, the PDM places the new record in a control interval that contains other records in the same list, if possible.
- ◆ If the new record is the first record of a new list of related records, or belongs to a list in a full control interval, the PDM puts it in a control interval that is below its load limit, if there is one.
- ◆ A control interval that is above its load limit accepts records only in lists that have already started in that control interval. However, when all control intervals are above their load limit, new lists may start in one of these control intervals, and subsequent sign-offs return the status LOAD.

The searching the PDM must do in the last case can cause severe performance degradation. Sign-on and sign-off PDML functions, and most subsequent PDML function requests, can remain pending while this search is ongoing.

Sometimes a new list starts in a block in which other lists have already started because the block is in a control interval that is below its load limit. This could cause parts of each list to overflow into different blocks or control intervals if many other additions are performed, thus causing chain fragmentation across control intervals. One technique that avoids this situation when loading a related data set is to force each new list to start in a new control interval by setting a low load limit. You can later use the DBA RESET function to increase the load limit after determining the average length of your lists. Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260, for a description of how to use the RESET function.

## Establishing the primary linkpath

A related record can be a member of as many relationships or linkpath chains as needed. However, since only one occurrence of the record exists for all the relationships, it cannot be placed in an optimum physical location for each linkpath of which it is a member. It is important to identify the primary linkpath for each related data set and ensure that as many programs as possible use it when adding records. You need the following information to determine the primary, or preferred, linkpath:

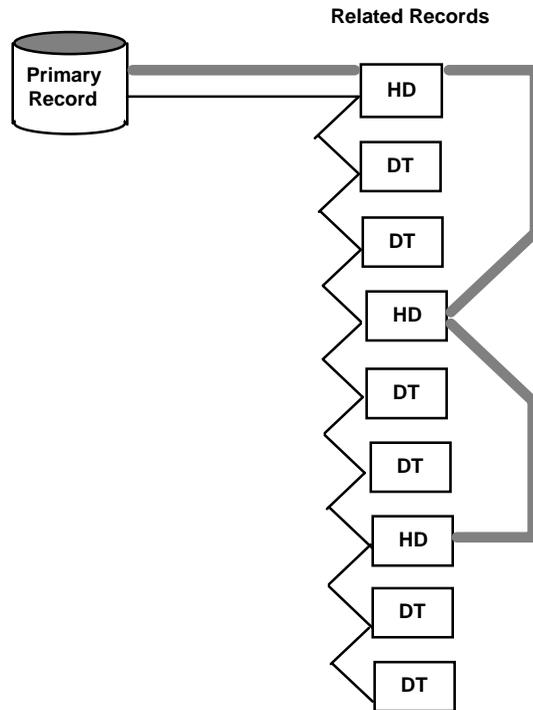
- ◆ Average length of each chain
- ◆ Frequency of access along each chain

As a rule, the primary linkpath should either identify the longest chains or the chains most frequently accessed. In practice, the primary linkpath is usually a compromise between the two.

To ensure that programs use the primary linkpath, either specify the primary linkpath to the PDM through the PDML linkpath parameter, or make sure all logical views that are allowed to add records to the file use the primary linkpath name in the ACCESS statements. (Note that in this context, “primary” refers to the preferred linkpath chain.)

## Using coded records

Processing long lists of related records is time consuming. You can alleviate this problem by using coded records with a header-and-detail structure. For example, in the following figure, the lines on either side of the related records represent different linkpaths. You can scan down the HD records until you are close to the required record. You can then change linkpaths and scan serially through the DT records.



Code-directed reads are more efficient than reading all the records on the linkpath until you find a record with the required code.

## Defining logical units of work

SUPRA Server is designed to process logical units of work. A logical unit of work is a group of database requests that must be completed together or not at all. For example, a logical unit of work involving transferring funds from a checking to a savings account might require subtracting the amount from an account balance record in a checking data set, and adding the amount to an account balance record in a savings data set. Either the entire logical unit of work or none of it should be done. Partially completed logical units of work result in logical data corruption. As in this example, if the application were to only deduct the amount from the checking account without also crediting the amount in the savings account, the account balance would be incorrect. This is an example of logical data corruption.

Any errors that occur within the processing of the logical unit of work require that all updates that occurred within the logical unit of work be reset. This can be accomplished by the application issuing a RESET function. However, if no problems occur while processing the logical unit of work, then the application can make that entire logical unit of work permanent by issuing a COMMIT function.

Any RESET or COMMIT function will reset or make permanent, respectively, all updates that have occurred since the most recent COMMIT, RESET, or SINON function.

The following example illustrates a sample program structure which could be used to implement a logical unit of work:

```
INITIALIZATION (Sign on)
While not finished, do:
    screen input and validation
    Database Update Processing
    If error, then reset
        else commit
end
TERMINATION (Sign off)
```

If your program updates database records, the records are held until the next RESET, COMIT, or SINOF.

If a task terminates before signing off, as with a CONTROL/Y, a VMS STOP command, or a DELETE/ENTRY on a batch job, the PDM will automatically issue a RESET and SINOF on behalf of the aborted task. This is called a dynamic sign off.

## Managing buffers



---

You can improve online responsiveness and dramatically reduce some batch run times by using PDM Cache, a high-performance buffer-handling option (see “[Improving database performance with PDM cache](#)” on page 219).

---

Buffer organization and buffer use at run time affect performance. There is a trade-off between memory and disk I/O performance. As your buffers become larger and more numerous, they occupy more memory, but your disk I/Os become fewer and larger. As you reduce the number and size of your buffers, you increase the memory available, at the cost of disk I/O performance. However, since commercial applications more often suffer from being I/O-bound than memory-bound, you should aim to improve disk I/O. Consider the following when deciding on your buffer strategy:

- ◆ You can allocate a pool of buffers defining the areas used for input from and output to primary and related data sets. The number of buffers in the buffer pool depends on how you use your database. For example, a multiuser database needs a larger pool of buffers than a single-user database.
- ◆ The PDM flushes the buffer contents to disk whenever a task performs a commit or signs off. The PDM also writes the buffer contents to disk when a task needs a block which is not in memory and no unused buffers exist. A reset also causes buffer flushes.
- ◆ The size of a data set record helps determine buffer size. The PDM calculates the size of a buffer automatically using the maximum block size of the data set. For example, if a data set contains five 300-byte records per block, its block size must be at least 1536 bytes.
- ◆ It can be useful to share buffers among data sets with the same block size. However, if data sets with different block sizes use the same buffer pool, the size of all buffers in the buffer pool will be set to the largest block size of any data set associated with the buffer pool.

- ◆ Related data sets are organized with all records on a chain as close together as possible. A large value for records-per-block allows many serial accesses without disk I/O. It can be helpful to specify a large number of records-per-block to cut down the number of I/Os.
- ◆ Another factor affecting the number of copies of buffers is simultaneous use of the database by multiple tasks. If the only copy of a buffer is in use, then another task must wait until the buffer is free, which may severely degrade performance. In a multitasking environment, you should monitor buffer handling to avoid thrashing. Thrashing takes place when excess I/O prevents useful work. You can change the number of buffer copies, alter the way buffers are shared between data sets, recompile the database, and monitor the performance of the database.

## Improving database performance with PDM cache

You can improve online responsiveness and dramatically reduce some batch run times by using PDM Cache. PDM Cache is a high-performance buffer-handling option which reduces CPU usage by the PDM server process. With PDM Cache, the PDM server process employs an alternate algorithm to search the buffer pool. This algorithm allows the use of larger buffer pools without dramatically draining the computer's processing reserve.



---

If PDM Cache is enabled without implementing the necessary tuning, performance benefits may be negligible. To effectively use PDM Cache, follow the guidelines below and use the DBA Toolkit offered through Cincom BCS Client Services. DBA Toolkit offers a dbmod generator, a PDM Monitor, a Fast Database Schema Generator utility, and a database statistics report generator.

---

Database performance tuning is a complex, ongoing activity dependent on numerous criteria and requiring regular monitoring and adjustments. There is no single setup which is optimal for all sites or even for the same site at different times. However, one of the most important ingredients in tuning any application's performance is its use of memory buffers. Tuning buffer use will allow you to maximize the benefits of PDM Cache. To effectively tune buffer use, it is helpful to review how buffers are searched.

## Understanding buffer search algorithms

When data records are accessed from a SUPRA Server database, they are maintained in memory buffer pools. These buffer pools are searched when an application task requests a record. The following search algorithms are used:

- ◆ **Serial Scan**—The list of buffers is linearly searched from beginning to end until the desired buffer is found. This is the only method used by the standard PDM. This method is used by PDM Cache V2 when there are fewer than 20 buffers (see Hashing).
- ◆ **Last Used, First Searched**—The list of pointers-to-buffers is constantly reorganized so that the last referenced buffer is placed at the beginning of the list. Thus, buffer handling is optimized for subsequent records located in the same buffer. This method is used by PDM Cache V1 unless there are sufficient private buffers to buffer the entire data set (see RAM Disk).
- ◆ **Hashing**—An internal algorithm determines (within 20 buffer searches) the actual buffer location within the list of buffers. The buffer can then be accessed directly. This method is used by PDM Cache V2 when there are at least 20 buffers.
- ◆ **RAM Disk**—A record's RRN is used to directly locate the actual buffer in memory. This method is activated in PDM Cache V1 when the DBA assigns a *private* buffer to a single file with as many copies of buffers as logical SUPRA blocks in the file. (A *private* buffer is a buffer exclusively assigned to a single database file.)

The following table shows algorithm use under different conditions (where  $n$  = the number of logical blocks in the file):

| PDM cache use | Number of buffers | Buffer search algorithms |                           |         |          |
|---------------|-------------------|--------------------------|---------------------------|---------|----------|
|               |                   | Serial scan              | Last used, first searched | Hashing | RAM disk |
| None          | 1 to 19           | X                        |                           |         |          |
|               | 20 to $n-1$       | X                        |                           |         |          |
|               | $n$               | X                        |                           |         |          |
| V1            | 1 to 19           |                          | X                         |         | X        |
|               | 20 to $n-1$       |                          | X                         |         |          |
| V2            | 1 to 19           | X                        |                           | X       |          |
|               | 20 to $n-1$       |                          |                           | X       |          |
|               | $n$               |                          |                           |         |          |

As can be seen in the preceding table, the number of buffers is a determining factor in algorithm use. Serial scanning is most efficient for a small number of copies of each buffer. As the number of buffers increases, performance can seriously degrade due to the CPU time required for each buffer search. In extreme cases, a search could require more time than a physical I/O, thereby negating the advantage of introducing a large number of buffers. This many-buffers situation is where the performance benefits of PDM Cache can be realized.

For example, let us say you have a standard PDM with a small number of buffers, and you wish to improve performance:

- ◆ If you add buffers without enabling PDM Cache, performance may improve.
- ◆ If you continue to add buffers without enabling PDM Cache, you will reach a point of diminishing returns. Eventually, performance will degrade.
- ◆ If you enable PDM Cache without adding buffers, performance will likely be unaffected.
- ◆ If you implement a combination of both PDM Cache *and* more buffers, a dramatic reduction in run times is likely.

## Tuning for PDM Cache use



---

As with any caching product, PDM Cache V2 requires the memory buffers to be populated with data before they can be effective. The PDM Cache fills a buffer when the data is first accessed from disk; buffers are not preloaded. Thus, there is no performance benefit for the first access to a SUPRA logical block. For example, if buffers are provided to buffer an entire data set in memory, and the data set is swept with DATBAS RDNXT functions, performance will be no better than if a small number of buffers had been allocated. After the buffers have been loaded with data, subsequent reads should be much faster.

---

### Minimizing dbmod load time

One of the impacts of using a large number of buffers is the additional time required by the PDM server process to load the corresponding dbmod file and create the memory buffers. If the dbmod is loaded once for many users in UPDATE mode, this is not an issue. But if dbmod is being unloaded and loaded between each job in a lengthy series of SINGLE mode jobs, this can be time-consuming. You can circumvent this problem by creating a special UPDATE dbmod with all the characteristics of a SINGLE mode dbmod (MAX-UPDATE-TASKS=1, no task or system logging, and no record-holding). This special dbmod can then allow batch jobs to sign on in UPDATE mode and still have exclusive use of the dbmod.



---

This special UPDATE dbmod cannot be generated with the standard csidba utility. It can only be generated using the dbmod generator feature of DBA Toolkit offered through Cincom BCS Client Services. Any support issues concerning this special dbmod must be handled by Client Services.

---

## Optimizing data set buffering

To take full advantage of PDM Cache, you must allocate the proper number of buffers to each data set. The proper number will vary by data set depending on the physical size of the data set and its activity. It will also vary by site depending on CPU speed, system load, and available memory. Small data sets and/or data sets with low I/O rates require fewer buffers. Larger data sets and/or data sets with high I/O rates require more buffers.

Before you can optimize your data set buffers, you must collect statistics about your database activity, particularly the logical I/O (LIO) and physical I/O (PIO) rates for each data set. A physical read is a read which requires disk access, while a logical read without a physical read is a buffer hit (an in-memory read which does not require disk access). LIOs are good for performance; PIOs are bad for performance. Providing more buffers increases the likelihood of an LIO over a PIO.

I/O statistics are written to the PDM log file when a file is closed (if the PDM input parameter STATISTICS=Y). I/O statistics can also be gathered online using the PDM Monitor feature of the DBA Toolkit. However, before making your buffer allocation decisions, you must also take into account the issue of PDM process memory (discussed next).

## Tuning PDM process memory

The more buffers you add in a dbmod, the more memory you need to allocate to the PDM server process. The amount of memory allocated depends on how much you have and how much you can afford to give to the process. However, once PDM Cache has been enabled, increasing buffers is “the name of the game.” Your goal is to balance performance and memory usage. Ideally, the PDM server process would have enough working set allocated to result in a zero page-fault rate ( always finding the desired pages in memory).

The memory that a particular dbmod is using can be determined after it is loaded by using the CSIOPCOM command DISPLAY/DATABASES and dividing the displayed size by 512.

If you increase buffers in a dbmod, the PDM server process will require more virtual memory to load the dbmod into memory. Insufficient virtual memory in the PDM server process will result in a CSTI140F operator message and an IDBM status returned to the application. PDM virtual memory is controlled by the PAGE\_FILE parameter in the CSISTRINP file. Check the PAGE\_FILE setting to see if it is adequate (using the method discussed later).

