

Cincom

SUPRA SERVER PDM

Tuning Guide
(OS/390 & VSE)

P26-0225-63



SUPRA[®] Server PDM Tuning Guide (OS/390 & VSE)

Publication Number P26-0225-63

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

This document contains unpublished, confidential, and proprietary information of Cincom.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

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

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

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

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

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

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

Attention:

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

Release information for this manual

The *SUPRA Server PDM Tuning Guide (OS/390 & VSE)*, P26-0225-63, is dated January 15, 2002. This document supports Release 2.7 of SUPRA Server PDM in IBM mainframe environments.

We welcome your comments

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

Cincom Technical Support for SUPRA Server PDM

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

E-mail: helpna@cincom.com

Phone: 1-800-727-3525

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



Contents

About this book	ix
Using this document.....	ix
Document organization	x
Revisions to this manual	x
Conventions	xi
SUPRA Server documentation series	xiv
General tuning guidelines	17
Setting tuning goals.....	18
Planning the tuning effort	21
Obtaining statistics	23
Obtaining PDM statistics	23
Obtaining statistics from interactive services	25
Obtaining statistics from PDM DML commands	28
Obtaining statistics from the Execution Statistics utility	29
Obtaining statistics from the File Statistics function.....	33
Obtaining statistics from the utilities Log Print function.....	44
Obtaining RDM statistics.....	46
Obtaining CICS statistics	47
Obtaining CICS connector statistics.....	47
Obtaining CICS-supplied statistics.....	48

Tuning the PDM	49
Tuning buffer pools	50
Calculating the buffer count value	50
Allocating buffer pools	51
Tuning the buffers.....	51
Tuning block size	54
Defining block sizes	54
Optimizing block size	56
Tuning CSIPARM parameters	57
Using system logging options	59
Calculating file capacity	61
Placing files on devices	63
Avoiding record contention	63
Tuning the number of concurrent PDM tasks	63
Using the file experience table to reduce I/O	64
Reorganizing files	65
Using PDM memory.....	66
Determining domain and dispatch priorities for central PDM	67
Using interval service (ISV) in attached mode.....	70
Optimizing your database design.....	71
Handling concurrent application updates.....	72
Coding commits in your application program	73
Optimizing user exits	73
Defining dynamic indexes.....	74
Tuning the optional PDM Extended Storage Support.....	75
Tuning the optional Buffer Cache Facility.....	77
 Tuning the Directory	 79
Using multiple load modules.....	79
Placing directory files on disk	80
Processing a Directory.....	80

Tuning the CICS connector	81
Tuning OPER CONNECT parameters.....	82
THREADS parameter.....	83
TASKS parameter.....	84
MEME parameter.....	85
Retry parameters.....	86
Turning off tracing and Sync for performance.....	87
Tuning RDM	89
Installing RDM.....	89
Defining tasks.....	90
Defining task and global view storage.....	93
Defining efficient views.....	96
Defining data and navigation.....	96
Determining view size.....	98
Making views bound.....	99
Making views global.....	100
Tuning DBAID and applications.....	102
Subsetting views.....	104
Using views.....	105
Using the online DBAID slot (RPTSIZE).....	106

Memory use and space calculations	107
MEMORY parameter calculation for the PDM	107
PDM system memory requirements	108
File memory requirements	109
Buffer memory requirements	110
User-variable memory requirements	111
Notes on the preceding tables	112
Sample PDM memory requirements	115
PDM memory requirements—small sites	116
PDM memory requirements—medium sites	120
PDM memory requirements—large sites	124
Sample PDM memory requirements notes	128
Memory requirements for VSAM PDM data sets	132
Buffer-size calculation for KSDS	132
VSAM control-block allocation	133
Minimum storage requirements for PDM VSAM files	137
Minimum size of PDM with VSAM files	138
Memory requirements for Directory Maintenance	139
Batch configurations	140
CICS configurations	141
Memory requirements for the CICS Connector	142
Resident Program requirements for SUPRA Server in a CICS environment	143
Performance considerations for VSE XPCC	144
XPCC memory usage	146
XPCC CPU usage	146
Application memory usage with XPCC	147
Index disk space calculations	147
Calculating index space	148
Sample program for index space calculation	152
 Index	 161

About this book

Using this document

This document is directed to personnel who manage the ongoing maintenance of SUPRA[®] Server for performance and response time tuning.

This manual is a renamed and renumbered combination of two former manuals for SUPRA Server 1.x: the *TIS/XA Tuning Guidelines (MVS)*, P26-0479, and the *TIS/XA Tuning Guidelines (VSE)*, P26-0489. These manuals are now obsolete.

The main changes to the Tuning Guide from previous versions are additions of information for new features. These are the RDM Extended Storage feature for OS/390 and VSE (ESA) and the optional PDM Extended Storage Support and Buffer Cache Facility for OS/390 and VSE (ESA). The PDM interfaces can now be resident in extended storage, which affects how the system is tuned. This version also discusses new parameters for the RDM C\$VOOTPM macro and the CICS Connector CSTXOPRM macro. Additionally, there are modifications to the sample space calculations in “[Memory use and space calculations](#),” starting on page 107. This manual does not contain tuning information for the SQL and HDMP support in SUPRA Server 2.4.

The MEMORY parameter calculations that previously appeared in the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250, are now only in the [appendix](#) of this manual.

Document organization

The information in this manual is organized as follows:

Chapter 1—General tuning guidelines

Describes how to set guidelines for performance tuning to achieve higher throughput and improve response time.

Chapter 2—Obtaining statistics

Describes how to obtain statistics under PDM, RDM, and CICS.

Chapter 3—Tuning the PDM

Describes how to improve performance by tuning system and application factors related to the Physical Data Manager (PDM).

Chapter 4—Tuning the Directory

Describes how to improve performance for the Directory.

Chapter 5—Tuning the CICS Connector

Describes how to improve performance for CICS SUPRA.

Chapter 6—Tuning RDM

Describes how to improve RDM performance.

Appendix—Memory use and space calculations

Provides memory requirements for SUPRA Server components and shows the calculations used to determine those requirements.

Index

Revisions to this manual

The chapter on tuning NORMAL has been removed from this document, as NORMAL is no longer shipped. Please refer to previous versions of SUPRA documentation for information on NORMAL.

Conventions

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

Convention	Description	Example
Constant width type	Represents screen images and segments of code.	<pre>PUT 'customer.dat' GET 'miller\customer.dat' PUT '\DEV\RMT0'</pre>
Slashed b (<i>b</i>)	Indicates a space (blank). The example indicates that 4 spaces appear between the keywords.	<pre>BEGIN 4 SERIAL</pre>
Brackets []	Indicate optional selection of parameters. (Do not attempt to enter brackets or to stack parameters.) Brackets indicate one of the following situations:	
	A single item enclosed by brackets indicates that the item is optional and can be omitted.	[WHERE <i>search-condition</i>]
	The example indicates that you can optionally enter a WHERE clause.	
	Stacked items enclosed by brackets represent optional alternatives, one of which can be selected.	<pre>[<u>WAIT</u> NOWAIT]</pre>
	The example indicates that you can optionally enter either WAIT or NOWAIT. (WAIT is underlined to signify that it is the default.)	

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

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

SUPRA Server documentation series

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

Overview

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

Getting started

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

General use

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

Database administration tasks

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

Application programming tasks

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

Report tasks

- ◆ *SPECTRA User's Guide*, P26-9561



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



Educational material is available from your regional Cincom education department.