Avoid trading a data set I/O for a pagefile I/O. Even if the PDM server process has enough pagefile quota, you must also ensure that it has enough working set to run at optimum efficiency. If the PDM’s working set is too small, the operating system will spend too much time page-faulting the PDM server process. The PDM working set is controlled by the EXTENT parameter in the CSISTRINP file. You need to see if the EXTENT setting is adequate (using the method discussed next).

To see if the PAGE\_FILE and EXTENT settings are adequate, use the following DCL command *after the PDM has been through heavy use*:

```
$SHOW PROCESS/ACCOUNTING/IDENTIFICATION=pdm-pid
```

This will display the peak virtual size (limited by PAGE\_FILE) and the peak working set size (limited by EXTENT). If either of these peak values is close to corresponding setting in the CSISTRINP file, the CSISTRINP setting should probably be increased (or it will soon need to be). If the peak working set size reaches the full value of the EXTENT setting, the PDM server process will likely suffer from page-faulting due to insufficient physical memory.



---

The PAGE\_FILE setting is limited by the VMS system parameter VIRTUALPAGECNT and the size of the SYS\$SYSTEM:PAGEFILE.SYS file.

The EXTENT setting is limited by the VMS system parameter WSMAX. It is not uncommon for each of these parameter values to be in the 100,000–500,000 range. The appropriate settings for a particular site should be determined through regular monitoring and iterative adjustments.

---

## CONTROL: Manufacturing tuning considerations

In most shops, the MRP/MPS review processor (MPSMP070) is the longest-running batch job of the nightly job stream. Established CONTROL: customers have learned that a dbmod dedicated solely to MP070 is crucial for reducing MP070 run time. This is still true today, but the PDM Cache can make MP070 run significantly faster. To optimize MP070 performance, consider the following:

- ◆ Identify the subset of data sets used by MPSMP070. This may include (but not be limited to): BATD, BATM, BILL, INVD, LOCM, MSGS, OORD, ORDM, PART, PTDM, RQMT, SCAL, SCRML, SITD, SITM, SRCG, TABM, and TABV.
- ◆ Ensure that the LOGICAL-RECORDS-PER-BLOCK for each data set gives a SUPRA logical block size of 4096 bytes (possible exceptions are SCAL, PLV1, and/or LOCM which may need to be 8192, and should therefore, *not* share a buffer pool with a 4096 data set).
- ◆ Identify the MP070 hot files which could use more buffer copies. This may include (but not be limited to): PART, PTDM, RQMT, ORDM, and OORD.
- ◆ Define more buffer copies for the batch environment to take advantage of additional free memory which is typically available at night.
- ◆ Use the DBA Toolkit's Fast Database Schema Generator utility to create a non-logging, non-record holding, MAX-TASKS=1, heavily buffered, UPDATE mode database (multiple tasks with no record-holding can cause corruption).
- ◆ Set the batch tasks' sinon mode to UPDATE instead of SINGLE. This is usually done by controlling the definition of the logical name CTRL\_DB\_MODE.
- ◆ Ensure that the read-ahead feature is active by defining the logical name CSI\_READAHEAD with a value of YES.
- ◆ Turn off indexing at the start of the batch stream, and perform an index populate at the end of the batch stream.

---

## Optimizing Relational Data Manager performance

To improve performance when accessing SUPRA Server PDM data sets under VMS, you must understand what causes the Relational Data Manager (RDM) to scan records when fulfilling a view request.

The RDM is the means by which programs access the physical database. Make sure the views you create access the database efficiently; otherwise, performance can deteriorate.

### Accessing data sets

To obtain the best performance results when accessing data sets, define physical keys as logical keys in your RDM views. The next best option is to define secondary keys as logical keys. If you assign a logical key to a data item that is not a physical key or a secondary key, or that is only part of a physical key or secondary key, the RDM serially scans the data set to find records with the specified value. This type of processing degrades performance. However, the performance cost of defining logical keys that are not physical keys or secondary keys must be balanced against the advantages. It is less efficient for an application program to search for a record than it is for the RDM to do the search.

In a primary data set, the physical key is always unique, so any logical key assigned to the physical key is unique. A physical key in a related data set may or may not be unique. A secondary key may or may not have been specified as unique.

You can specify unique logical keys or nonunique logical keys. If you assign a unique logical key to a related data set physical key that is not unique, the RDM must search the linkpath chain from beginning to end to ensure that the key is not duplicated before it can insert a record. If you assign a unique logical key to a data item that is not a unique physical key or a unique secondary key, the RDM must search the entire data set before inserting or updating a record. This degrades performance. If you specify a nonunique logical key, the RDM does not need to check for duplicates before inserting or updating a record. Therefore, when your application requirements permit, use nonunique logical keys for fields which are not unique physical keys or unique secondary keys.

Any time you need a data item that is not the physical key of a primary data set to be a unique logical key, consider defining a unique secondary key on it. Also consider secondary keys for data items that will be used as nonunique logical keys.

## Choosing an RDM access method

The RDM offers two methods of specifying ACCESS statements:

- ◆ Specific access syntax which consists of the USING and/or VIA clause and a specified key, linkpath, or secondary key.
- ◆ Generalized access syntax consisting of the WHERE clause which instructs the RDM to determine the optimum access strategy in the following order of preference:

| Related data set | Primary data set | Related data set |
|------------------|------------------|------------------|
| 1st choice       | Control Key      | Linkpath         |
| 2nd choice       | Secondary Key    | Secondary key    |
| 3rd choice       | Sequential       | Sequential       |

In most cases, it is efficient to use the generalized access syntax and let the RDM choose the best access method. However, in some cases you might want to force the RDM to access the data via a secondary key, for example, producing a customer report ordered by customer number where customer number is the control key to the customer record. If you use the generalized access syntax, the RDM will access the customer records using the control key customer number. However, the records may not be retrieved in order. If you define an index (secondary key) on customer number, you can use the specific access syntax to force the RDM to retrieve the records in order by accessing them via the secondary key.

Once you have defined your views, use the DBAID command SHOW-NAVIGATION to check that the RDM is using the access method you want. Also, use the STATS commands to check that the RDM is not scanning the records—that there are few PDM reads for each RDML GET.

If you use indexes, you do not need to specify the ORDER clause in your RDM views. This reduces processing overhead.

## Using bound views

View binding allows you to have a preopened copy of a view stored on the Directory. View binding improves performance in the initial access to a view by reducing the processing time required for a request to open a view.

## Using Global Views

The Global View Facility stores a copy of preopened views in a file which programs use as a shared VMS global memory section. This saves the RDM the processing overhead of opening views when application programs first access them. The global view also contains SUPRA Server user information and passwords so the RDM can run without any access to the SUPRA Server Directory (by setting the logical CSI\_NODIRECTORY to TRUE).

Using global views not only reduces the processing time needed to open views, but it also reduces the memory needed by RDM tasks. This is because each RDM task can access a single copy of the preopened views stored in global memory.

You can decide which views to make global by determining the frequency of use of the views. You then define global views and generate the global view file using the Global View Creation function.

If you include a view in the global view file, the global view definition overrides the view definition on the Directory. To include a modified view (or remove a view) in the global view file, you must create a new global view file. Subsequent RDM tasks can use the new global view file. Refer to the *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220, for details of how to create and use global view files.

## Using indexes

You define indexes during database definition by connecting them to a data set. Each data set can have one or more indexes. Each index can contain one or more secondary keys and each secondary key can be connected to one or more data items. Consider the following when defining an index:

- ◆ For best performance, you should define no more than four secondary keys per index, and from one to four indexes per data set.
- ◆ It is more efficient to group all the secondary keys for one data set in one index, rather than create several indexes each containing one secondary key. However, if you want to be able to deactivate selected secondary keys, define each secondary key in its own index. When you deactivate an index, you implicitly deactivate all secondary keys it contains.
- ◆ If performance is a problem, you can make a small improvement by setting INDEX-READ-VERIFY = NO on the index attributes screen. This turns off the automatic check for index corruption and is fairly safe if used on an index you do not intend to deactivate. Frequent deactivation increases the risk of index records becoming out of date, no longer matching updated data records. If the index subsequently becomes corrupted and you have specified INDEX-READ-VERIFY=NO, results will be unpredictable.



---

You can obtain tremendous performance benefits from using indexes during read-only operations; however, indexing becomes an overhead during maintenance operations such as insert, update, or delete.

---

Using secondary keys to access data offers the following advantages over using primary keys and linkpaths:

- ◆ Fast online lookup of non-volatile data. For example, customer records can be keyed on customer number. However, you could define an index on customer name and initials for more flexible retrieval of customer records. The flexibility results from the ability to key into the customer record using name instead of number. Also, because secondary keys support generic reads (a search based on only part of the key), you can use wild card characters to retrieve records matching the partial key that is supplied. The use of secondary keys on records such as customer records, which seldom change once entered, incurs minimal performance overhead.
- ◆ Periodic report generation based on secondary keys rather than primary keys. For example, you may want to produce monthly reports ordered by product number within product class. The primary data set has product number as its primary key, and product class as a data field. To generate your reports, you define an index with secondary keys for product class and product number. You bring the index up-to-date and activate it once a month using the CHECK indexing utility. After you produce the report, you deactivate the index for the rest of the month to avoid the performance overhead of maintaining the index.
- ◆ Reduced overhead in maintaining a sequenced primary linkpath. By defining a secondary key on a sequencing field and the foreign key, the PDM no longer needs to maintain the chain of related records in sequence. Instead, it can use the index to retrieve records in order by inserting records at the bottom of the chain instead of sequentially searching the chain for the logical position. This may improve INSERT performance by avoiding the search and reducing physical I/O, and eliminates the need for the RDM ORDER clause. For example, to retrieve all identical surnames in first name order, place a secondary key on the sequencing field (first name).
- ◆ Sequenced secondary linkpath simulation. The WHERE clause (generalized access syntax) instructs the RDM to select the optimum access method. When accessing related data sets using the WHERE clause, the RDM will always use a linkpath in preference to a secondary key. If you define the data item used in the linkpath as a secondary key, the RDM will maintain an index on that data item, which, in effect, sequences the secondary linkpath. Referential integrity is ensured if you are using the RDM. If you are not using the RDM, you must make sure the application program includes logic to enforce referential integrity.

## Designing application programs

Application program design can have a far-reaching effect on the efficiency of your system. The following guidelines apply to all programs that access the PDM.

### Record holding

Record holding is a feature of the PDM that protects records from being updated by two or more tasks at the same time. The maximum number of records that can be held for a database is determined by the product of `MAX-HELD-RECORDS` and `MAX-UPDATE-TASKS`.

When the PDM input parameter `MULTIHOLD=N`, a task can explicitly read and hold only one record per file per logical unit of work. With each subsequent request to explicitly read and hold a record, all previous explicitly held records for the file are released for the task. With `MULTIHOLD=N`, only the latest record explicitly read will be held for the task.

When the PDM input parameter `MULTIHOLD=Y`, a task can explicitly read and hold as many records as the record holding table permits.

All records held by a task will be released at the next commit, reset, or sign-off.

### Holding records with RDML

- ◆ GET commands with the UPDATE option cause the records retrieved to be held.
- ◆ INSERT, DELETE, and UPDATE commands cause all records needed to process the function to be held. This includes records needed to maintain the linkpaths or synonym chains.

### Holding records with PDML

- ◆ Use of the end parameter “END.” on read functions causes the retrieved record to be held. Read functions are RDNXT, READD, READM, READR, READV, and READX.
- ◆ Functions which add, delete, or update records cause all records needed to process the function to be held. This includes records needed to maintain the linkpaths or synonym chains. These functions are ADD-M, ADDVA, ADDVB, ADDVC, ADDVR, DEL-M, DELVD, WRITM, and WRITV.
- ◆ The CNTRL function is also used to hold records.



---

Record holding is disabled when a task signs on to a database in SINGLE mode (refer to the *SUPRA Server PDM Programming Guide (UNIX & VMS)*, P25-0240), formats files, runs the DBA utilities, or when MAX-HELD-RECORDS is set to zero.

---

## Managing record holding

RDM application programs may use the RDML commands GET, GET FOR UPDATE, and UPDATE. GET retrieves the record, but does not hold it. GET FOR UPDATE retrieves and holds the record. UPDATE holds and updates the record. The duration of the record holding and the success of the update depend on how you combine these commands along with the COMMIT and RESET commands in a program.

The following examples illustrate three methods of retrieving and updating records. Use the method in Example 1 to minimize record holding. Use the method in Example 2 if it is important that the update succeed. Use the method in Example 3 to ensure that the record is not modified or deleted until the end of your logical unit of work.

| Example | Code                                    | Explanation                                                                                                                                                                                                                                                                                                        |
|---------|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | GET<br>.<br>.<br>.<br>UPDATE            | Retrieves the record, but does not hold it until later in the program when the program performs the UPDATE. This minimizes record holding.                                                                                                                                                                         |
| 2       | GET FOR UPDATE<br>.<br>.<br>.<br>UPDATE | Retrieves and holds the record. By the time the program updates the record, it will have held it for some time. This method ensures the success of the update by preventing any other program from updating the record. However, it can impair performance if used on records which are accessed by many programs. |
| 3       | GET FOR UPDATE<br>.<br>.<br>.<br>.      | Holds the record without updating it. This method ensures that other programs cannot update the record until the end of your logical unit of work, but will impair performance if used on records which are accessed by many programs.                                                                             |

Consider the following regarding record holding:

- ◆ If a process requires exclusive use of your database, you can select the single-task option with no logging. The single-task option bypasses all record-holding logic and disables task level recovery, which thereby improves performance. Always create a recovery point before running a task in single mode. Failure of a single mode task requires a restore of the last recovery point.
- ◆ As the number of held records increases, the time taken to search the holding table increases.
- ◆ Note that the holding table is a pool of entries available for all users. Thus, if you define MAX-UPDATE-TASKS to be 10, and MAX-HELD-RECORDS to be 100, then one user holding 1000 records could use the entire record holding table.
- ◆ You can control the number of times the PDM will try to get and hold a record being held by another task (retries) and the time between retries by setting the PDM input parameters RETRY and INTERVAL (see “[Entering parameters for the PDM input file](#)” on page 125). The PDM performs the number of retries specified in RETRY, and waits for the length of time specified in INTERVAL between retries before returning a HELD status to an application task. If you encounter an excessive number of HELDs, you can try to increase the number of retries and/or the interval between retries. More retries use CPU cycles, which can cause performance deterioration. A longer interval between retries reduces CPU cycles; however, the record may be stolen by another task before the interval is up. For overall system performance, it is better to increase the interval and reduce the number of retries.
- ◆ It is much more efficient for the PDM to do the retries for a record held by another task than it is for an application program.

- ◆ These three methods minimize record holding:
  - **Avoiding excessive record holding:** When more than one task is signed on to a database, records that are held by another task are temporarily unavailable. Since some records are accessed frequently by many programs, ensure that each record hold is as infrequent and as brief as possible. Do not hold a record until it is necessary to do so, and commit as soon as possible after updating, or reset as soon as the need has been determined. To this end, make sure that no program holds records while it waits for a response from the user. This allows other tasks to access those records that would otherwise be held.
  - **Reducing the duration of record holding:** A program that needs to update several records may place a hold on each record, collect the update information, update each record, and then release the record holds. If the performance of your system is degraded by numerous processes trying to access the same records concurrently, you can reorganize your application programs so they gather all the information necessary to update the records first, and obtain the record holds only when all the update information has been collected. This reduces the amount of time each record is held by a process and helps minimize record contention.
  - **Altering the order of record holding:** Sometimes many tasks compete to update a subset of data. You can reduce record contention in such cases by modifying the order in which your application programs issue requests for record holds. Ensure that your programs first request holds on the records that are more likely to be held by other tasks. Then, if another task already holds one of these records, the reset requires less processing time.

## Preventing a deadly embrace

In some cases, two tasks may request the same database records simultaneously. For example, Task A holds Record x and starts processing. Meanwhile, Task B holds Record y and starts processing. At some point, Task A tries to hold Record y. Since y is already held by B, Task A receives a HELD status. During Task B's processing, it tries to hold Record x, which is held by A. Since x is already held by A, Task B receives a HELD status. Therefore, if neither A nor B releases their held records, then neither task will ever complete since they are waiting for each other. This situation is called a deadlock, deadly embrace, or fatal embrace.

Any task receiving a HELD status can retry the function a finite number of times. If the HELD status persists, the application should issue a RESET and then retry its processing from the most recent commit point. The RESET command resolves a deadlock by releasing all held resources for the logical unit of work. Contention between logical units of work can cause poor performance. Consider having the PDM do the retries instead of your application program (see [“Managing record holding”](#) on page 234).

## Optimizing the frequency of commits

Consider the frequency of commits. Excessive commits and infrequent commits can both be inefficient. Excessive commits increase physical I/O and affect performance. Long commit intervals lead to large record-holding tables which take excessive time to search. Held records are unavailable to other users until the next commit, reset, or sign-off function is performed. See [“Defining logical units of work”](#) on page 216 for further details on logical units of work.

## Understanding client read-ahead buffering

Client read-ahead buffering is a mechanism used by the database access program (DATBAS) and PDM to improve performance of both sequential and serial reads when no record holding is being done. When your application executes a read function, RDNXT, READV, READR, or READX with the end parameter set to RLSE, the record is read and returned to your application without holding the record. It is also possible to turn off record holding by using the SINGLE SIGN-ON mode as explained in the *SUPRA Server PDM Programming Guide (UNIX & VMS)*, P25-0240.

When DATBAS, either single-task or multitask, gets a request for a read function with no record holding, it requests the record from the PDM as usual. If the application makes another request without changing the file, the element list, the qualifier, the reference, or the end parameters, DATBAS requests that PDM place as many records as possible into the interprocess communication area without causing any additional I/O. In the case of READX, DATBAS requests that 10 records be returned if they will fit into the interprocess communication area.

Upon receiving the interprocess communication area from PDM, DATBAS copies it to one of three read-ahead buffers created automatically in the applications local memory. There is a read-ahead buffer for RDNXT, one for READV and READR, and one for READX.

When the application makes another request for the same file, the record is extracted from the read-ahead buffer. No access to the PDM is required. This continues until all the records in the buffer are exhausted, or one of the parameters changes, or a COMMIT function is executed.

Because there are three read-ahead buffers, it is possible to intermix the functions without losing the buffers. Your application could perform a mix of RDNXT, READV or READR, and READX as well as any other read or update functions and retain all the previously read records. Many programs perform sequences of reads to find data they are looking for.

As an example your program might do a RDNXT through the primary data set. For each primary record found, the program might perform a series of READV functions to read a set of related records. Both the RDNXT and READV functions in this example would be able to use the read-ahead buffers. If the program does a READV from two different data sets, alternating back and forth, no read-ahead buffering is possible for the READV functions. The performance of the application could be improved by accessing all of the records in one set and then accessing all the records in the other set. This process best uses read-ahead buffering.

## Context position considerations

The position within a data set is retained by your application in the qualifier parameter. Your position is never lost regardless of the read-ahead buffering.

## Application programming considerations

The application programmer does not have to specifically code for read-ahead buffers. The behavior of the PDM is identical with or without read-ahead buffering with the following exceptions:

- ◆ Because more records are being returned to the application program's process, there is a higher likelihood that the data read will be updated by another task while the reading application has the records in the read-ahead buffers. This could cause some applications to behave slightly differently with read-ahead buffering active. It may be more likely to get IVRP errors when reading with READV and READR when read-ahead is active.
- ◆ When read-ahead buffering is active, the behavior of \*FILL= can be slightly modified. The values passed in the data area are not necessarily the ones used to fill the data area of the returned records. When DATBAS requests multiple records from PDM, the data area containing the \*FILL= values are passed to PDM and are used to fill all of the records read. This works for most applications. However, this will not work for applications that modify the \*FILL= data on each DATBAS call.

## Turning off read-ahead buffering

If your application fits either of the above patterns, you will have to turn off the read-ahead feature. This can be done by defining the logical name CSI\_READAHEAD with the following VMS command:

```
-$ DEFINE CSI_READAHEAD NO
```

Note this is a process logical name and read-ahead can be turned off without affecting other applications running on the same PDM server. To turn read-ahead off for all applications define it in a shared logical name table such as group or system.

## Turning on read-ahead buffering

To turn on the read-ahead feature, you may either remove the CSI\_READAHEAD logical name with the deassign command or change its value to YES with the define command

## Printing read-ahead buffer statistics

Additionally, the logical name CSI\_READAHEAD\_STATISTICS may be set to the value of YES in order to produce read-ahead statistics. The statistics are printed to the console to which the CSI\_CONSOLE logical name points. If the CSI\_READAHEAD\_STATISTICS logical name is not defined, no statistics will be printed.

The following message will be printed to the console:

```
SINOF statistics (program name)(database)Total Calls Buffered Calls
      RDNXT                (nnnn)      (nnnn)
      READV/READR          (nnnn)      (nnnn)
      READX                (nnnn)      (nnnn)
```

The Total Calls column shows the total number of calls made by the application for each function. The Buffered Calls column shows the number of calls that were executed using the read-ahead buffers. These functions were performed without accessing the PDM server.

The CSI\_READAHEAD and CSI\_READAHEAD\_STATISTICS logical names can be defined in any logical name table. They do not have to be defined in the same table. By defining them in a process table, the values assigned will apply only to the process. By defining them in a group table, they will apply to all members of the group. By defining them to a system table, they will apply to all users of the PDM system.

## **Remote application considerations**

Read-ahead buffering is supported for remote applications linked with DATBAS and using either DECnet (from a VMS node) or TCP/IP (from either a UNIX node or a PC Windows node). The logical names CSI\_READAHEAD and CSI\_READAHEAD\_STATISTICS must be defined on the remote machine.

## **Performance tuning using the MAXDATA PDM input parameter**

Read-ahead buffering uses the data area defined by the MAXDATA PDM input parameter to transfer records from PDM to DATBAS. The size of this area and the size of the data requested in the element list parameter limit the number of records that can be transferred in one request to the PDM server. Performance is usually improved by reducing the number of PDM server requests. By increasing the MAXDATA PDM input parameter, more records can be transferred from PDM to DATBAS in one PDM server request, reducing the number of PDM server requests required to read a data set or set of records. This is especially important if the amount of data being requested in your program is very large. The minimum and default value for MAXDATA is 4096, the maximum is 32767.

## PDM application guidelines

The following suggestions apply only to PDML applications:

- ◆ **Controlling data item lists.** It is preferable to minimize the length of the data item lists in your application programs. Do not request data items that the program does not process. Also, keep the sequence of the data item lists the same as the physical sequence of the data items in the record wherever possible. This speeds up processing and improves performance.
- ◆ **Binding data items.** Bind frequently accessed data items to increase efficiency. The program searches the data item list only one time - the first time it is used. Data item binding is worthwhile whenever the same data item list is used more than once, regardless of the number of data items. Refer to the *SUPRA Server PDM Programming Guide (UNIX & VMS)*, P25-0240, for further information about binding data items.
- ◆ **Selecting access routes.** When you issue an ADDVA, ADDVB, ADDVC, or ADDVR, make sure you choose the most efficient linkpath.

# 8

## Migrating a database

This chapter provides details on how to move a database description and all data for the database. The move may be to or from a Cincom database management system (DBMS) on an operating environment other than VMS, or to or from a Cincom DBMS on VMS. In all cases, the following are required:

- ◆ Data Definition Language (DDL) file describing your database
- ◆ Sequential files containing your data



---

Make a back-up (create a recovery point) of your SUPRA Server Directory database (SUPRAD) before you try to load DDL and/or any data into a database (see “[Creating a recovery point](#)” on page 199). Be sure to back up SUPRAD.MOD, the UDD files, and any logs for SUPRAD. Make sure no tasks are accessing the Directory when you create the back-up copy.

---

Use these basic migration steps on your source system:

1. Generate DDL file from the source database
2. Generate a sequential file containing the data from each data set

Use these basic migration steps on your target system:

1. Back up the source SUPRA Server Dictionary and all associated data sets
2. Load the DDL
3. Validate and compile the database description
4. Format data sets and log files
5. Load data from the sequential files into the data sets



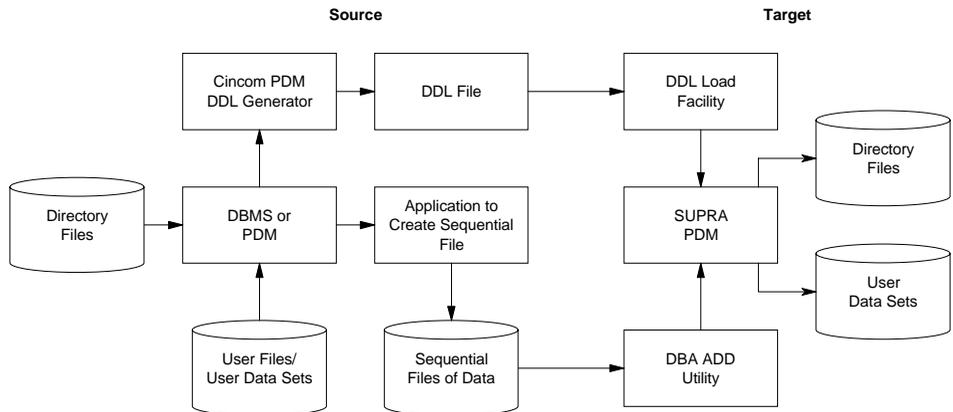
---

SUPRA Server for VMS does not currently support DDL for logical data items or logical views.

---

## Migrating into SUPRA Server

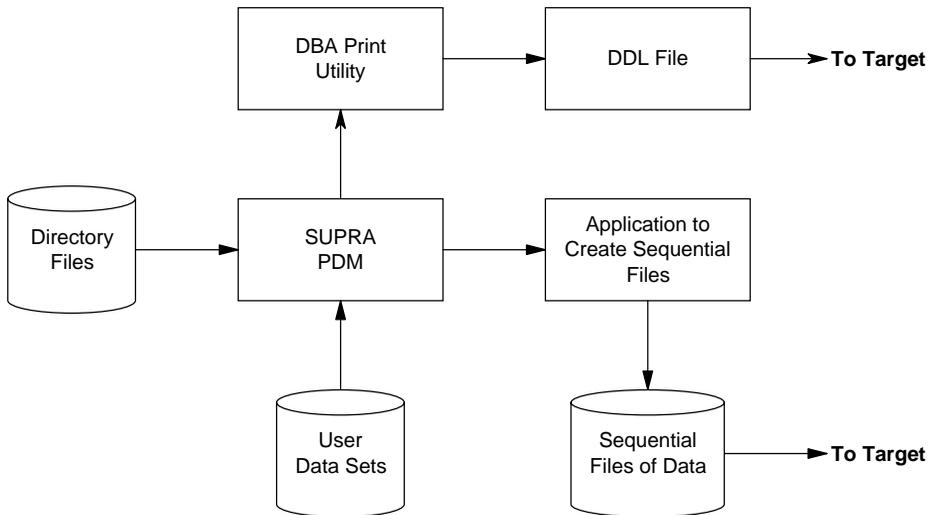
When you migrate your database into SUPRA Server, use the documentation from the source system to generate the DDL input file. Generate sequential files of data by generating one from each of your data sets. Move to the new operating environment and do any required character set conversions before you attempt the migration.



## Migrating from SUPRA Server

When migrating from OpenVMS SUPRA Server, you must generate the DDL file and the sequential files of data. See “Generating a DDL file” below for details on generating the DDL file that will be used on the target system.

You must copy the data from each data set in your SUPRA Server database and put it into sequential files. These files will be used on the target system to add records to the new database. Character set conversion may be done either as you create the sequential files, or after these files have been moved to the target system. Operating environment restrictions may apply.



---

## Generating a DDL file

Before you can migrate your database to another operating environment or release of SUPRA Server, you must first generate a copy of your database definition in DDL format. To do this, you must set the logical definition COB\$SWITCHES, then select the DBA Print function. The logical name causes DBA Print to generate a copy of the database description in DDL format.

The steps to generate a DDL file are:

1. Back up (create a recovery point) the SUPRA Server Directory database (SUPRAD) and all associated files (see “[Creating a recovery point](#)” on page 199).
2. Set the COBOL switch COB\$SWITCHES as follows:  

```
$ DEFINE /USER_MODE COB$SWITCHES 1
```
3. Sign on to DBA and select Print (Function 7) from the Database Description Function Menu to create a DDL file of the database you specify.