1

General tuning guidelines

Guidelines help you set specific tuning goals and plan your tuning efforts. The objectives of performance tuning are to achieve higher throughput and to improve response time. Throughput is the amount of work a computer system can do per unit of time. This relates to response time reduction and its impact on personnel productivity as well as overall batch throughput. Several factors affect throughput and response time:

- ◆ Physical Input/Output (I/O) processing, such as DASD, tape, terminal
- ◆ Central Processing Unit (CPU) usage
- ◆ Memory usage
- ◆ Buffer count and size
- ◆ DASD configuration

Setting tuning goals

Different tuning methods alter the various factors in different ways. For example, additional buffers can improve I/O speed, but sacrifice memory consumption and CPU usage. If the physical resources (real memory, I/O channels, CPU and DASD space) are not adequate to support the concurrency (threads and tasks) requested, performance degrades.

It is important to weigh available physical resources against the needed concurrency. You want the maximum concurrency you can have without having throughput decrease unacceptably or response time increase unacceptably. Therefore, before attempting to tune your software, you should set specific tuning goals. Some possible software tuning goals are:

- ◆ Reducing response time to improve productivity
- ◆ Decreasing CPU time to decrease operational costs
- ◆ Balancing memory consumption to maximize system throughput
- ◆ Balancing physical I/O to avoid contention and degradation

The following table lists factors that can affect response time, CPU time, memory consumption, physical I/O and channel use. Make appropriate modifications based on the results of reviewing these factors. Use “[Tuning the PDM](#)” on page 49, “[Tuning the Directory](#)” on page 79, “[Tuning the CICS connector](#)” on page 81, and , “[Tuning RDM](#)” on page 89 for specific tuning of each of the factors.

Impact	Considerations
Response time	<p>Use global views and bound views when possible.</p> <p>Review and optimize view design.</p> <p>Release a view as soon as a task has no more use for it. Avoid opening a view multiple times in one task or pseudoconversation.</p> <p>Review RDM heap rolling. Set the number of heaps (HEAP#) as close as possible to the number of concurrent RDM tasks (RDMUSR#) without exceeding it.</p> <p>Make use of subset view opens where possible.</p> <p>Review data set placement (high activity versus low activity).</p> <p>Review buffering and block sizes in the Directory, PDM files, and user files.</p> <p>Review Directory and user file activity (I/O).</p> <p>Review file structure and the amount of free space available in database files.</p> <p>Review application design and code.</p> <p>Review CICS file and transaction statistics.</p> <p>Review degree of task/thread concurrency in CICS and the PDM.</p> <p>Review operating mode (central, attached, cross address space).</p>
CPU time	<p>Check the number of PDM threads, PDM execution priority, and the number of CICS Connector threads. In general, applications must have easy access to resources when they are needed.</p> <p>Review application design and code.</p>
Memory consumption, fixed	<p>You can place some executable code in a separate address space or above the 16 MB line. You can downsize Directory Maintenance into multiple executable modules. Code can be shared, such as SPECTRA or RDM code, by placing it in the shared area.</p>

Impact	Considerations
Memory consumption, variable	<p>Memory requirements vary with network size, degree of concurrency desired, number and size of files, etc.</p> <p>Use storage above the 16 MB line where possible.</p> <p>Review MANTIS global programs.</p> <p>Review RDM global views and number and size of RDM heaps.</p> <p>Review execution time parameters in CICS.</p> <p>Review PDM buffering, blocking, concurrency and transaction size.</p> <p>Consider central mode for the PDM.</p> <p>For severe memory constraints, consider using multiple CICS systems with a central PDM for nonproduction components like DBAID, Directory Maintenance, MANTIS development and SPECTRA development.</p>
Physical I/O, channel use	<p>Review data set placement (high versus low activity) and channel usage.</p> <p>Review operating system data set placement and channel usage (swap and paging data sets).</p> <p>Where Execution Statistics show considerable wait time, consider splitting the high-use data sets into multiple extents assigned to different physical devices.</p> <p>Review buffering and block sizes of the Directory files, PDM files, and user files.</p> <p>Review the system logging options used.</p> <p>Review your application design and code.</p> <p>Consider using the file experience table of the PDM.</p> <p>Review the sizes of logical units of work.</p> <p>Review PDM concurrency and transaction concurrency.</p> <p>Review the use of alternate paths.</p> <p>Use secondary key access where requested data is wholly contained in the secondary key.</p> <p>Review MANTIS clusters, binding and global programs.</p> <p>Review CICS Temporary Storage.</p> <p>Review RDM heap rolling. Set the number of heaps (HEAP#) as close as possible to the number of concurrent RDM tasks (RDMUSR#) without exceeding it.</p>

Planning the tuning effort

Tuning is a continuing process involving overall system performance and application efficiency. Consider the following factors when planning your tuning efforts.

- ◆ To effectively tune your system, you must know where computer time is spent. Therefore, you must gather statistics and maintain performance records. Use this information before and after each tuning effort to compare the differences and measure the effect. See “[Obtaining statistics](#)” on page 23 for information on obtaining statistics.

Statistics are available from the following components:

- Physical Data Manager (PDM)
- Relational Data Manager (RDM)
- CICS Connector

Additional statistics are available from the following:

- OS/390 systems
- CICS
- Other monitor systems

You can use statistics to break a logical unit of work into its components, as shown below, and to analyze the percent of time required for each. Make your initial improvements where the most time is spent. For best performance, keep time-in-memory short for each transaction.

- I/O and run time (system function)
- Queuing time (system function)
- Processing time (application function)

- ◆ Each component must have the required resources to accomplish its tasks. However, tuning does not always mean allocating more resources. In some cases, you can achieve greater efficiency by allocating fewer resources. For example, too much concurrency may cause lower system throughput.
- ◆ Maintain a log to keep a record of any changes to your system or environment. Consider these changes when measuring tuning effectiveness.
- ◆ Performance also depends upon how efficiently you design and write your application programs. For example, you can affect performance by:
 - Using bound global views where possible and eliminating any redundant views
 - Algorithm selection
 - Your database design

2

Obtaining statistics

Statistics provide information about your system that you can use to monitor performance. You should use statistics periodically to check for performance changes and to monitor how your tuning efforts affect SUPRA. You can obtain statistics for the PDM, the RDM, and the CICS Connector.

Obtaining PDM statistics

The executing PDM provides both file-level and system-level statistics. Some are informational items from CSIPARM or Directory information, and some are accumulated run-time statistics. Two utilities provide other file-level statistics by scanning a file. These various statistics are available through the following sources:

- ◆ Interactive Services (see “[Obtaining statistics from interactive services](#)” on page 25)
- ◆ PDM DML commands SHOWX and RSTAT (see “[Obtaining statistics from PDM DML commands](#)” on page 28)
- ◆ Execution Statistics utility (see “[Obtaining statistics from the Execution Statistics utility](#)” on page 29)
- ◆ File Statistics function, DBA Utilities (see “[Obtaining statistics from the File Statistics function](#)” on page 33)
- ◆ Log Print function (System Log file), DBA Utilities (see “[Obtaining statistics from the utilities Log Print function](#)” on page 44)

The first three previous sources are interrelated as follows:

Source	Run-time accumulations	Extensive physical & logical information
Interactive Services display (uses an internal SHOWX)	x	x
PDM DML commands:		
SHOWX command (to data area)	x	x
RSTAT command (to STAT file)	x	-
Execution Statistics printed from STAT file after PDM shutdown	x	-

The physical/logical information sources are available at all times to Interactive Services and to a SHOWX DML command. However, the run-time accumulations are available only if the environment description specifies Yes in the Statistics Indicator parameter (statistics turned on).

To turn on statistics gathering, the PDM requires a Statistics Log File and buffer pool defined in the schema, related to the current environment description, and present in the PDM JCL. The Statistics Log File is a sequential output file that can be on disk or tape. It can be a dummy file if you do not plan to use RSTAT WRITE or the Execution Statistics utility. If it is a dummy file, you can still use Interactive Services or a SHOWX DML command.