---

A user mode logical is only used by the next image you execute in your process. If you submit the print to run in batch, remember to make the logical name available to the batch process. If you do this by creating the definition in the group logical name table, any other user in the same UIC group printing a database description will generate a DDL file instead of the normal print listing file.

---

## Using the DDL Load Facility

The DDL Load Facility, which is identified by the logical CSIDDLLOAD, loads a database description onto the SUPRA Server Directory using an existing Data Definition Language (DDL) file. Use the DDL Load Facility to load a database description from one Directory to another.



---

You can use Batch Directory Maintenance to copy a database. For more information, refer to the *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220.

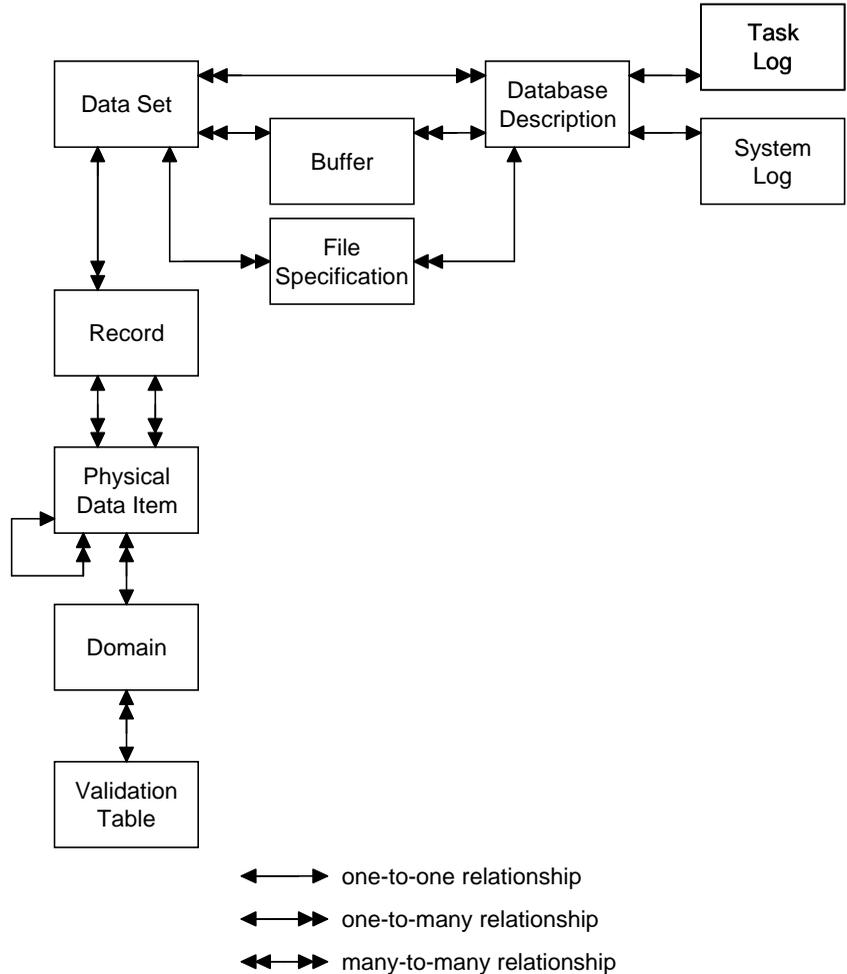
---

The SUPRA Server Directory allows no duplication of database and data set names. If you are starting with a clean Directory, you will have no problems. However, if you already have databases and data sets defined, you must ensure that the name of the database and the names of the data sets you are about to add do not already exist on the Directory.

Be especially careful if the DDL Load Facility has already been run unsuccessfully in update mode, as the Directory may have been partially updated. Delete the database descriptions through DBA before trying again, or restore the back-up copy of the Directory and use the restored version. If you attempt to load a database under a name that already exists on the Directory, SUPRA Server responds with a message and returns to the command level without completing the load.

The DDL Load Facility generates a listing output file named DDLLOAD.LOG containing the DDL transactions and any errors. This file is created in your current default directory.

The following figure illustrates a data model of the physical directory structure. The DDL Load Facility adds the entities, their attributes, and the relationships according to this framework. Each box represents an entity and each line represents a relationship.





## Loading the DDL file

After you have successfully signed on, you are prompted to give the full file specification for the DDL input file. Select CHECK ONLY or UPDATE mode, and specify the name of the database.

```
CINCOM SYSTEMS          DDL LOAD FUNCTION

File containing DDL or <return> to exit
: testdb.ddl

CHECK ONLY or UPDATE data directory (C or U) :
(<PF4> will select U) : U

Do you want to take options for each data set in turn? (Y/N)
(<PF4> will select NO) : Y

Change Name of database ? (<PF4> will select TESTDB)
or <PF1> to exit : <PF4>
```

---

### FILE CONTAINING DDL

**Description** Identifies the complete file specification of the DDL input file.

**Format** Up to 64 alphanumeric characters

**Consideration** If you enter the name of a file that does not exist or cannot be opened, an error message is displayed and you are allowed three more attempts. If these further attempts fail, CSDDLLOAD terminates.

## CHECK ONLY OR UPDATE

**Description** Indicates whether you want to check or update the Directory.

- Options**
- C Check-only mode does not affect the Directory.
  - U Update mode updates the SUPRA Server Directory with your data.

**Considerations**

- ◆ Check-only mode performs a syntax check and does not update the Directory.
- ◆ CSDDLLOAD is not a validation routine and assumes that the DDL is correct.

---

## OPTIONS FOR EACH DATA SET

**Description** Indicates whether you want to use the DDL Load Options screen.

- Options**
- Y This gives the DDL Load Options screen for each data set specified in the DDL file.
  - N Each data set is automatically added and not renamed.

**Consideration** You should answer Y if you already have data sets in your SUPRA Server Directory which have the same name as the ones in your DDL file.



## ENTER CHOICE NO.

**Description** Specifies the action for each data set.

- Options**
- 1 Loads the data set.
  - 2 Loads the data set under a different name (useful if a data set of that name already exists on the Directory).
  - 3 Omits the data set from the load.

### Considerations

- ◆ If you select option 2 and rename SET1 to SET2, as in the following example, the SUPRA Server Directory data set will be called SET2, and all its records and data item names will be prefixed by SET2. The file specification names will not be changed, so remember to change them after the load is finished, if required, using DBA functions (refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260).
- ◆ If you select option 2 and rename a primary data set, check for linkpaths to related data sets that reference the original name. Change the linkpath references in the related data set using DBA functions.
- ◆ If you select option 3 to ignore this data set, it has the same effect as if all DDL for the data set had been removed from the input file. You may have specified a buffer intended for this data set in the DLL. This buffer will be created on the SUPRA Server Directory, but it will not belong to any data set.

**Example**

The following example shows the screens that are displayed if you select option 2 for Enter Choice No:

```
CINCOM SYSTEMS          DDLLOAD LOAD FUNCTION
New name for data set SET1 : SET2
```

There is a time interval between each menu while the data set, records, data items, and file specifications are loaded onto the SUPRA Server Directory. The system then displays a confirmation screen:

```
CINCOM SYSTEMS          DDLLOAD RUN SUMMARY

ddlload.log created for your information

Number of DDL errors :    0

<< Press any key to terminate DDLLOAD >>
```

The entities and relationships are added to the Directory as they are read. Comments that belong to a particular entity are also added to the Directory, regardless of whether they occur in one contiguous block in the DDL file.

## Checking CSIDDLLOAD error conditions

The DDL Load Facility uses the DDL input file and generates a listing output file. It is not a validation routine, and will stop if an error is encountered on either of these files.

### Status codes

If a bad status occurs on either the input file or the listing output file, the DDL Load Facility displays a message indicating the name of the file and the error status code from the operating system. If CSIDDLLOAD is attempting to open your DDL file, the error message is displayed after three unsuccessful attempts.

### SUPRA Server error messages

SUPRA Server Directory errors, internal coding errors, and some I/O errors result in an error condition for which the system displays a SUPRA Server error message. For the corrective action for any message you may receive, refer to the *SUPRA Server PDM Messages and Codes Reference Manual (PDM/RDM Support for UNIX & VMS)*, P25-0022.

### Listing output file messages

Error messages in the listing output file indicate an error in the DDL. This does not normally occur. However, warnings may be issued for DDL statements that are not valid for OpenVMS SUPRA Server.

---

## Compiling the database description

After you have successfully loaded your database description into the SUPRA Server Directory using the DDL Load Facility, use the DBA Facilities (or COMBAT, the batch database compile program) to validate and compile your database description. (Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260.)

---

## Formatting data sets

Before you can add records to the newly compiled database, you must format all files (data sets and recovery log files). Use the DBA Administration Facilities (or CSTUFMT) to format the files. Refer to the *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260, for details.

---

## Adding records

Before you can use your new database, you must load your data. The data from your source system database was copied into sequential files that have been transferred to the target system. To add this data to your newly formatted data sets, you can use the Utilities option of the DBA Administration Facilities to add records to each data set. Refer to the *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220, for details. Alternatively, you can write an application to add the data to the database.

# A

## Example user exits

This appendix presents example user exits written in COBOL and FORTRAN. The COBOL example uses both the before and after exits for VAX and ALPHA; the FORTRAN example uses only the after exit. Command files for compiling and linking both examples are provided at the end of each section.



For all Alpha versions and beginning with version 2.4 on VAX, you must include a new additional parameter, Nargs or p0. See “COBOL user exits” below and “FORTRAN user exit” on page 267. This new parameter is a placeholder containing the number of arguments passed. It is unused but must be present.

### COBOL user exits

The COBOL user exits trace PDML parameters to check whether they are being passed correctly between the application program and the PDM. The before exit displays the parameters before the call to the PDM. The after exit displays the parameters after the call to the PDM. In addition, the after exit shows the status of each PDML call. The traced PDML functions are ADDVA, ADDVB, ADDVC, DELVD, READR, and READV.

**COBOL user exit 1**

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CSD_UPDM_USEREX1.
*
*   This is an example of a BEFORE exit.
*   (a BEFORE exit is called before the parameters are passed to
*   SUPRA PDM for execution)
*   This example processes RELATED DML functions only.
*
ENVIRONMENT DIVISION.
*
DATA DIVISION.
*
WORKING-STORAGE SECTION.
*
01  I                               PIC S9(4)  COMP.
*
01  KEY-HEX.
    03  KEY-HEX-CHAR                PIC X(8)   OCCURS 2.
*
01  KEY-CHARACTER.
    03  KEY-CHAR                    PIC X     OCCURS 8.
*
01  REFER-CHARACTER.
    03  REFER-CHAR                  PIC X     OCCURS 4.
*
01  REFER-HEX                       PIC X(8).
*
LINKAGE SECTION.
*
01  NARGS                            PIC S9(9)  COMP.
01  P1-FUNCTION                       PIC X(5).
01  P2-STATUS                         PIC X(4).
01  P3-DATASET.
    03  P3-UDD                       PIC XXX.
    03  P3-FILLER                     PIC X.
01  P4-REFER.
    03  P4-REFER-COMP                PIC 9(9)  COMP.
01  P5-LINKPATH                       PIC X(8).
01  P6-CTRL-KEY.
    03  P6-CTRL-KEY-COMP             PIC 9(9)  COMP OCCURS 2.
*   For this example the Control Key (above) is 8 bytes
01  P7-ELEM-LIST                      PIC X(28).
*   For this example the Element list is 3 element names + END.
01  P8-DATA-AREA                      PIC X(24).
*   For this example the Data Area is 24 bytes.
01  P9-ENDP                          PIC X(4).

```

```
*
PROCEDURE DIVISION USING NARGS P1-FUNCTION P2-STATUS
                    P3-DATASET P4-REFER
                    P5-LINKPATH P6-CTRL-KEY
                    P7-ELEM-LIST P8-DATA-AREA P9-ENDP.

*
MAIN SECTION.
*
M1.
*   Ignore calls to Directory Database.
    IF P3-UDD = "UDD" GO TO MX.
*
*   Check Function Type
    IF P1-FUNCTION = "ADDVA" OR "ADDVB" OR "ADDVC"
        OR "ADDVR" OR "DELVD" OR "READV"
        OR "READD" OR "READR"
        PERFORM RELATED-FUNCTION.
*
*   Ignore any other function
MX.
    EXIT PROGRAM.
*
RELATED-FUNCTION SECTION.
*
RF1.
*   Convert Refer parameter value to hex
    MOVE P4-REFER TO REFER-CHARACTER.
    CALL "OTS$CVT_L_TZ" USING BY REFERENCE P4-REFER-COMP
                                BY DESCRIPTOR REFER-HEX
                                BY VALUE 8
                                OMITTED.
*
*   Ensure input Refer value is printable
    PERFORM CHECK-REFER-VALUE TEST BEFORE
        VARYING I FROM 1 BY 1 UNTIL I > 4.
*
*   Convert Control Key value to hex
    MOVE P6-CTRL-KEY TO KEY-CHARACTER.
    PERFORM CONVERT-KEY TEST BEFORE
        VARYING I FROM 1 BY 1 UNTIL I > 2.
```

```

*
* Ensure input Control Key value is printable
  PERFORM CHECK-KEY-VALUE TEST BEFORE
    VARYING I FROM 1 BY 1 UNTIL I > 8.
*
* Display parameters (element list & data area ignored)
  DISPLAY "Passed to PDM :- ".
  DISPLAY " Function : " P1-FUNCTION.
  DISPLAY " Status : " P2-STATUS.
  DISPLAY " Dataset : " P3-DATASET.
  DISPLAY " Refer : " REFER-CHARACTER " = " REFER-HEX.
  DISPLAY " Linkpath : " P5-LINKPATH.
  DISPLAY " Key : " KEY-CHARACTER " = "
    KEY-HEX-CHAR(1) " " KEY-HEX-CHAR(2).
  DISPLAY " Endp : " P9-ENDP.
*
RFX.
  EXIT.
*
CHECK-REFER-VALUE SECTION.
*
CRV1.
  IF REFER-CHAR(I) NOT NUMERIC
    AND REFER-CHAR(I) NOT ALPHABETIC
      MOVE "." TO REFER-CHAR(I).
CRVX.
  EXIT.
*
CONVERT-KEY SECTION.
*
CK1.
  CALL "OTS$CVT_L_TZ" USING BY REFERENCE P6-CTRL-KEY-COMP(I)
    BY DESCRIPTOR KEY-HEX-CHAR(I)
    BY VALUE 8
    OMITTED.
CKX.
  EXIT.
*
CHECK-KEY-VALUE SECTION.
*
CKV1.
  IF KEY-CHAR(I) NOT NUMERIC
    AND KEY-CHAR(I) NOT ALPHABETIC
      MOVE "." TO KEY-CHAR(I).
CKVX.
  EXIT.

```

**COBOL user exit 2**

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CSD_UPDM_USEREX2.
*
*   This is an example of an AFTER exit.
*   (an AFTER exit is called after the parameters are passed
*   back from SUPRA PDM's execution)
*   This example processes RELATED DML functions only.
*
ENVIRONMENT DIVISION.
*
DATA DIVISION.
*
WORKING-STORAGE SECTION.
*
01  I                                PIC S9(4)  COMP.
*
01  KEY-HEX.
    03  KEY-HEX-CHAR                PIC X(8)   OCCURS 2.
*
01  KEY-CHARACTER.
    03  KEY-CHAR                    PIC X     OCCURS 8.
*
01  REFER-CHARACTER.
    03  REFER-CHAR                  PIC X     OCCURS 4.
*
01  REFER-HEX                       PIC X(8).
*
LINKAGE SECTION.
*
01  NARGS                            PIC S9(9)  COMP.
01  P1-FUNCTION                       PIC X(5).
01  P2-STATUS                         PIC X(4).
01  P3-DATASET.
    03  P3-UDD                      PIC XXX.
    03  P3-FILLER                    PIC X.
01  P4-REFER.
    03  P4-REFER-COMP                PIC 9(9)  COMP.
01  P5-LINKPATH                       PIC X(8).
01  P6-CTRL-KEY.
    03  P6-CTRL-KEY-COMP              PIC 9(9)  COMP OCCURS 2.
*   For this example the Control Key (above) is 8 bytes
01  P7-ELEM-LIST                       PIC X(28).
*   For this example the Element list is 3 element names + END.
01  P8-DATA-AREA                       PIC X(24).
*   For this example the Data Area is 24 bytes.
01  P9-ENDP                           PIC X(4).