The PDM opens the Statistics File at initialization and closes it at termination. When the Statistics Indicator is Yes, the PDM automatically writes accumulated values to the file when initialization is complete, and resets the counters to zero. You can cause the PDM to write or clear the accumulations at any time during execution by using the RSTAT DML command in an application (see “[Obtaining statistics from PDM DML commands](#)” on page 28). Then at PDM termination, the PDM automatically writes accumulated values since the most recent reset (since initialization if no RSTAT CLEAR was issued).

To activate statistics accumulation, use Directory Maintenance and enter Y for the Statistics Indicator in your environment description. Refer to the [SUPRA Server PDM Directory Online User's Guide \(OS/390 & VSE\)](#), P26-1260, or the [SUPRA Server PDM Directory Batch User's Guide \(OS/390 & VSE\)](#), P26-1261, for information on defining the Statistics File and maintaining the environment description parameters. For additional instructions on defining the Statistics File, refer to the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250.

Obtaining statistics from interactive services

Use Interactive Services to obtain online system and file statistics from the PDM. First, select Interactive Services from the Software Selection Facility menu. The Interactive Services Main Menu appears as shown in the following:

```
INTERACTIVE SERVICES MAIN MENU                                300

1. PHYSICAL FILE SERVICES
2. PDM SERVICES
3. USER EXTENSION SERVICES

ENTER OPTION NUMBER :      :
```

PA2-EXIT

Options 1 and 2 are explained below. Option 3, User Extension Services, is not a statistics option. It enables you to add your own utility programs to the Interactive Services Main Menu.

- ◆ Select option 1, Physical File Services, to display the Physical File Services menu (not shown). Select the File Statistics option; then enter the name of one file. The screen displays the first page of the statistics for the named file. These are accumulations for this file since initialization or since an RSTAT CLEAR issued by any application.

FILE STATISTICS		510
FOR FILE C\$-#		
F1.01	TOTAL LOGICAL READS	22
F1.02	TOTAL PHYSICAL READS	3
F1.03	TOTAL IN-MEMORY HITS	19
F1.04	TOTAL IN-MEMORY HITS ON UPDATED BUFFER	6
F1.05	TOTAL PHYSICAL UPDATES FORCED BY PHYSICAL READ.	0
F2.01	AVG. LOGICAL READS	7.33
F2.02	% OF LOGICAL READS WHICH WERE IN-MEMORY HITS .	86.36%
F2.03	% OF IN-MEMORY HITS TO AN UPDATED BUFFER	31.58%
F2.04	% OF PHYSICAL READS FORCING A PHYSICAL UPDATE .	0.00%
PA2-EXIT		-MORE-
WMM:PF7-UP	PF8-DOWN	PF9-END PF10-LEFT PF11-RIGHT PF12-HOME
		001 001

To display the remaining statistics for this file, you page down the screen. They are the same statistics items you receive with the Execution Statistics utility (see the file report page in “[Obtaining statistics from the Execution Statistics utility](#)” on page 29 for the complete list).

The cumulative statistics are not meaningful for this file unless the environment description Statistics Indicator is Yes (see “[Obtaining PDM statistics](#)” on page 23).

- ◆ Select option 2, PDM Services, to display the PDM Services menu (not shown). Select the PDM System Statistics option. The screen displays the first page of the system statistics. These are accumulations since initialization or since an RSTAT CLEAR issued by any application.

PDM STATISTICS FOR name		610	
S1.01	DATE AND TIME STATISTICS WERE LAST RESET ..	9/3/92	14:36:59
	CURRENT TASKS	2	
S2.01	TOTAL TASKS	4	
S2.02	MAXIMUM CONCURRENT TASKS	2	
S3.01	CURRENT RECORD HOLDING ENTRIES IN USE	0	
S3.02	MAXIMUM RECORD HOLDING ENTRIES USED	0	
S3.03	CURRENT MONITOR ENTRIES IN USE	0	
S3.04	MAXIMUM MONITOR ENTRIES USED	0	
S4.01	TOTAL READ COMMANDS	11	57.89%
S4.02	TOTAL UPDATE COMMANDS	0	0.00%
S4.03	TOTAL ADD AND DELETE COMMANDS	0	0.00%
S4.04	TOTAL OTHER COMMANDS	8	42.11%
PA2-EXIT		-MORE-	
WMM:PF7-UP PF8-DOWN PF9-END PF10-LEFT PF11-RIGHT PF12-HOME		001 001	

To display the remaining system statistics, you page down the screen. They are the same statistics items you receive with the Execution Statistics utility (see the system report page in [“Obtaining statistics from the Execution Statistics utility”](#) on page 29 for the complete list). Some system and file statistics have changed title and/or meaning from earlier SUPRA Server releases.

The cumulative statistics for the system are not meaningful unless the environment description Statistics Indicator is Yes (see [“Obtaining PDM statistics”](#) on page 23).

The other features and options of Interactive Services do not concern statistics. For detailed instructions on using Interactive Services, refer to the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250.

Obtaining statistics from PDM DML commands

You can use two PDM DML commands (RSTAT and SHOWX) in an application program to obtain PDM statistics while the system is running.

- ◆ RSTAT WRITE writes accumulated statistics to the PDM Statistics File. You can code RSTAT WRITE with the option to write only system statistics, only file statistics, named files or all files. RSTAT defaults to writing only system statistics (no file statistics). RSTAT WRITE does not clear/reset the accumulators. You can code RSTAT CLEAR to reset the PDM statistics counters to zeros without writing to the file. You could use variations of the options to monitor the system, a certain file, a certain stand-alone application, and so on.

After shutting down the PDM, use the Execution Statistics utility (see [“Obtaining statistics from the Execution Statistics utility”](#) on page 29) to report the statistics RSTAT writes to the file. You can use the PDM Interactive Services (see [“Obtaining statistics from interactive services”](#) on page 25) to see current statistics periodically while still running.

- ◆ SHOWX can retrieve any of the RSTAT statistics into the program’s data area using data item keywords. In addition, SHOWX can retrieve a multitude of Directory and internal PDM physical and logical information. You can retrieve all or part of the statistical data or the other available information. The program can then format and display or print the results.

See [“Obtaining PDM statistics”](#) on page 23 for the general discussion of PDM statistics. Refer to the [SUPRA Server PDM DML Programming Guide \(OS/390 & VSE\)](#), P26-4340, for detailed information on using the RSTAT and SHOWX DML commands. The RSTAT description includes layout illustrations of the records on the Statistics File. See [“Obtaining statistics from interactive services”](#) on page 25 and [“Obtaining statistics from the Execution Statistics utility”](#) on page 29 for example statistics reporting.

Obtaining statistics from the Execution Statistics utility

The Execution Statistics utility (CSUXSTAT) generates formatted reports of the counters contained in the Statistics File of a terminated PDM.. This section presents samples of the statistics reports. (For instructions on executing the utility, refer to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260.)

The utility provides groups of statistics according to the records on the file. At initialization, the PDM writes a group of records to the Statistics File then resets the counters. A group consists of a system record and file records. The PDM then keeps accumulating statistics, finally writing a group of records at termination.

In addition, any task can use the RSTAT PDM DML command (see “[Obtaining statistics from PDM DML commands](#)” on page 28) to write or reset accumulated statistics between initialization and termination. You write a group of records by issuing an RSTAT WRITE at any time. RSTAT WRITE does not clear/reset the accumulators. You can reset the counters by issuing an RSTAT CLEAR at any time.

The termination group of records contains all accumulated counters since initialization or since the most recent RSTAT CLEAR issued by any application.