```

```
*
PROCEDURE DIVISION USING NARGS P1-FUNCTION P2-STATUS
                    P3-DATASET P4-REFER
                    P5-LINKPATH P6-CTRL-KEY
                    P7-ELEM-LIST P8-DATA-AREA P9-ENDP.

*
MAIN SECTION.
*
M1.
* Ignore calls to Directory Database.
  IF P3-UDD = "UDD" GO TO MX.
*
* Check Function Type
  IF P1-FUNCTION = "ADDVA" OR "ADDVB" OR "ADDVC"
    OR "ADDVR" OR "DELVD" OR "READV"
    OR "READD" OR "READR"
    PERFORM RELATED-FUNCTION.
*
* Ignore any other function
MX.
  EXIT PROGRAM.
*
RELATED-FUNCTION SECTION.
*
RF1.
* Convert Refer parameter value to hex
  MOVE P4-REFER TO REFER-CHARACTER.
  CALL "OTS$CVT_L_TZ" USING BY REFERENCE P4-REFER-COMP
    BY DESCRIPTOR REFER-HEX
    BY VALUE 8
    OMITTED.
*
* Ensure input Refer value is printable
  PERFORM CHECK-REFER-VALUE TEST BEFORE
    VARYING I FROM 1 BY 1 UNTIL I > 4.
*
* Convert Control Key value to hex
  MOVE P6-CTRL-KEY TO KEY-CHARACTER.
  PERFORM CONVERT-KEY TEST BEFORE
    VARYING I FROM 1 BY 1 UNTIL I > 2.
```

```

*
* Ensure input Control Key value is printable
  PERFORM CHECK-KEY-VALUE TEST BEFORE
      VARYING I FROM 1 BY 1 UNTIL I > 8.
*
* Display parameters (element list & data area ignored)
  DISPLAY "Passed back from PDM :- ".
  DISPLAY "  Function : " P1-FUNCTION.
  DISPLAY "  Status : " P2-STATUS.
  DISPLAY "  Dataset : " P3-DATASET.
  DISPLAY "  Refer : " REFER-CHARACTER " = " REFER-HEX.
  DISPLAY "  Linkpath : " P5-LINKPATH.
  DISPLAY "  Key : " KEY-CHARACTER " = "
      KEY-HEX-CHAR(1) " " KEY-HEX-CHAR(2).
  DISPLAY "  Endp : " P9-ENDP.
*
RFX.
  EXIT.
*
CHECK-REFER-VALUE SECTION.
*
CRV1.
  IF REFER-CHAR(I) NOT NUMERIC
    AND REFER-CHAR(I) NOT ALPHABETIC
    MOVE "." TO REFER-CHAR(I).
CRVX.
  EXIT.
*
CONVERT-KEY SECTION.
*
CK1.
  CALL "OTS$CVT_L_TZ" USING BY REFERENCE P6-CTRL-KEY-COMP(I)
      BY DESCRIPTOR KEY-HEX-CHAR(I)
      BY VALUE 8
      OMITTED.
CKX.
  EXIT.
*
CHECK-KEY-VALUE SECTION.
*
CKV1.
  IF KEY-CHAR(I) NOT NUMERIC
    AND KEY-CHAR(I) NOT ALPHABETIC
    MOVE "." TO KEY-CHAR(I).
CKVX.
  EXIT.

```

## COBOL command file to compile and link the exits

```
$!  
$! Command file to compile and link the sample COBOL PDM User Exit.  
$!  
$! Compile user exit 1 or Before Exit  
$ Cobol/noansi csi_pdm_exit1  
$!  
$! Compile User exit 2 or After Exit.  
$ Cobol/noansi csi_pdm_exit2  
$!  
$! Link the two compiled objects to form an Executable Image.  
$! Note that the names of the exits, CSD_UPDM_USEREX1 and  
$! CSD_UPDM_USEREX2 MUST be declared as UNIVERSAL symbols.  
$!  
$! To activate the User exits, do the following before running your  
$! SUPRAPDM Applications :  
$! DEFINE CSI_USEREX dev:[directory]CSI_PDM_EXIT.EXE  
$!  
$ Link/share=csi_pdm_exit.exe csi_pdm_exit1, csi_pdm_exit2, -supra_  
examples:CSI_USEREX.OPT/OPT  
  
$ Exit
```

Where the file SUPRA\_EXAMPLES:CSI\_USEREX.OPT has a VAX and an AXP version to take care of the option differences between systems:

### VAX version of SUPRA\_EXAMPLES:CSI\_USEREX.OPT

```
supra_examples:csi_userex.opt  
UNIVERSAL=CSD_UPDM_USEREX1  
UNIVERSAL=CSD_UPDM_USEREX2
```

### AXP version of SUPRA\_EXAMPLES:CSI\_USEREX.OPT

```
Supra_examples:csi_userex.opt  
symbol_vector=(CSD_UPDM_USEREX1=procedure,CSD_UPDM_USEREX2=proced  
ure)
```

## FORTRAN user exit

The FORTRAN user exit traces the primary and related user database functions to check that the PDML parameters are passed correctly between the application program and the PDM. As this example shows, you do not need to use both before and after exits.

### FORTRAN user exit

```

subroutine csd_updm_userex2(p0,p1,p2,p3,p4,p5,p6)
c
c SUPRAPDM After Exit example :
c This sample Fortran program traces some of the parameters
c after making a DML call to the user database files.
c
c
c Note that this exit uses Numeric dummy arguments
c because they are passed by reference from the PDM.
c Dummy Numeric arguments may not be Equivalenced to character
c strings, thus the mapping of the arguments from numerical
c addresses to character strings are done in two steps.
c
implicit none
character*5 function
character*4 stat, dataset
character*8 linkpath
integer*4 p0
real*8 p1,pp1
integer*4 p2,p3,pp2,pp3
real*8 p4,p5,pp5,p6
equivalence (pp1,function),(pp2,stat)
equivalence (pp3,dataset),(pp5,linkpath)
c
c
c pp1=p1
c pp2=p2
c pp3=p3
c
c Ensure we only trace calls to user databases by examining
c the dataset argument - this parameter is only available if
c it is one of a dataset access functions.
c
c if ( ((function .eq. 'ADD-M' ) .or.
1 (function .eq. 'DEL-M' ) .or.
2 (function .eq. 'READM' ) .or.
3 (function .eq. 'WRITM' )) .and.
4 ( dataset .ne. 'UDD1' ) .and.
5 ( dataset .ne. 'UDD2' ) .and.
6 ( dataset .ne. 'UDD3' ) ) goto 20
c
c if ( ((function .eq. 'ADDVA' ) .or.
1 (function .eq. 'ADDVB' ) .or.
2 (function .eq. 'ADDVC' ) .or.
3 (function .eq. 'ADDVR' ) .or.
4 (function .eq. 'DELVD' ) .or.
5 (function .eq. 'READD' ) .or.
6 (function .eq. 'READR' ) .or.
7 (function .eq. 'READV' ) .or.

```

```

8      (function .eq. 'WRITV')) .and.
9      ( dataset .ne. 'UDD1' ) .and.
1     ( dataset .ne. 'UDD2' ) .and.
2     ( dataset .ne. 'UDD3' ) ) goto 30
c
c     Ignore the rest of the functions
      goto 99
c
20     write (*,100) function,stat,dataset,p4
      goto 99
30     pp5=p5
      write (*,110) function,stat,dataset,linkpath,p6
99     return
c
100    format (' ',A5,' ',A4,' ',A4,' ',Z16)
110    format (' ',A5,' ',A4,' ',A4,' ',A8,' ',Z16)
      end

```

## FORTRAN command file to compile and link the exit

```

$!
$! Command file to compile and link the sample FORTRAN PDM User
Exit
$!
$! Please note that this example uses only the After exit
$!
$! Compile the Fortran program containing the After Exit.
$ FORTRAN CSI_PDM_EXIT_FOR
$!
$! Link the compile object to form an Executable Image.
$! Note that the name of the exit, CSD_UPDM_EXIT2 MUST be
declared
$! as UNIVERSAL Symbol.
$!
$! To activate the User Exit, do the following before running
your
$! SUPRAPDM applications :
$!
$!      $DEFINE CSI_USEREX dev:[directory]CSI_PDM_EXIT_FOR.EXE
$!
$! To deactivate the exit, DEASSIGN the logical name CSI_USEREX.
$ Link/share=CSI_PDM_EXIT_FOR.EXE CSI_PDM_EXIT_FOR.OBJ, -
supra_examples:CSI_USEREX.OPT/OPT
$ Exit

```

# B

## PDM statistics output

This Appendix describes the statistics written to the log file when you select STATISTICS=Y in the PDM input file. The log file is identified by the logical CSIPDMLOG.

### CSTI008S

```
(database), (process name), (task id), TASK SINOF STATISTIC  
READS    WRITES    ADDS    DELETES    MISC    HELDS    IHELDS  
(reads) (writes) (adds)  (deletes) (misc)  (helds) (int helds)
```

**Explanation:** The PDM produces this message whenever a task signs off. The columns have the following meaning:

- ◆ **READS** The total number of READ DML functions issued by the task.
- ◆ **WRITES** The total number of WRITE DML functions issued by the task.
- ◆ **ADDS** The total number of ADD DML functions issued by the task.
- ◆ **DELETES** The total number of DELETE DML functions issued by the task.
- ◆ **MISC** The total number of COMMIT, RESET, SINON, and SINOF DML functions issued by the task.
- ◆ **HELDS** The number of times any function issued by this task had to be retried because of a record hold. For example, if a function attempts to access a record that is already held for update, that function is backed out, a hold request is added to the queue for that record, and the function is placed in the retrying queue. When the record is free, the function is transferred to the allocating queue ready to be restarted.
- ◆ **IHELDS** The number of internal held conditions encountered. In practice, this value is the same as the HELDS column except that, in this case, the function was able to restart immediately instead of waiting in the retrying queue.

**CSTI011S**      (*database*), (*data set*), DML FILE FUNCTION STATISTICS,  
READS            WRITES            ADDS            DELETES            MISC  
(*reads*)        (*writes*)        (*adds*)        (*deletes*)        (*misc*)

**Explanation:** The PDM produces this message whenever it physically closes a file. A file is physically closed when the last task to use it issues a logical close for that file. This could be as the last task signs off during normal processing or as a result of an operator command to close down the database or the PDM (SHUTDOWN, UNLOAD).

The columns have the following meaning:

- ◆ **READS** The total number of READ DML functions used on the file since it was physically opened.
- ◆ **WRITES** The total number of WRITE DML functions used on the file since it was physically opened.
- ◆ **ADDS** The total number of ADD DML functions used on the file since it was physically opened.
- ◆ **DELETES** The total number of DELETE DML functions used on the file since it was physically opened.
- ◆ **MISC** The total number of RQLOC DML functions used on the file since it was physically opened.

**CSTI012S**

(database), (data set), PHYSICAL FILE I/O & MISC FILE STATISTICS,

|                             |                             |                        |                          |
|-----------------------------|-----------------------------|------------------------|--------------------------|
| BUFSNAVL<br>(buf not avail) | BFLUSHES<br>(buf flushes)   | PREADS<br>(phys reads) | PWRITES<br>(phys writes) |
| LREADS<br>(logical reads)   | LWRITES<br>(logical writes) | HELDS<br>(helds)       | IHELDS<br>(int helds)    |

**Explanation:** This message is issued with CSTI011L. The columns have the following meaning:

- ◆ **BUFSNAVL** The number of times a logical read function had to wait for a buffer to be freed.
- ◆ **BFLUSHES** The number of times a logical read function had to flush a buffer to use it. A read function first searches the pool of buffers for the record. If the record is not there, the read function searches for an empty buffer to use. If no empty buffer is available, the read function searches for an unlocked buffer. A BFLUSH occurs when an unlocked buffer is found that contains modified records which must be flushed before the buffer can be used.
- ◆ **PREADS** The number of physical reads performed on the file. A physical read reads in as much of the file as can be contained in one buffer. In VMS, for example, 25 physical reads would be needed to read a file that has 50 VMS sectors and a buffer size of 1024 bytes. If a file has too few buffers, or if the buffers are too small, then the number of physical reads needed increases.
- ◆ **PWRITES** The number of physical writes performed on the file. Normally this value equals the number of BFLUSHES plus the number of buffers.
- ◆ **LREADS** The number of logical read operations performed on the file. The target of a logical read is one database record which may be found in one of the buffers for the file. A physical read is performed if the database record is not found in a buffer. The number of logical reads should therefore greatly exceed the number of physical reads. The number of logical reads should also exceed the number of DML functions since, for example, one ADDVR could generate three or four logical reads.