When the Execution Statistics utility processes the file from the PDM run, the report contains at least two groups, initialization and termination. In between those two, it contains a group for each RSTAT WRITE that occurred during that PDM execution.

Each group report shows a single group of statistics that were placed in the Statistics File at the same time. A group report contains the following pages:

- ◆ Group identification page
- ◆ System statistics page
- ◆ Set of file statistics pages (one page for each file defined to the PDM)
- ◆ File statistics totals page

The following listing shows the system statistics page of a group report.



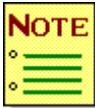
These PDM system-level statistics are cumulative for all tasks from all interfaces. They begin when the statistics were last reset by PDM initialization or by an RSTAT CLEAR from any task. They end when the group was written to the Statistics File by an RSTAT WRITE from any task or by PDM shutdown. They are not specific to any task or interface unless it was the only one active during this group.



The title and/or meaning of some system and file statistics have changed from earlier SUPRA Server releases. Refer to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260, for specifics.

FUNCTION = EXECUTION STATISTICS			
GROUP 2.1			
SYSTEM STATISTICS			
S1.01	DATE AND TIME STATISTICS WERE LAST RESET	JUN. 17, 1992	15:50:56
S1.02	FIRST GROUP OF EXECUTION STATISTICS SINCE RESET?	YES	
S2.01	TOTAL TASKS		2
S2.02	MAXIMUM CONCURRENT TASKS		2
S3.01	CURRENT RECORD HOLDING ENTRIES IN USE		0
S3.02	MAXIMUM RECORD HOLDING ENTRIES USED		0
S3.03	CURRENT MONITOR ENTRIES(READ LOCKS) IN USE		0
S3.04	MAXIMUM MONITOR ENTRIES(READ LOCKS) USED		0
S4.01	TOTAL READ COMMANDS		40 71.43%
S4.02	TOTAL UPDATE COMMANDS		0 0.00%
S4.03	TOTAL ADD AND DELETE COMMANDS		0 0.00%
S4.04	TOTAL OTHER COMMANDS		14 25.00%
S5.01	TOTAL COMMANDS ISSUED TO THE PDM		56
S5.02	MAXIMUM NUMBER OF COMMANDS AT COMMAND STARTS		0
S5.03	SUM OF COMMANDS AT COMMAND STARTS		0
S5.04	AVERAGE NUMBER OF COMMANDS AT COMMAND STARTS (S5.03/S5.01)		0.00AVG.
S6.01	TOTAL ELAPSED TIME ON COMMANDS ISSUED TO THE PDM		00:00:07.903
S6.02	AVERAGE ELAPSED TIME PER COMMAND ISSUED TO THE PDM (S6.01/S5.01)		00:00:00.141
S6.03	MAXIMUM ELAPSED TIME FOR ANY COMMAND ISSUED TO THE PDM		00:00:00.829
S7.01	NUMBER OF TIMES PDM WAS INACTIVE		112
S7.02	AVERAGE NUMBER OF TIMES PDM WAS INACTIVE PER COMMAND (S7.01/S5.01)		2.00AVG.
S8.01	AMOUNT OF TIME PDM WAS ACTIVE (HH:MM:SS.SS)	00:00:04.281	41.12%
S8.02	AMOUNT OF TIME PDM WAS INACTIVE	00:00:06.131	58.88%
S8.03	TOTAL PDM TIME (S8.01+S8.02)	00:00:10.412	100.00%
S8.04	AVERAGE AMOUNT OF TIME PDM WAS ACTIVE PER COMMAND (S8.01/S5.01)	00:00:00.076	
S9.01	TOTAL BYTES OF MEMORY (IN K)		1,460
S9.02	TOTAL BYTES OF MEMORY USED AT PRESENT TIME (IN K)		528 36.16%
S9.03	MAXIMUM BYTES OF MEMORY USED (IN K)		532 36.44%
S9.04	THRESHOLD MEMORY		90%
S10.01	TOTAL BYTES OF XA MEMORY (IN K)		6,144
S10.02	TOTAL BYTES OF XA MEMORY USED AT PRESENT TIME (IN K)		148 2.41%
S10.03	MAXIMUM BYTES OF XA MEMORY USED (IN K)		148 2.41%

The following listing shows the next page of the group report (the first file statistics page).



One page prints for each file defined to the PDM when the group was written to the Statistics File. (This includes all defined database, Directory, log, and statistics files.)

The statistics describe the activity for the file from the time the statistics were last reset until the group was written to the Statistics File by an RSTAT WRITE from any task or by a PDM termination. These statistics are at the PDM file level and, therefore, are not specific to a task unless it was the only task using that file during this group.

FUNCTION = EXECUTION STATISTICS			
FILE STATISTICS			GROUP 7.9
FOR FILE M001			
FILE TYPE	PRIMARY	LOGICAL RECORD LENGTH	47
FILE CODED	NO	BLOCK SIZE	9,400
FILE DDNAME	M001	BLOCKS PER TRACK	4
ACCESS METHOD	BDAM	RECORDS PER BLOCK	200
BUFFER POOL	USRM	TOTAL LOGICAL RECORDS	1,200
		CONTROL INTERVAL SIZE	0
F1.01	TOTAL LOGICAL READS		0
F1.02	TOTAL PHYSICAL READS		0
F1.03	TOTAL IN-MEMORY HITS	(F1.01-F1.02)	0
F1.04	TOTAL IN-MEMORY HITS ON UPDATED BUFFER		0
F1.05	TOTAL PHYSICAL UPDATES FORCED BY A PHYSICAL READ		0
F2.01	AVERAGE LOGICAL READS PER PHYSICAL READ	(F1.01/F1.02)	>>>>>.>>AVG.
F2.02	% OF LOGICAL READS WHICH WERE IN-MEMORY HITS	((F1.03/F1.01)*100)	>>>>>.>>%
F2.03	% OF IN-MEMORY HITS WHICH WERE TO AN UPDATED BUFFER	((F1.04/F1.03)*100)	>>>>>.>>%
F2.04	% OF PHYSICAL READS FORCING A PHYSICAL UPDATE	((F1.05/F1.02)*100)	>>>>>.>>%
F3.01	TOTAL LOGICAL UPDATES		0
F3.02	TOTAL PHYSICAL UPDATES		0
F3.03	TOTAL MULTIPLE LOGICAL UPDATES TO THE SAME BUFFER		0
F3.04	AVERAGE LOGICAL UPDATES PER PHYSICAL UPDATE	(F3.01/F3.02)	>>>>>.>>AVG.
F3.05	% OF PHYSICAL UPDATES WHICH WERE MULTIPLE UPDATES	((F3.03/F3.02)*100)	>>>>>.>>%
F4.01	TOTAL LOGICAL I/O	(F1.01-F3.01)	0
F4.02	TOTAL PHYSICAL I/O	(F1.02-F3.02)	0
F4.03	AVERAGE LOGICAL I/O PER PHYSICAL I/O	(F4.01/F4.02)	>>>>>.>>AVG.
F5.01	% OF LOGICAL I/O WHICH WERE LOGICAL READS	((F1.01/F4.01)*100)	>>>>>.>>%
F5.02	% OF LOGICAL I/O WHICH WERE LOGICAL UPDATES	((F3.01/F4.01)*100)	>>>>>.>>%
F5.03	% OF PHYSICAL I/O WHICH WERE PHYSICAL READS	((F1.02/F4.02)*100)	>>>>>.>>%
F5.04	% OF PHYSICAL I/O WHICH WERE PHYSICAL UPDATES	((F3.02/F4.02)*100)	>>>>>.>>%
F6.01	TOTAL PHYSICAL UPDATE WAITS DUE TO LOG FILE UPDATES		0
F7.01	TOTAL STOLEN LOCKED RECORDS		0
F7.02	TOTAL SUCCESSFUL LOCKS WITHOUT WAITING		0
F7.03	TOTAL SUCCESSFUL LOCKS AFTER WAITING		0
F7.04	TOTAL SUCCESSFUL LOCKS	(F7.01+F7.02+F7.03)	0
F7.05	TOTAL FAILED LOCKS WITHOUT WAITING		0
F7.06	TOTAL FAILED LOCKS AFTER WAITING		0
F7.07	TOTAL EMBRACES DETECTED		0
F7.08	TOTAL FAILED LOCKS	(F7.05+F7.06+F7.07)	0
F8.01	TOTAL LOCKS REQUESTED	(F7.04+F7.08)	0
F8.02	% OF TOTAL LOCKS WHICH WERE SUCCESSFUL	((F7.04/F8.01)*100)	>>>>>.>>AVG.
F8.03	% OF TOTAL LOCKS WHICH FAILED	((F7.08/F8.01)*100)	>>>>>.>>%

After the individual file statistics pages, a totals page for file statistics appears for each group report. The following listing shows a totals page.