- ◆ **LWRITES** The number of logical writes used on the file. Currently, this column gives useful figures only for the task log and system log.
  
- ◆ **HELDS** The number of times any function issued for this file had to be retried because of a record hold. For example, if a function attempts to access a record that is already held for update, that function is backed out, a hold request is added to the queue for that record, and the function is placed in the retrying queue. When the record is free, the function is transferred to the allocating queue ready to be restarted.
  
- ◆ **IHELDS** The number of internal held conditions encountered. In practice, this value is the same as the HELDS column, except in this case, the function was able to restart immediately instead of waiting in the retrying queue.

**CSTI055S**

(*database*), DATABASE STATISTICS,

|                   |                   |                           |                  |                  |
|-------------------|-------------------|---------------------------|------------------|------------------|
| SINONS            | SINOFs            | DYNSINOFs                 | LFULs            | DFULs            |
| ( <i>sinons</i> ) | ( <i>sinofs</i> ) | ( <i>dynamic sinofs</i> ) | ( <i>lfuls</i> ) | ( <i>dfuls</i> ) |

**Explanation:** The PDM produces this message when the last task signs off from the specified database. The columns have the following meaning:

- ◆ **SINONS** The number of SINON DML functions issued for the database.
- ◆ **SINOFs** The number of SINOF DML functions issued for the database.
- ◆ **DYNSINOFs** The number of dynamic SINOFs issued for the database. Dynamic SINOFs include applications which exit without signing off or applications which are signed off as a result of an operator command.
- ◆ **LFULs** Reserved for future use.
- ◆ **DFULs** Reserved for future use.

**CSTI056S***(database)*, DATABASE STATISTICS,

| CONTASKS      | CONFUNCS      | THREADS    | NFILFNCS      | TLFBUFSTLS    | CONUPTSKS     |
|---------------|---------------|------------|---------------|---------------|---------------|
| <i>nnnnnn</i> | <i>nnnnnn</i> | <i>nnn</i> | <i>nnnnnn</i> | <i>nnnnnn</i> | <i>nnnnnn</i> |

**Explanation:** This message is produced with CSTI055I. The columns have the following meaning:

- ◆ **CONTASKS** The maximum number of tasks concurrently signed on to the database.
- ◆ **CONFUNCS** The maximum number of functions concurrently issued to the database. Since a task can issue only one function at a time, the CONFUNCS figure should be less than or equal to the CONTASKS figure.
- ◆ **THREADS** The maximum number of threads concurrently active, executing a function on the database.
- ◆ **NFILFNCS** Reserved for future use.
- ◆ **TLFBUFSTLS** The number of times a task log file (TLF) buffer had to be “stolen.” When a thread attempts to acquire a TLF buffer, it first tries to get a free buffer. If none are available, the task must steal a buffer that is not currently locked after first flushing it.
- ◆ **CONUPTSKS** The maximum number of update tasks concurrently signed on to the database.

# C

## Example mailbox-reading program

This appendix presents a COBOL program to read the PDM messages from a mailbox. In this example, the mailbox is SUPRA24\_000114, constructed using the equivalence name for CSI\_PDMID (SUPRA24 in this case) and the UIC group in which the PDM is running (000114 in this case). Substitute your own mailbox name to use this program.

Before you can read PDM messages from a mailbox, you must:

- ◆ Set the PDM input file parameter MRELAY=Y to send all PDM messages from CSIPDM to your mailbox.
- ◆ Define the logical name CSI\_MRELAY to TRUE to send all console messages from CSIDAP to your mailbox.

See “[Automating operator communication](#)” on page 183 for a detailed description of how to write a user interface to the SUPRA Server PDM.

## Example Mailbox-Reading Program, MAIL-BOX-TRAP.COB

```

*****
*
*
*   Program : mail-box-trap.cob
*
*
*   Sample program to read messages from a mailbox which
*   SUPRAPDM sends operator messages to.
*
*
*   To compile and link
*       $ Cobol/Noansi   Mail-box-trap
*       $ Link Mail-box-trap
*
*
*   To execute
*       $ define/table=lnm$process_directory -
*         lnm$temporary_mailbox lnm$group
*       $ Run mail-box-trap
*
*
*
*****
Identification division.
Program-id.  mail_box_trap.
*
*
Data division.
*
Working-storage section.
*
01   stat                pic 9(9) comp.
01   max_msg             pic 9(4) comp value 1024.
01   buf_quo             pic 9(4) comp value 2048.
01   mbx_name            pic x(15) value "          " .
01   mbx_chan            pic 9(4) comp.
01   mbx_mess            pic x(132).
01   msg_len             pic 9(9) comp value 132.
01   pid_in_hex         pic x(8).
01   iosb.
03   iosb-stat          pic 9(4) comp.
03   iosb-count         pic 9(4) comp.
03   iosb-devdata       pic 9(9) comp.
01   nothing            pic 9(9) comp value 0.
01   io_func            pic 9(9) comp value 49.
*
*                               IO$_READVBLK.
*
Procedure division.

Start-program.

```

```

Perform get_mail_box_id.

Perform create_mail_box.
*
Infinite_loop.
*
*   This is the main code that reads the mailbox messages.
*   Instead of using SYS$QIOW, you can use SYS$QIO with
*   a completion AST, so that your program can perform
*   another useful function while it is waiting for a
*   mailbox message.
*
*   Move spaces to mbx_mess.
*   call "SYS$QIOW" using      by value      nothing,
*                               mbx_chan,
*                               io_func,
*                               by reference iosb,
*                               by value     nothing,
*                               by value     nothing,
*                               by reference mbx_mess,
*                               by value     msg_len,
*                               by value     nothing,
*                                       nothing,
*                                       nothing,
*                                       giving stat.
*
*   if stat not = 1
*       perform read_error.
*
*   Convert the PID of the sender to hexadecimal format.
*
*   Call "OTS$CVT_L_TZ"      Using      By Reference iosb-devdata,
*                               By Descriptor pid-in-hex,
*                               By Value 8,
*                               nothing.
*
*   Display the message received.
*
*   Display "Suprapdm Message received from PID ", pid-in-hex
*   UNDERLINED.
*   Display mbx_mess.
*   Display ".
*
*   Go to read another message
*
*   Go to infinite_loop.
*
get_mail_box_id.

```

```

*
*   This section gets the mailbox name from SYS$INPUT.
*   The mailbox name is made up of the form xxxxxxxx_nnnnnn
*   where
*       xxxxxxxx is the translated value of CSI_SYSPDMID or
CSI_PDMID,
*       nnnnnn  is the group number of the PDM if the PDM is
*       group-wide, 000000 if the PDM is system wide or multiple
*       system-wide.
*
*   e.g., SUPRA24_000114
*   if CSI_PDMID is "SUPRA24" and PDM is running in Group
000114.
*
*   Display "Please Enter Mail Box Name :" Underlined with no
advancing.
*   Accept mbx_name.
*   Call "STR$UPCASE" using by descriptor mbx_name,
*       mbx_name.
*
*
*
*   This section calls the system service SYS$CREMBX to create
*   a mailbox with the name obtained.
*
create_mail_box.
*       call "SYS$CREMBX" using by value nothing,
*                               by reference mbx_chan,
*                               by value max_msg,
*                               by value buf_quo,
*                               by value nothing,
*                               by value nothing,
*                               by descriptor mbx_name,
*                               giving stat.
*
*       if stat not = 1
*           call "SYS$EXIT" using by value stat.
*
*       Display "Commencing read from mailbox " with no
advancing.
*       Display mbx_name BOLD.
*       Display "< Terminate by pressing CTRL/Y. >" BOLD.
*
*
*
read_error.
*       call "SYS$DASSGN" using      by value      mbx_chan.
*       Display "Error reading from Mailbox, status is ".
*       call "SYS$EXIT" using by value stat.
*       exit program.
*

```

# D

## Optional SUPRA Server logicals

The following table lists optional SUPRA Server logicals. These logicals change the normal behavior of SUPRA Server. The logicals can be placed in any logical name table accessible to your process.

| Logical name              | Equivalence name | Function                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CSI_ALLOW_DUP_RECORD_CODE | TRUE             | Allows duplicate record codes in a user element list as was needed by ULTRA.                                                                                                                                                                                                                                                                                                                                                              |
| CSI_REINIT_ON_SINON       | TRUE             | Forces the logical CSI_PREFIX to be translated at each SINON, allowing an application to SINOF, redefine CSI_PREFIX, then SINON again using a different database.                                                                                                                                                                                                                                                                         |
| CSI_SCHEDULED_WAKEUP      | FALSE            | When TRUE (the default and previous behavior) then the datbas client does scheduled wakeup system service calls to guarantee that the client never hangs in HIB state due to a missed wakeup call. However, these extra system calls can in some situations incur a substantial performance overhead. Optionally setting this logical name to FALSE disables the scheduled wakeup calls at the expense of potential client process hangs. |
| CSUBGRN_CONTINUE_ON_ERROR | TRUE             | BGRN will continue to execute regardless of the number of errors.<br>Warning: Use of this logical could result in an infinite loop.                                                                                                                                                                                                                                                                                                       |
| DBAID_HELP_NOSPAWN        | TRUE             | Changes the CSVDBAID HELP command to call LBR\$OUTPUT_HELP instead of spawning to do a DCL \$HELP.                                                                                                                                                                                                                                                                                                                                        |



# E

## SUPRA Server logical names

The table later in this section lists the logical names needed to run SUPRA Server. The Table column gives the logical name table in which you can place the logical name:

- ◆ **P** Process table
- ◆ **G** Group table
- ◆ **S** System table
- ◆ **MSW** Multiple systemwide shareable table, which is identified by the logical name table `CSI_PDM_pdmname`.
- ◆ **G/S** Group table if you are running a groupwide PDM, system table if you are running a systemwide PDM.
- ◆ **G/S/MSW** Group table if you are running a groupwide PDM, system table if you are running a systemwide PDM, or the multiple systemwide table if you are running a multiple systemwide PDM.
- ◆ **Any** It does not matter which table, provided you define the logical name in a logical name table accessible to your process.

For more information on logical names, see [“Defining your operating environment”](#) on page 57. For more information about logical name tables, see [“Configuring the PDM”](#) on page 33.

The scope of a logical name is defined by the logical name tables in which it appears. One logical name can appear in more than one logical name table.

A logical name which occurs in more than one table can be associated with a different equivalence name in each logical name table. To avoid confusion, the system searches logical name tables in a particular order and uses the first logical name found, regardless of any other occurrences. The default logical name table search sequence is PROCESS, JOB, GROUP, and finally SYSTEM. However, you can change this order to suit your requirements. Refer to your VMS documentation.

In the following table, several equivalence names contain the variable *svc/vl*. This variable is replaced with the service level of SUPRA Server.

| Logical name       | Table   | Equivalence name                          | Function                                                          |
|--------------------|---------|-------------------------------------------|-------------------------------------------------------------------|
| BATCH_GLOBAL_INPUT | Any     | dev:[dir]filename.ext                     | Input text file used to create a global view file.                |
| BGRN               | Any     | SUPRA_EXE:<br>CSUBGRN_ <i>svc/vl</i> .EXE | Provides background DBA utilities.                                |
| CHANGEDB           | Any     | SUPRA_COMS:<br>CHANGEDB.CLD               | Command definition file for Fast utilities.                       |
| COMBAT             | Any     | SUPRA_COMS :<br>COMBAT.CLD                | Command definition file for validate, compile, and print program. |
| CSDUSSERV          | G/S     | SUPRAPDM                                  | Used by users who upgraded from ULTRA.                            |
| CSI_*              | G/S     | dev:[dir]PDM_START_*.COM                  | PDM start-up command procedure.                                   |
| CSI_AUTOSTART      | Any     | YES or NO                                 | Enable/disable automatic PDM start-up.                            |
| CSI_CONSOLE        | G/S/MSW | OPERn                                     | Operator console that CSIDAP messages are sent to.                |

\* Refers to the 6-digit group number for a groupwide PDM, 000000 for a systemwide PDM, or the translation of CSI\_SYSPDMID for a multiple systemwide PDM.