Except for items T1.00 and T6.02-T6.05, these statistics sum the corresponding statistics on the individual file statistics pages. They include the Directory files, log files, and the statistics file, in addition to the database files.

These statistics add PDM activity for all files from the time the statistics were last reset until the statistics record group was written to the Statistics File by an RSTAT WRITE from any task or by PDM termination. They are not specific to a task that issues an RSTAT command.

FUNCTION = EXECUTION STATISTICS			
FILE STATISTICS		GROUP 7.9	
TOTALS FOR GROUP 7			
T1.00	NUMBER OF FILES FOR WHICH STATISTICS WERE ACCUMULATED		18
T1.01	TOTAL LOGICAL READS		1,197
T1.02	TOTAL PHYSICAL READS		32
T1.03	TOTAL IN-MEMORY HITS	(T1.01-T1.02)	1,165
T1.04	TOTAL IN-MEMORY HITS ON UPDATED BUFFER		591
T1.05	TOTAL PHYSICAL UPDATES FORCED BY A PHYSICAL READ		0
T2.01	AVERAGE LOGICAL READS PER PHYSICAL READ	(T1.01/T1.02)	37.41AVG.
T2.02	% OF LOGICAL READS WHICH WERE IN-MEMORY HITS	((T1.03/T1.01)*100)	97.33%
T2.03	% OF IN-MEMORY HITS WHICH WERE TO AN UPDATED BUFFER	((T1.04/T1.03)*100)	50.73%
T2.04	% OF PHYSICAL READS FORCING A PHYSICAL UPDATE	((T1.05/T1.02)*100)	0.00%
T3.01	TOTAL LOGICAL UPDATES		604
T3.02	TOTAL PHYSICAL UPDATES		353
T3.03	TOTAL MULTIPLE LOGICAL UPDATES TO THE SAME BUFFER		0
T3.04	AVERAGE LOGICAL UPDATES PER PHYSICAL UPDATE	(T3.01/T3.02)	1.71AVG.
T3.05	% OF PHYSICAL UPDATES WHICH WERE MULTIPLE UPDATES	((T3.03/T3.02)*100)	0.00%
T4.01	TOTAL LOGICAL I/O	(T1.01-T3.01)	1,801
T4.02	TOTAL PHYSICAL I/O	(T1.02-T3.02)	385
T4.03	AVERAGE LOGICAL I/O PER PHYSICAL I/O	(T4.01/T4.02)	4.68AVG.
T5.01	% OF LOGICAL I/O WHICH WERE LOGICAL READS	((T1.01/T4.01)*100)	66.46%
T5.02	% OF LOGICAL I/O WHICH WERE LOGICAL UPDATES	((T3.01/T4.01)*100)	33.54%
T5.03	% OF PHYSICAL I/O WHICH WERE PHYSICAL READS	((T1.02/T4.02)*100)	8.31%
T5.04	% OF PHYSICAL I/O WHICH WERE PHYSICAL UPDATES	((T3.02/T4.02)*100)	91.69%
T6.01	TOTAL PHYSICAL UPDATE WAITS DUE TO LOG FILE UPDATES		0 >>>>>.>>%
T6.02	TOTAL LOGICAL UPDATE WAITS DUE TO LOG FILE UPDATES		0 >>>>>.>>%
T6.03	TOTAL UPDATE WAITS DUE TO LOG FILE UPDATES	(T6.01+T6.02)	0 0.00%
T6.04	TOTAL NUMBER OF UPDATES TO THE LOG FILES(S)		590
T6.05	AVERAGE LOGICAL LOG FILE UPDATE PER LOG FILE WAIT	(T6.04/T6.03)	0 >>>>>.>>AVG.
T7.01	TOTAL STOLEN LOCKED RECORDS		0
T7.02	TOTAL SUCCESSFUL LOCKS WITHOUT WAITING		0
T7.03	TOTAL SUCCESSFUL LOCKS AFTER WAITING		0
T7.04	TOTAL SUCCESSFUL LOCKS	(T7.01+T7.02+T7.03)	0
T7.05	TOTAL FAILED LOCKS WITHOUT WAITING		0
T7.06	TOTAL FAILED LOCKS AFTER WAITING		0
T7.07	TOTAL EMBRACES DETECTED		0
T7.08	TOTAL FAILED LOCKS	(T7.05+T7.06+T7.07)	0
T8.01	TOTAL LOCKS REQUESTED	(T7.04+T7.08)	0
T8.02	% OF TOTAL LOCKS WHICH WERE SUCCESSFUL	((T7.04/T8.01)*100)	>>>>>.>>AVG.
T8.03	% OF TOTAL LOCKS WHICH FAILED	((T7.08/T8.01)*100)	>>>>>.>>%

Obtaining statistics from the File Statistics function

The File Statistics function of the SUPRA DBA Utilities reports various physical and logical characteristics of a file (not user activity as in “[Obtaining statistics from the Execution Statistics utility](#)” on page 29). This applies only to those database files that are in SUPRA native format.

Using these reports, you can monitor file growth and predict expansion needs. You can also use these reports on a cyclical basis to gather information to optimize file performance.

To use the File Statistics function, you code a Utilities Command Language (UCL) program. Refer to the [SUPRA Server PDM DBA Utilities User's Guide \(OS/390 & VSE\)](#), P26-6260, for information on coding this utility function.

You can request all statistics on all files or specific statistics on specific files and specific linkpaths. The PDM reads the requested files and calculates the current values. Using the ALL option can use much time and resources. Therefore, use the specific options when you do not need information on all files, all linkpaths, and all statistics.

The following sections contain examples of the various reports you can receive when you use the File Statistics function. The reports are:

- ◆ [Basic File Information](#)
- ◆ [Current File Size](#)
- ◆ [Linkpath Statistics](#)
- ◆ [Chain Statistics](#)
- ◆ [Code Statistics](#)

Requesting basic file information

To receive the Basic File Information report, you code BASE or ALL on the STATISTICS statement in your UCL. This reports the physical and logical characteristics of a primary or related file.