| Logical name          | Table   | Equivalence name                           | Function                                                              |
|-----------------------|---------|--------------------------------------------|-----------------------------------------------------------------------|
| CSIDAP                | G/S     | SUPRA_EXE:<br>CSIDAP_svc/vl.EXE            | PDM interface module.                                                 |
| CSIDAP_DEB            | Any     | SUPRA_EXE<br>CSIDAP_DEB_svc/vl.EXE         | Used for testing by Cincom Support.                                   |
| CSIDBA                | Any     | SUPRA_EXE:<br>CSIDBA_svc/vl.EXE            | Provides DBA facilities.                                              |
| CSI_DBA               | Any     | SUPRA_REPORT                               | Directory containing Batch Directory Maintenance and Directory Views. |
| CSIDBAUTL             | Any     | SUPRA_EXE:<br>CSIDBAUTL_svc/vl.EXE         | Provides online DBA utilities.                                        |
| CSIDBPDM              | Any     | SUPRA_AUXIL:<br>CSIDBPDM_svc/vl.EXE        | Database testing tool.                                                |
| CSIDBVER              | Any     | SUPRA_EXE<br>CSIDBVER_svc/vl.EXE           | Database integrity verification utility.                              |
| CSIDDLLOAD            | Any     | SUPRA_EXE:<br>CSIDDLLOAD_svc/vl.EXE        | Loads TOTAL-compatible DDL.                                           |
| [xxx_]CSI_DIRDB       | G/S/MSW | dev:[dir]                                  | Location of Directory database (SUPRAD).                              |
| CSI_DMPANL            | G/S/MSW | dev:[dir]CSI_DMP.ANL                       | Location of PDM crash dump file.                                      |
| CSIDMPANL             | Any     | SUPRA_EXE:<br>CSIDMPANL_svc/vl.EXE         | Provides a PDM dump analysis.                                         |
| CSI_EXEC_DISPATCH     | G/S     | SUPRA_EXE:CSI_EXEC_DISPATCH_svc/vl.EXE     | Executive mode dispatcher.                                            |
| CSI_EXEC_DISPATCH_DEB | Any     | SUPRA_EXE:CSI_EXEC_DISPATCH_DEB_svc/vl.EXE | Used for testing by Cincom Support.                                   |
| CSI_FINDPDM           | G/S     | dev:[dir]CSI_FINDPDM.COM                   | Command file to enable automatic CSI_FINDPDM start-up.                |
| CSIGIM                | G/S     | SUPRA_EXE:<br>CSIGIM_svc/vl.EXE            | General Interface Module.                                             |

| Logical name                | Table | Equivalence name                                     | Function                                                       |
|-----------------------------|-------|------------------------------------------------------|----------------------------------------------------------------|
| CSIGIM_DEB                  | Any   | SUPRA_EXE:<br>CSIGIM_DEB_svc/vl.EXE                  | Used for testing by Cincom Support.                            |
| CSIINDEX                    | Any   | SUPRA_COMS:CSIINDEX<br>.CLD                          | Command definition file for index utilities.                   |
| CSI_KERNEL_<br>DISPATCH     | G/S   | SUPRA_EXE:CSI_<br>KERNEL_DISPATCH_<br>svc/vl.EXE     | Kernel mode dispatcher.                                        |
| CSI_KERNEL_<br>DISPATCH_DEB | Any   | SUPRA_EXE:CSI_<br>KERNEL_DISPATCH_<br>DEB_svc/vl.EXE | Used for testing by Cincom Support.                            |
| CSILOCKS                    | Any   | SUPRA_AUXIL:<br>CSILOCKS_svc/vl.EXE                  | Reports on VMS locks granted to a process.                     |
| CSI_MRELAY                  | Any   | TRUE or FALSE                                        | Send CSIDAP to a mailbox to be read by a user-written program. |
| CSI_NODIRECTORY             | Any   | TRUE or FALSE                                        | Prevents RDM from signing on to the SUPRA Server Directory.    |
| CSIOAUTH                    | Any   | SUPRA_EXE:<br>CSIOAUTH_svc/vl.EXE                    | CSIOPCOM command authorization program.                        |
| CSIOPCOM                    | Any   | SUPRA_EXE:<br>CSIOPCOM_svc/vl.EXE                    | Alternative to VMS OPCOM utility.                              |
| CSIOPCOM_AUTH               | Any   | dev:[dir]CSIOPCOM_<br>AUTH.LIS                       | CSIOPCOM command authorization file.                           |
| CSIOPCOM_SNAPS              | Any   | dev:[dir]CSIOPCOM_<br>SNAPS.LIS                      | File for dumped CSIOPCOM screens.                              |
| CSIPDM                      | G/S   | SUPRA_EXE:<br>CSIPDM_svc/vl.EXE                      | PDM image executed by the detached PDM process.                |

| Logical name             | Table   | Equivalence name                   | Function                                                                                              |
|--------------------------|---------|------------------------------------|-------------------------------------------------------------------------------------------------------|
| CSIPDM_DEB               | Any     | SUPRA_EXE:<br>CSIPDM_DEB_svc\I.EXE | Used for testing by Cincom Support.                                                                   |
| CSI_PDMID                | G/S     | 1–8 character name                 | Identifies the name of the PDM for group or systemwide PDMs.                                          |
| CSIPDMINP                | G/S/MSW | <i>dev:[dir]</i> PDM_OPTIONS_*.INP | PDM input parameter file.                                                                             |
| CSIPDMLOG                | G/S/MSW | <i>dev:[dir]</i> CSIPDM.LOG        | Log file for PDM messages.                                                                            |
| CSIPDM_PATCH             | G/S/MSW | <i>(security patch contents)</i>   | Security patch logical for the PDM.                                                                   |
| CSI_PREFIX               | Any     | 1–3 character prefix               | Specifies the prefix used to distinguish databases of the same name used in the same group or system. |
| CSI_READAHEAD            | Any     | YES (or TRUE) or NO (or FALSE)     | Use readahead buffering.<br>Default = YES                                                             |
| CSI_READAHEAD_STATISTICS | Any     | YES (or TRUE) or NO (or FALSE)     | Print read-ahead statistics at sign off. Statistics are printed to CSI_CONSOLE.<br>Default = NO       |
| CSI_RMS_RU_ON            | Any     | TRUE or FALSE                      | Enable RMS Journaling.                                                                                |
| CSI_SCHEMA               | Any     | 6-character <i>database name</i>   | Database name used by all RDM applications including DBAID and Global View creation.                  |

\* Refers to the 6-digit group number for a groupwide PDM, 000000 for a systemwide PDM, or the translation of CSI\_SYSPDMID for a multiple systemwide PDM.

| Logical name | Table   | Equivalence name                         | Function                                    |
|--------------|---------|------------------------------------------|---------------------------------------------|
| CSI_SMENU    | Any     | SUPRA_EXE:<br>CSI_SMENU_svc/vl.EXE       | Menu for SUPRA Server components.           |
| CSISTR       | G/S     | SUPRA_EXE:<br>CSISTR_svc/vl.EXE          | Starts up the detached PDM process.         |
| CSISTRINP    | G/S/MSW | dev:[dir]CSISTR.INP                      | PDM start-up resource file.                 |
| CSISTRLOG    | G/S/MSW | dev:[dir]CSISTR.LOG                      | Output file for PDM start-up program.       |
| CSI_SYSPDMID | Any     | 1–8 character name                       | Name of multiple systemwide PDM.            |
| CSI_USEREX   | Any     | filename.EXE                             | PDM user exit.                              |
| CSI_VAL_EXIT | Any     | dev:[dir]image-name.EXE                  | RDM Domain Validation Exit.                 |
| CSI_WILD_EN  | Any     | (any valid character)                    | Equal or next wild card character.          |
| CSI_WILD_EQ  | Any     | (any valid character)                    | Equal only wild card character.             |
| CSMCHANGEDB  | Any     | SUPRA_EXE:<br>CSMCHANGEDB_svc/vl.<br>EXE | Fast Utilities program.                     |
| CSMCOMBAT    | Any     | SUPRA_EXE:<br>CSMCOMBAT_svc/vl.EXE       | Batch validate, compile, and print program. |
| CSTUDSLF     | G/S/MSW | SUPRA_EXE:<br>CSTUDSLF_svc/vl.EXE        | System log dump program.                    |
| CSTUFMT      | Any     | SUPRA_EXE:<br>CSTUFMT_svc/vl.EXE         | Stand-alone format program.                 |
| CSTUFMTSHR   | Any     | SUPRA_EXE:<br>CSTUFMTSHR_svc/vl.<br>EXE  | Shareable format image.                     |
| CSTUIDX      | Any     | SUPRA_EXE:<br>CSTUIDX_svc/vl.EXE         | Stand-alone index maintenance utility.      |

| Logical name              | Table   | Equivalence name                        | Function                                                     |
|---------------------------|---------|-----------------------------------------|--------------------------------------------------------------|
| CSTUIDXSHR                | Any     | SUPRA_EXE<br>CSTUIDXSHR_svc/vl.EXE      | Shareable index maintenance image.                           |
| CSTURCV                   | Any     | SUPRA_EXE:<br>CSTURCV_svc/vl.EXE        | Stand-alone recovery program.                                |
| CSTURCVSHR                | Any     | SUPRA_EXE:<br>CSTURCVSHR_svc/vl.<br>EXE | Shareable recovery image.                                    |
| CSVBASIC                  | Any     | SUPRA_EXE:<br>CSVBASIC_svc/vl.EXE       | BASIC RDML preprocessor.                                     |
| CSVCOBOL                  | Any     | SUPRA_EXE:<br>CSVCOBOL_svc/vl.EXE       | COBOL RDML preprocessor.                                     |
| CSVDBAID                  | Any     | SUPRA_EXE:<br>CSVDBAID_svc/vl.EXE       | DBAID utility.                                               |
| CSVFORTRA                 | Any     | SUPRA_EXE:<br>CSVFORTRA_svc/vl.EXE      | FORTRAN RDML preprocessor.                                   |
| CSVGLOBAL                 | Any     | SUPRA_EXE:<br>CSVGLOBAL_svc/vl.EXE      | Global view creation utility.                                |
| CSVIPLVS                  | G/S     | SUPRA_EXE:<br>CSVIPLVS_svc/vl.EXE       | SUPRA Server Relational Data Manager (RDM).                  |
| CSVIPLVS_DEB              | G/S     | SUPRA_EXE:CSVIPLVS_<br>DEB_svc/vl.EXE   | Used for testing by Cincom Support.                          |
| CSVLINK                   | Any     | CSVLINK SUPRA_COMS:<br>CSVLINK.COM      | Command file to link RDM application programs.               |
| CSXSCREEN                 | Any     | SUPRA_EXE:<br>CSXSCREEN_svc/vl.EXE      | Screen handler.                                              |
| [xxx_]dbname              | G/S/MSW | dev:[dir]filename.MOD                   | Points to compiled database description file.                |
| [xxx_]dbname_CSI_PDM_MACS | Any     | list of machines                        | Preferred machine list for a database.                       |
| DBAEDT                    | Any     | SUPRA_EXE:<br>DBAEDT.EDT                | Customized DBA EDT environment for Logical View Maintenance. |

| Logical name         | Table | Equivalence name                   | Function                                                        |
|----------------------|-------|------------------------------------|-----------------------------------------------------------------|
| DBVER                | Any   | SUPRA_COMS:<br>CSIDBVER.CLD        | Command verb for CSIDBVER utility.                              |
| DUMPSLF_[xxx_]dbname | G/S   | dev:[dir]filename.ext              | Location of the system log dump input file.                     |
| GETLOCKS             | Any   | SUPRA_AUXIL:<br>GETLOCKS.CLD       | Command verb for CSILOCKS utility.                              |
| GVSHEMA              | Any   | dev:[dir]filename.GBL              | Specifies the global view file.                                 |
| GVSHEMA_SYS          | Any   | TRUE or FALSE                      | Indicates that the global view file is to be loaded systemwide. |
| RUNBASIC             | Any   | SUPRA_COMS:<br>RUNBASIC.COM        | Command file to run the BASIC RDML preprocessor.                |
| RUNCOBOL             | Any   | SUPRA_COMS:<br>RUNCOBOL.COM        | Command file to run COBOL RDML preprocessor.                    |
| RUNCSV               | Any   | RUNCSV SUPRA_COMS:<br>RUNCSV.COM   | Command file to run an RDM application.                         |
| RUNDBAID             | Any   | SUPRA_COMS:<br>RUNDBAID.COM        | Command file to run DBAID.                                      |
| RUNDIRM              | Any   | SUPRA_COMS:RUNDIRM<br>.COM         | Command file to run DIRM.                                       |
| RUNFORTRA            | Any   | SUPRA_COMS:<br>RUNFORTRA.COM       | Command file to run FORTRAN RDML preprocessor.                  |
| SUPRA                | Any   | SUPRA_EXE:<br>CSI_SMENU_svc/vl.EXE | SUPRA Server main menu.                                         |
| SUPRA_AUXIL          | Any   | dev:[dir.SUPRA.PDM_24:<br>AUXIL]   | Directory containing SUPRA Server utilities.                    |
| SUPRA_BASE           | Any   | dev:[dir.SUPRA.PDM_24]             | Base SUPRA Server directory.                                    |

| Logical name              | Table   | Equivalence name                         | Function                                                                |
|---------------------------|---------|------------------------------------------|-------------------------------------------------------------------------|
| SUPRA_BURRYS              | Any     | <i>dev:[dir.SUPRA.PDM_24.BURRYS]</i>     | Directory containing the BURRYS database.                               |
| SUPRA_CLEAN_DICT          | Any     | <i>dev:[dir.SUPRA.PDM_24.CLEAN_DICT]</i> | Directory containing an unused Directory database SUPRAD.               |
| SUPRA_CLEAN_EXE           | Any     | <i>dev:[dir.SUPRA.PDM_24.CLEAN_EXE]</i>  | Directory not currently used.                                           |
| SUPRA_COMS                | Any     | <i>dev:[dir.SUPRA.PDM_24.COMS]</i>       | Directory containing SUPRA Server command procedures.                   |
| [xxx_]SUPRAD              | G/S/MSW | CSI_DIRDB:<br>SUPRAD.MOD                 | Directory compiled database description.                                |
| [xxx_]SUPRAD_CSI_PDM_MACS | G/S/MSW | list of machines                         | Preferred machine list for Directory compiled database description.     |
| SUPRA_DICT                | Any     | <i>dev:[dir.SUPRA.PDM_24.DICT]</i>       | Directory available for your use to hold a Directory database (SUPRAD). |
| SUPRA_EXAMPLES            | Any     | <i>dev:[dir.SUPRA.PDM_24.EXAMPLES]</i>   | Directory containing example command procedures and information.        |
| SUPRA_EXE                 | G/S     | <i>dev:[dir.SUPRA.PDM_24.EXE]</i>        | Directory containing SUPRA Server images.                               |
| SUPRA-HELP                | Any     | SUPRA_HELP:<br>SUPRA-HELP.HLB            | Help library for SUPRA Server.                                          |
| SUPRA_HELP                | Any     | <i>dev:[dir.SUPRA.PDM_24.HELP]</i>       | Directory for the SUPRA Server help library.                            |

| Logical name      | Table   | Equivalence name                       | Function                                                                                  |
|-------------------|---------|----------------------------------------|-------------------------------------------------------------------------------------------|
| SUPRA_LIBRARY     | Any     | <i>dev:[dir.SUPRA.PDM_24.LIBRARY]</i>  | Directory for the SUPRA Server database Class configuration files.                        |
| SUPRA_PATCH_WORK  | Any     | <i>dev:[dir.SUPRA.PATCH_WORK]</i>      | Area provided for applying maintenance.                                                   |
| SUPRAPDM          | G/S     | CSIGIM                                 | Used to link the SUPRA Server shareable image with applications.                          |
| SUPRA_PDM_PATCHES | Any     | <i>dev:[dir.SUPRA.PDM_24.PATCHES]</i>  | Area containing patches for SUPRA Server images (VAX) and security codes (VAX and Alpha). |
| SUPRA_REPORT      | Any     | <i>dev:[dir.SUPRA.PDM_24.REPORT]</i>   | Directory containing Batch Directory Maintenance and Directory Views.                     |
| SUPRA_TEST_EXE    | G/S     | <i>dev:[dir.SUPRA.PDM_24.TEST_EXE]</i> | Directory for test versions of the SUPRA Server images.                                   |
| SUPRA_UPGRADE     | G/S     | <i>dev:[dir.SUPRA.PDM_24.UPGRADE]</i>  | Directory containing SUPRA Server upgrade procedures.                                     |
| SYS\$ULTRA        | G/S/MSW | CSI_DIRDB                              | For users who upgraded from ULTRA.                                                        |
| ULTRADBMS         | G/S     | SUPRAPDM                               | For users who upgraded from ULTRA 1.4.                                                    |
| ULTRAPDM          | G/S     | SUPRAPDM                               | For users who upgraded from ULTRA 1.5.                                                    |