For key-sequenced data sets (KSDS), you receive no information on block size or record capacity. For BDAM files, the control interval size is not applicable. The following listing is an example of the information on this report:

FUNCTION = FILE STATISTICS	FILE = C\$-#
FILE TYPE = PRIMARY	
B A S I C F I L E I N F O R M A T I O N	
SCHEMA NAME	CINDIRSC
ACCESS METHOD	BDAM
LOGICAL RECORD LENGTH	374
BLOCKSIZE	4488
CONTROL INTERVAL SIZE	N/A
LOGICAL RECORDS PER BLOCK	12
LOGICAL BLOCKS IN FILE	600
MAXIMUM DATA RECORDS	7199
CONTROL RECORDS	1
TOTAL LOGICAL RECORDS	7200

Requesting current file size

To receive the Current File Size report, you code SIZE or ALL on the STATISTICS statement in your UCL. Use the report periodically to monitor the growth of primary and related files and to determine when you need to expand them. The report prints record and block statistics. The following listing shows statistics on the primary file C\$-#:

FUNCTION = FILE STATISTICS		FILE = C\$-#	
FILE TYPE = PRIMARY			
CURRENT FILE SIZE			
		ACTUAL NUMBER	% OF FILE CAPACITY
RECORD STATISTICS			
ACTIVE DATA RECORDS		2627	36.486
CONTROL RECORDS		1	0.014
RECORDS IN USE		2628	36.500
UNUSED RECORDS		4572	63.500
TOTAL LOGICAL RECORDS		7200	100.000
BLOCK STATISTICS			
EMPTY BLOCKS		4	0.667
BLOCKS IN USE		596	99.333
FULL BLOCKS		1	0.167
LOGICAL BLOCKS IN FILE		600	100.000
AVERAGE DATA RECORDS/BLOCK IN ENTIRE FILE			4.378
AVERAGE DATA RECORDS/BLOCK IN BLOCKS WITH DATA RECS			4.408

The File Statistics function calculates the average data records per block in the entire file and the average data records per block in the blocks with data records. The report shows those averages below the statistics. For key-sequenced data sets (KSDS), the only size statistics available are the numbers of active data records, control records, and records in use.

Requesting linkpath statistics

To receive the Linkpath Statistics report, you code LINK or ALL on the STATISTICS statement in your UCL. Use this report for a primary file and its associated related files to verify the accuracy of linkages. Compare the report figures between files. You receive one report for each linkpath on a file (up to 15). If there are more than 15, you can name the file and the linkpaths you want.

The following listing is an example linkpath report for a primary file. For key-sequenced data sets (KSDS), you do not receive statistics on file capacity or maximum data records.

FUNCTION = FILE STATISTICS				FILE = C\$-#		
FILE TYPE = PRIMARY						
LINKPATH	L I N K P A T H		S T A T I S T I C S			
	RECORDS WITH ACTIVE LINKPATH	% OF ACTIVE DATA RECORDS	% OF FILE CAPACITY	RECORDS WITH NON-ACTIVE LINKPATH	% OF ACTIVE DATA RECORDS	% OF FILE CAPACITY
C\$-#LKHD	1259	47.925	17.489	1368	52.075	19.003
C\$-#LKST	1259	47.925	17.489	1368	52.075	19.003
C\$-#LKWU	2453	93.376	34.074	174	6.624	2.417
C\$-#LKDA	767	29.197	10.654	1860	70.803	25.837
C\$-#LKTT	1262	48.040	17.530	1365	51.960	18.961
ACTIVE DATA RECORDS		2627			2627	
MAXIMUM DATA RECORDS			7199			7199

For a coded related file, you receive the total number of records and the number for each record code on each linkpath. The following listing is an example linkpath report for a related file:

FUNCTION = FILE STATISTICS				FILE = C\$-S	
FILE TYPE = RELATED					
LINKPATH	L I N K P A T H		S T A T I S T I C S		
	TOTALS FOR	RECORD CODE	NUMBER OF RECORDS	% OF ACTIVE DATA RECORDS	% OF FILE CAPACITY
C\$-#LKST	TOTAL		7392	100.000	23.072
	DT		6020	81.439	18.790
	HD		1372	18.561	4.282
ACTIVE DATA RECORDS			2627		
MAXIMUM DATA RECORDS					32039

Requesting chain statistics

Code CHAIN or ALL on the STATISTICS statement in your UCL to receive chain length and chain migration reports for both primary and related files. The reports are as follows:

- ◆ Chain Length Statistics report (for primary or related files)
- ◆ Chain Migration Statistics report (for primary or related files)
- ◆ Synonym Statistics report (for primary files only)

The Chain Length Statistics report for primary files shows the number of records randomized to the same home location. With this report, you can monitor the physical structure of chains and their accessing characteristics. The NUMBER IN CHAIN value is the number of primary records chained together (having the same home location).

The following listing shows an example of this report:

FUNCTION = FILE STATISTICS		FILE = C\$-N	
FILE TYPE = PRIMARY			
CHAIN LENGTH STATISTICS			
RECORDS RANDOMIZED TO SAME HOME LOCATION			
	NUMBER IN CHAIN	NUMBER OF CHAINS	% OF TOTAL CHAINS
	1	1879	84.336
	2	305	13.669
	3	40	1.795
	4	3	0.135
	5	1	0.045
	6	0	0.000
	7	0	0.000
	8	0	0.000
	9	0	0.000
	10	0	0.000
OVER	10	0	0.000
	TOTAL CHAINS		2228
CHAIN LENGTH -	MINIMUM	1	
	MAXIMUM	5	
	AVERAGE	1.179	

The Chain Length Statistics report for related files shows the number of records on linkpath chains. You receive one report for each linkpath on the file (up to 15). If there are more than 15, you can name the file and the linkpaths you want. You must use this method for a coded related file that has no linkpath in the base internal record.

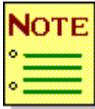
The NUMBER IN CHAIN value is the number of records chained together (or having the same control key value). The report tailors the 10 ranges of the NUMBER IN CHAIN column (see UNDER 2, 2 - 43, etc., in the sample report). It is tailored to the particular linkpath based on the distribution of chain lengths in the file. At least 80% of that linkpath's chains fall into the eight inside ranges; the remainder are under the minimum and over the maximum.

The following listing shows an example of the report:

FUNCTION = FILE STATISTICS		FILE = C\$-S		
FILE TYPE = RELATED				
C H A I N L E N G T H S T A T I S T I C S				
LINKPATH = C\$-#LKST				
		NUMBER IN CHAIN	NUMBER OF CHAINS	% OF TOTAL CHAINS
UNDER		2	0	0.000
	2 -	43	1243	98.729
	44 -	85	2	0.159
	86 -	127	8	0.635
	128 -	169	1	0.079
	170 -	211	2	0.159
	212 -	253	0	0.000
	254 -	295	0	0.000
	296 -	337	2	0.159
	338 -	379	0	0.000
	380 -	421	0	0.000
OVER		421	1	0.079
		TOTAL CHAINS		1259
CHAIN LENGTH -		MINIMUM	2	
		MAXIMUM	425	
		AVERAGE	5.871	

The Chain Migration Statistics report for primary files shows the number of block boundaries traversed by synonym chains. (The NUMBER OF DIFFERENT BLOCKS ENCOUNTERED portion of the report has been disabled pending performance enhancement in a later release. It will show all zeros.)

For an insert that randomizes to a synonym, the PDM attempts to place the record in the same block as the home synonym so that only one physical read is needed later. Your report should show very little boundary crossing. Use the report periodically to determine whether you need to reorganize a primary file to place synonyms in the same block.



Reorganizing means to carry out an unload/reload, possibly with a change in block size (see “[Reorganizing files](#)” on page 65).

The following listing shows an example of the report for a primary file:

FUNCTION = FILE STATISTICS			FILE = C\$-#			
FILE TYPE = PRIMARY						
C H A I N M I G R A T I O N S T A T I S T I C S						
NUMBER OF BLOCK BOUNDARIES TRAVERSED ENCOUNTERED			NUMBER OF DIFFERENT BLOCKS			
NUMBER OF BLOCK BOUNDARIES	NUMBER OF CHAINS	% OF TOTAL CHAINS	NUMBER OF BLOCKS	NUMBER OF CHAINS	% OF TOTAL CHAINS	
1	2	0.091	0	0	0.000	
2	0	0.000	0	0	0.000	
3	0	0.000	0	0	0.000	
4	0	0.000	0	0	0.000	
5	0	0.000	0	0	0.000	
6	0	0.000	0	0	0.000	
7	0	0.000	0	0	0.000	
8	0	0.000	0	0	0.000	
9	0	0.000	0	0	0.000	
10	0	0.000	0	0	0.000	
OVER	10	0.000	0	0	0.000	
TOTAL CHAINS		2193	TOTAL CHAINS		0	

The Chain Migration Statistics report for related files shows the number of block boundaries traversed by linkpath chains. (The NUMBER OF DIFFERENT BLOCKS ENCOUNTERED portion of the report has been disabled pending performance enhancement in a later release. It will show all zeros.)

For an insert, the PDM attempts to cluster the new record near other records in the chain using the *control* linkpath. The control linkpath should have the fewest boundary crossings. Use the report periodically to determine whether you need to reorganize a related file along the control linkpath to increase efficiency.



Reorganizing means to carry out an unload/reload, possibly with a change in block size (see “[Reorganizing files](#)” on page 65).

You receive one report for each base linkpath on the file (up to 15). If there are more than 15, you can name the file and the linkpaths you want. You must use this method for a coded related file that has no linkpath in the base internal record.

The NUMBER OF CHAINS value is the number of linkpath chains that cross that number of blocks. The report tailors the 10 ranges of the BLOCK BOUNDARIES column (see 1-10 and OVER 10 in the sample report). It is tailored to the particular linkpath based on the number of blocks traversed by the linkpath's chains. At least 80% of the linkpath's chains fall into the eight inside ranges; the remainder are under the minimum and over the maximum (see the [Chain Length report](#) earlier in this section for a different tailored range).

The following listing shows the report for a related file:

FUNCTION = FILE STATISTICS			FILE = C\$-S		
FILE TYPE = RELATED					
C H A I N M I G R A T I O N S T A T I S T I C S					
LINKPATH = C\$-#LKST					
NUMBER OF BLOCK BOUNDARIES TRAVERSED			NUMBER OF DIFFERENT BLOCKS ENCOUNTERED		
NUMBER OF	NUMBER	% OF	NUMBER	NUMBER	% OF
BLOCK	OF	TOTAL	OF	OF	TOTAL
BOUNDARIES	CHAINS	CHAINS	BLOCKS	CHAINS	CHAINS
1	13	1.033	0	0	0.000
2	1	0.079	0	0	0.000
3	3	0.238	0	0	0.000
4	0	0.000	0	0	0.000
5	0	0.000	0	0	0.000
6	1	0.079	0	0	0.000
7	0	0.000	0	0	0.000
8	0	0.000	0	0	0.000
9	0	0.000	0	0	0.000
10	0	0.000	0	0	0.000
OVER	10	0.000	0	0	0.000
TOTAL CHAINS		1259	TOTAL CHAINS		0
AVERAGE NUMBER OF READS TO TRAVERSE ENTIRE CHAIN - 1.024					

The Synonym Statistics report for a primary file (obtained when using CHAIN or ALL) shows the number of home and synonym records in a primary file and the percentages of capacity. With this report, you can monitor the number of out-of-block synonyms in primary files. Use this along with the Chain Length and Chain Migration reports for the primary file.

The report also shows the average number of physical reads to obtain a record. The report calculates the average as follows: (active data records + number of records not in home block)/active data records. For an example, use the numbers in the following listing. Add the active data records, 2626, to the number of records not in home block, 0, and divide by the active data records, 2626. The result is 1.

FUNCTION = FILE STATISTICS		FILE = C\$-N	
FILE TYPE = PRIMARY			
SYNONYM STATISTICS			
	ACTUAL	% OF	
	NUMBER	FILE	
		CAPACITY	
RECORDS AT HOME LOCATION	2228	31.473	
RECORDS NOT AT HOME LOCATION	398	5.622	
RECORDS IN HOME BLOCK	2626	37.	
RECORDS NOT IN HOME BLOCK	0	0.000	
ACTIVE DATA RECORDS	2626		
MAXIMUM DATA RECORDS		7079	
AVERAGE NUMBER OF READS TO OBTAIN A RECORD	1.000		

Requesting code statistics

To receive the Record Code Statistics report, include CODE or ALL on the STATISTICS statement in your UCL. You receive the statistics on only the coded files. If you name noncoded files in your UCL (or use FILE=(ALL)), the function ignores CODE for those files. You receive one report for each file having record codes.

With this report, you get the number of coded records, their percentage of active records, and their percentage of total file capacity.

The following listing is an example of this report:

FUNCTION = FILE STATISTICS		FILE = C\$-T	
FILE TYPE = RELATED			
RECORD CODE	NUMBER OF RECORDS	% OF ACTIVE DATA RECORDS	% OF FILE CAPACITY
LT	515	28.891	4.471
ST	1262	71.019	10.956
ACTIVE DATA RECORDS		1777	
MAXIMUM DATA RECORDS			11519

Obtaining statistics from the utilities Log Print function

The System Log Print function of the SUPRA DBA Utilities prints database record images from the System Log File. You can also use it to report statistics on the records it contains and the database update activity it represents.

To use the Log Print function, you code a Utilities Command Language (UCL) program. Refer to the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260, for information on coding this utility function.

You may select the following statistics to report:

- ◆ **BASE.** Basic summary on the log file itself. These statistics, which you can use for tuning, include spanned records and unused bytes as a percentage of total bytes.
- ◆ **ALL.** BASE plus a breakdown of activity on database files.
- ◆ **NONE.** No statistics (just record images).

The following listing is an example of a Log Print report of the Directory file's activity using STATISTICS=(ALL). In the TOTAL NUMBER OF TASKS ON THE LOG, for CICS this includes one for the interface plus all of the tasks that signed on through the interface.

Exits from Log Print enable you to perform several analyses of System Log File blocks including monitoring the reading and printing of those blocks. For details on Log Print exits, refer to the *SUPRA Server PDM Logging and Recovery Guide (OS/390 & VSE)*, P26-2223, and the *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260.

```

FUNCTION = LOG-PRINT                                FILE =
CSUL2161I : SYSTEM LOG FILE ANALYSIS STATISTICS (BASE).

LOG FILE NAME                                       LOGFILE
NUMBER OF LOG VOLUMES                               1
LOG DEVICE TYPE                                     DISK

NUMBER OF LOG BLOCKS                                73
NUMBER OF SPANNING BLOCKS                           0
SPANNING BLOCKS AS A PERCENTAGE OF TOTAL BLOCKS     0.00%

NUMBER OF LOG RECORDS                               616
AVERAGE NUMBER OF LOG RECORDS PER BLOCK           8.44AVG.
NUMBER OF SPANNING RECORDS                           0
SPANNING RECORDS AS A PERCENTAGE OF TOTAL RECORDS  0.00%
NUMBER OF RECORD BYTES                              139,771
NUMBER OF UNUSED BYTES                              590,229
UNUSED BYTES AS A PERCENTAGE OF TOTAL BYTES        80.85%

NUMBER OF COMMAND RECORDS                           297
COMMAND RECORDS AS A PERCENTAGE OF TOTAL RECORDS   48.21%
NUMBER OF BEFORE IMAGES                             156
BEFORE IMAGES AS A PERCENTAGE OF TOTAL RECORDS     25.32%
NUMBER OF AFTER IMAGES                              156
AFTER IMAGES AS A PERCENTAGE OF TOTAL RECORDS     25.32%
NUMBER OF CONTROL RECORDS                           7
CONTROL RECORDS AS A PERCENTAGE OF TOTAL RECORDS   1.14%

TOTAL NUMBER OF TASKS ON THE LOG                     2
NUMBER OF UPDATE TASKS ON THE LOG                   1
NUMBER OF TASKS SIGNED ON AT THE END OF THE LOG     0

NUMBER OF FILES ON THE LOG                           5

CSUL2163I : END OF SYSTEM LOG FILE ANALYSIS STATISTICS (BASE).