# Index

## [

[xxx\_]CSI\_DIRDB logical 90  
[xxx\_]dbname logical 94  
[xxx\_]dbname\_CSI\_PDM\_MACS  
logical 95  
[xxx\_]SUPRAD logical 96, 97

## A

access control lists (ACL) 141  
bypassing 143  
defining 144  
access method  
choosing 228  
defining 208  
ACLCHECK, PDM input  
parameter 126  
using with UICCHECK 139  
ACTIVATE, an index 147  
administration utilities 25  
LOGICALS.COM 26, 63  
PDM\_LOGICALS\_\*.COM 26,  
87  
SUPRA\_SYMBOL.COM 26, 57  
SUPRA\_SYSTEM.COM 26, 58  
application programs, designing  
232  
AST\_LIMIT, setting 118  
AUTOGEN, using 111  
automatic PDM initiation 42  
creating an input file for 125  
automatic restart 21

## B

BASIC preprocessor, logical  
name for 78  
BATCH, interface to CSIOPCOM  
182  
BATCH\_GLOBAL\_INPUT logical  
99  
BGRN logical 70

binding  
data items 242  
views 229  
binding views 24  
block size, calculating to optimize  
performance 211  
bound views 229  
BUFFER\_LIMIT 119  
buffers  
managing 217  
search algorithms 220

## C

call-by-reference 56  
chains  
avoiding fragmentation 208  
linkpath 214  
synonym 210  
change files 204  
CHANGEDB command, using on  
UDD files 204  
CHANGEDB logical 86  
checking indices 194  
clustered PDM access 22  
COBOL preprocessor, logical  
name for 78  
coded records, using 215  
COMBAT logical 86  
COMMITTS, optimizing frequency  
of 237  
communicating with the PDM  
through CSIOPCOM 174  
through the REPLY command  
189  
components, SUPRA 20  
console, displaying PDM  
messages at 40, 90  
CONSOLE, PDM input  
parameters 126  
contiguous disk files 208  
control interval, defining 213  
CSDUSSERV logical 70  
CSI\_\* logical 89  
CSI\_ALLOW\_DUP\_RECORD\_C  
ODE logical 279  
CSI\_AUTOSTART  
logical 89  
PDM initiation parameter 39  
CSI\_CONSOLE logical 90  
CSI\_DBA logical 71  
CSI\_DMPANL logical 90  
CSI\_EXEC\_DISPATCH logical  
73

- CSI\_EXEC\_DISPATCH\_DEB
    - logical 73
  - CSI\_FINDPDM
    - logical 100
    - to locate active PDM 48
  - CSI\_KERNEL\_DISPATCH logical 74
  - CSI\_KERNEL\_DISPATCH\_DEB
    - logical 74
  - CSI\_MRELAY logical 91, 100
  - CSI\_NODIRECTORY logical 101
  - CSI\_PDMMID logical 92
  - CSI\_PREFIX logical 102
  - CSI\_REINIT\_ON\_SINON logical 279
  - CSI\_RMS\_RU\_ON logical 103
  - CSI\_SCHEMA, specifying a database 50
  - CSI\_SYSPDMID logical 109
  - CSI\_USEREX logical 104
  - CSI\_VAL\_EXIT logical 104
  - CSI\_WILD\_EN logical 104
  - CSI\_WILD\_EQ logical 105
  - CSIDAP logical 42, 70
  - CSIDAP\_DEB logical 71
  - CSIDBA logical 71
  - CSIDBAUTL logical 71
  - CSIDBLLOAD logical 72
  - CSIDBPDM logical 71
  - CSIDBVER logical 72
  - CSIDMPANL logical 72
  - CSIGIM logical 73
  - CSIGIM\_DEB logical 74
  - CSIINDEX logical 74
  - CSIOLOCKS logical 75
  - CSIOAUTH logical 75
  - CSIOCOM\_AUTH logical, to give access to PDM commands 91
  - CSIOPCOM
    - batch interface 182
    - displaying pop-up menus 178
    - dumping screens 179
    - function key support 176
    - initiating 174
    - LIST command 180
    - SETcommand 180
  - CSIOPCOM logical 75
  - CSIOPCOM\_SNAPS logical 91
  - CSIPDM logical 75
  - CSIPDM\_DEB logical 75
  - CSIPDM\_PATCH logical 76
  - CSIPDMINP logical 92
  - CSIPDMLOG logical 93
  - CSISMENU logical 76
  - CSISTR logical 76
  - CSISTRINP logical 103
  - CSISTRLOG logical 93
  - CSMCHANGEDB logical 76
  - CSMCOMBAT logical 77
  - CSTFMTSHR logical 77
  - CSTUDSLF logical 77
  - CSTUFMT logical 77
  - CSTUIDX logical 78
  - CSTUIDXSHR logical 78
  - CSTURCV logical 78
  - CSTURCVSHR logical 78
  - CSUBGRN\_CONTINUE\_ON\_ER
    - ROR logical 279
  - CSVBASIC logical 78
  - CSVCOBOL logical 78
  - CSVDBAID logical 79
  - CSVFORTRA logical 79
  - CSVGLOBAL logical 79
  - CSVIPLVS logical 79
  - CSVIPLVS\_DEB logical 79
  - CSVLINK logical 86
  - CSXSCREEN logical 80
- D**
- data items
    - binding 242
  - data sets
    - accessing 227
    - controlling lists of 242
    - formatting 258
    - general information about 207
    - modifying 204
    - primary data sets 209
    - redundant 209
    - related data sets 212
    - sizes, estimating 196
  - database
    - compiling 194
    - disabling 151
    - displaying status of 154
    - dumping log of 158
    - enabling 160
    - migrating 243
    - prefix 51
    - printing 194
    - protection 141
    - shutting down 167

database (*cont.*)  
 specifying read-only access for  
 164  
 unloading 169  
 updating 171  
 validating 194  
 DATABASE-DESCRIPTIONS,  
 user name 197  
 DATA-DICTIONARY, user name  
 197  
 DBA  
 invoking 27  
 DBA utilities, to modify UDD files  
 206  
 DBAEDT logical 80  
 DBAID 24  
 invoking 27  
 DBAID\_HELP\_NOSPAWN  
 logical 279  
 DBVER logical 80  
 DEACTIVATE, an index 149  
 directory 17  
 buffers 202  
 changes and corresponding  
 action 202  
 changing database definition  
 198  
 modifying 200  
 structure 18  
 SUPRAD structure 195  
 DISABLE, an database 151  
 DISPLAY, a database 154  
 dump system log 158  
 DUMPSLF, PDM operator  
 command 158  
 DUMPSLF\_[xxx\_] dbname  
 logical 107  
 DYNLOCK, PDM input  
 parameters 127

**E**

ENABLING, a database 160  
 ENQUEUE\_LIMIT, setting 120  
 enrolling programs into the  
 directory 197  
 EXTENT, setting 121

**F**

fast utilities  
 description 194  
 using to alter directory data  
 sets 204  
 file density 211  
 FILE\_LIMIT, setting 121  
 format, functions 194  
 formatting, using CSTUIDX 194

**G**

global views, using 24, 229  
 groupwide PDM 34, 37, 38  
 GVSHEMA logical 107  
 GVSHEMA\_SYS logical 108

**H**

heterogeneous cluster and  
 network support 22  
 home address for a primary  
 record 210

**I**

IDXCNVERR, PDM input  
 parameters 128  
 IDXDUPERR, PDM input  
 parameters 128  
 IDXTIMEOUT, PDM input  
 parameters 128  
 index  
 activating 147  
 deactivating 149  
 file protection 144  
 populating 162  
 usage guidelines 230  
 initiating  
 SUPRA 27  
 the PDM 39, 48  
 INTERVAL, PDM input  
 parameters 129  
 IO\_BUFFERED, setting 122  
 IO\_DIRECT, setting 122

- L**
- LOGFLUSH, PDM input
    - parameters 129
  - logical view binding 24
- M**
- manual PDM initiation 40
  - MAXDATA, PDM input
    - parameters 130
  - MAXIMUM\_WORKING\_SET,
    - setting 122
  - MAXTASKS, PDM input
    - parameters 131
  - MAXTHREADS, PDM input
    - parameters 132
  - modifying
    - data sets 204
    - SUPRA directory database 200
    - SUPRAD files
      - using DBA utilities 206
      - using fast utilities 204
  - MRELAY, PDM input parameters
    - 133
  - MULTIHOLD, PDM input
    - parameters 133
  - multiple sytemwide PDM 36, 38
- N**
- network
    - checking machines for active
      - PDM 48
    - initiating the PDM on 48
    - PDM access 22
- O**
- OPERATOR, PDM input
    - parameters 134
- P**
- PAGE\_FILE, setting 123
  - PDM input parameters
    - ACLCHECK 126
    - CONSOLE 126
    - DYNSLOCK 127
    - entering 125
    - IDXCNVERR 128
    - IDXDUPERR 128
    - IDXTIMEOUT 128
    - INTERVAL 129
    - LOGFLUSH 129
    - MAXDATA 130
    - MAXTASKS 131
    - MAXTHREADS 132
    - MRELAY 133
    - MULTIHOLD 133
    - OPERATOR 134
    - PDMNAME 134
    - PRIORITY 135
    - RETRY 136
    - SINGLEUNLOAD 136
    - STATISTICS 137
    - SYSTOPCOM 138
    - TIMEOUT 138
    - UICCHECK 139
  - PDM operator commands
    - ACTIVATE 147
    - DEACTIVATE 149
    - DISABLE 151
    - DISPLAY 154
    - DUMPSLF 158
    - ENABLE 160
    - POPULATE 162
    - READONLY 164
    - SHUTDOWN 167
    - UNLOAD 169
    - UPDATE 171
  - PDMNAME, PDM input
    - parameters 134
  - Physical Data Manager (PDM)
    - automatic initiation 42
    - check for active PDM 48
    - configuring 33
    - CSI\_AUTOSTART 42
    - CSI\_FINDPDM 48
    - CSI\_PDMID 92
    - CSIPDMINP 92
    - CSIPDMLOG 93
    - CSISTRINP 103
    - CSISTRLOG 93
    - groupwide 34, 37, 38
    - initializing 39
    - manual initiation 40
    - multiple systemwide 36, 38
    - statistics 269
    - systemwide 35, 37
    - user exits 259
  - POPULATE, an index 162
  - populating 194

prefix, using 51  
 printing read-ahead buffer  
   statistics 240  
 PRIORITY, PDM input  
   parameters 135  
 PUBLIC user name, for enrolling  
   programs 197

## Q

QUEUE\_LIMIT, setting 123

## R

read-ahead buffering  
   context position considerations  
     239  
   program considerations 239  
   program exceptions 239  
   remote application  
     considerations 241  
   tuning 241  
   turning off 240  
   turning on 240  
 read-ahead buffering,  
   understanding 238  
 READONLY, PDM operator  
   command 164  
 recovery point, creating for  
   database 199  
 Relational Data Manipulation  
   Language (RDML) 21  
 RETRY, PDM input parameters  
   136  
 RUNBASIC logical 80  
 RUNCOBOL logical 81  
 RUNCSV logical 86  
 RUNDBAID logical 81  
 RUNDIRM logical 81  
 RUNFORTRA logical 81

## S

shadow recovery 22  
 SHUTDOWN, PDM operator  
   command 167  
 SINGLEUNLOAD, PDM input  
   parameters 136  
 SPECTRA 29  
 statistics, for the PDM 269

STATISTICS, PDM input  
   parameters 137  
 SUBPROCESS\_LIMIT, setting  
   123  
 SUPRA  
   administration utilities 25  
   cluster support 22  
   components 20  
   facilities menu 27  
   images, privileges required for  
     60  
   logical 81  
   network support 22  
   related products 29  
 SUPRA directory database  
   (SUPRAD)  
   data set sizes, estimating 196  
   definiton of, changing 198  
   general information about 195  
   modifying 200  
   modifying data sets 204  
   recovery point, creating 199  
   structure of 195  
   user names, setting up 197  
 SUPRA\_AUXIL logical 82  
 SUPRA\_BASE logical 82  
 SUPRA\_BURRYS logical 82  
 SUPRA\_CLEAN\_DICT logical 82  
 SUPRA\_CLEAN\_EXE logical 82  
 SUPRA\_COMS logical 83  
 SUPRA\_DICT logical 83  
 SUPRA\_EXAMPLES logical 83  
 SUPRA\_EXE logical 83  
 SUPRA\_PATCH\_WORK logical  
   84  
 SUPRA\_PDM\_PATCHES logical  
   84  
 SUPRA\_REPORT logical 85  
 SUPRA\_TEST\_EXE logical 85  
 SUPRA-HELP logical 84  
 SUPRAPDM logical 84  
 SYS\$ULTRA logical 85  
 SYSOPCOM, PDM input  
   parameters 138  
 system level recovery 22  
 systemwide PDM 35, 37

## T

task level recovery 22  
 TIMEOUT, PDM input  
   parameters 138

## U

- UDD files 195
  - modifying 204
- UICCHECK, PDM input
  - parameters 139
- ULTRADBMS logical 86
- ULTRAPDM logical 85
- UNLOAD, PDM operator
  - command 169
- UPDATE, PDM operator
  - command 171
- user exits
  - COBOL examples 259
  - FORTTRAN example 267
  - writing 55

- user names
  - DATABASE-DESCRIPTIONS 197
  - DATA-DICTIONARY 197
  - PUBLIC 197
  - setting up 197
  - UTILITIES 197, 201
- UTILITIES user name 197, 201

## V

- view binding 24
- views
  - global 24

## W

- WORKING\_SET, setting 124