```

```

FUNCTION = LOG-PRINT                                FILE =
CSUL2130I : SYSTEM LOG FILE ANALYSIS STATISTICS (EXTENDED).

FILE ACCESS FILE      ADD      DELETE      READ      WRITE      BEFORE  AFTER  CONTROL
NAME METHOD  TYPE      COMMANDS  COMMANDS  COMMANDS  COMMANDS  IMAGES  IMAGES RECORDS
CS-#  BDAM  PRIMARY      0         9         53         1        68     68      1
CS-D  BDAM  RELATED      0         2         3          0         3      3       1
CS-N  BDAM  PRIMARY      0         9         32         0        15     15      1
CS-S  BDAM  RELATED      0        25        137        0        58     58      1
CS-T  BDAM  RELATED      0        10         10         0        12     12      1
TOTALS      0        55        235         1       156     156      5

CSUL2131I : END OF SYSTEM LOG FILE ANALYSIS STATISTICS (EXTENDED).

```

Obtaining RDM statistics

The Relational Data Manager (RDM) gathers statistics that show the number of RDML requests made by a user, plus the number of PDM DML requests made by the RDM when processing the user's RDML requests.

The following commands control the gathering and printing of RDM statistics for a DBAID session. They pertain only to the logical views being used by the DBAID user who issues the command.

- ◆ **STATS-ON**—Initializes statistics to zero and enables statistics' gathering on views for both the logical and physical levels
- ◆ **STATS-OFF**—Disables statistics' gathering
- ◆ **PRINT-STATS**—Prints the current statistics
- ◆ **STATS**—Displays current statistics online for all open views or a specified view

In addition to using DBAID, an RDML application can gather statistics for an execution. Before an RDML call, move the following statements into the TIS-OPTIONS field in the TIS-CONTROL-AREA:

- ◆ **SSTA**—Means start stats, corresponds to STATS-ON
- ◆ **ESTA**—Means end stats, corresponds to STATS-OFF
- ◆ **PSTA**—Means print stats, corresponds to PRINT-STATS

For information on interpreting statistics and for a sample statistics report, refer to the [SUPRA Server PDM RDM Administration Guide \(OS/390 & VSE\)](#), P26-8220.

Obtaining CICS statistics

The CICS Connector provides statistics that are written on the CICS Connector Activity Audit Trail. Your CICS system can also collect statistics.

Obtaining CICS connector statistics

The CICS Connector Activity Audit Trail logs messages about the status and activity of the CICS connector, including these items:

- ◆ All operator control commands issued
- ◆ Responses from the Operator Control Program
- ◆ Messages about connects, disconnects, abends, and task recoveries
- ◆ Log entries
- ◆ Trace entries

You can use these statistics to redefine the parameters you supply at connect or in the CSTXOTBL defaults table.

You use the CSTXOPRM macro (which defines the CSTXOTBL defaults module) to identify the message queue ID with the MESSQID parameter. This relates to the DESTID in your DCT and is used to write your SUPRA CICS Connector messages. You must define this message ID in your DCT (see the sample DFHDCT on your SUPRA CICSTBLS library).

To print the statistics, route your DDNAME MSGUSR to your printer, or print the data set associated with that name. This assumes that your DCT entries are the same as or similar to the SUPRA sample DFHDCT on your SUPRA CICSTBLS library. Messages from the CICS Connector Activity Audit Trail are prefixed with CSTX.

To display statistics at a terminal, use the STATUS command with the OPER transaction. The STATUS command returns run-time statistics about the following items:

- ◆ The number of tasks signed on to the PDM
- ◆ The number of CFUL, TFUL, and ICOR statuses automatically retried
- ◆ The number of CFUL, TFUL, and ICOR statuses returned to application programs

For details about the STATUS command, refer to the *SUPRA Server PDM CICS Connector Systems Programming Guide (OS/390 & VSE)*, P26-7452.

Obtaining CICS-supplied statistics

For CICS statistics, refer to your IBM CICS documentation. The possibilities for statistics vary according to release level.

3

Tuning the PDM

You can substantially improve performance by using tuning system and application factors related to the Physical Data Manager (PDM). This chapter provides PDM system-tuning considerations for [buffer pools](#), [block size](#), [CSIPARM parameters](#), [logging options](#), [file use](#), [task concurrency](#), [memory use](#), and [dispatch priorities](#). This chapter also provides application considerations for [database design](#), [concurrent updating](#), [committing work](#), and [using exits](#).

Many of the tuning strategies in this chapter involve PDM statistics. For guidelines on running those statistics, see [“Obtaining statistics”](#) on page 23. Memory use and space calculations are also important when tuning the PDM. For additional information on memory use, see the [“Memory use and space calculations”](#) on page 107.

Tuning buffer pools

Buffer pools are groups of buffers you specify when defining the schema/environment description. Use buffer count to specify the number of buffers in a buffer pool. You determine the size to use by the block size of the files that will use the buffer pool (CISIZE for ESDS files). If multiple files are sharing the buffer pool, the largest block size from the list of sharing files determines the buffer size. To compute the total memory requirement of a buffer pool, the PDM multiplies the buffer count by the largest block size of the files you relate to the buffer pool:

```
memory = buffer count * largest block size
```

This section provides guidelines for reallocating buffer pools and buffers to improve performance. If you have the PDM Buffer Cache Facility for BDAM files in place, also see [“Tuning the optional Buffer Cache Facility”](#) on page 77 for tuning considerations.

Calculating the buffer count value

For primary files related to a buffer pool, the buffer count should be large enough to handle the average number of users accessing any of the shared files times 2. The factor of 2 guards against peak periods and out-of-block synonyms. For example, 10 primary data sets might be sharing a buffer pool. Transactions or users are accessing 3 of the 10 files at any one time. On the average, 4 users are executing the application using these files. Therefore, in this example you would use a direct buffer count of 24 (3 files * 4 users * 2 factor).

For related files, use a similar formula. Instead of a factor of 2, the formula for related files needs to consider the average length of a linkpath chain. For example, 10 related files might be sharing a buffer pool. The average chain length is 40. The average records per block is 10, and $40 / 10 = 4$. With a typical transaction traversing 2 of the files and on the average 3 users, the buffer count for this buffer pool should be 24 (2 files * 3 users * 4 factor).



For index files, we recommend a minimum of 8 buffers per buffer pool. Allocate additional copies for the buffer pool in a multi-user environment. For a sample program for index file calculation, see [“Sample program for index space calculation”](#) on page 152.

Allocating buffer pools

When allocating buffer pools, consider the varying file block sizes and file activity in your environment. To conserve memory space, files with similar block sizes should share buffer pools, while files with dissimilar block sizes should not share buffer pools. It may be beneficial to place a small, high-activity file in its own buffer pool with enough buffers for the entire file to reside in memory. Files used concurrently should not share the same buffer pool.

Define buffer pools for the Directory and Task Log Files in the boot environment entity. Although you can relate buffer pools to those files in the Directory environment description, the PDM uses only the buffering information in the boot environment description for those files. You can define buffer pools for other files only in the Directory environment description entity.

Tuning the buffers

When tuning your system, ensure you have enough buffers allocated in buffer pools. Use the DBA Execution Statistics utility and PDM Interactive Services File Statistics to determine which data has high or low use, and the ratio of physical versus logical I/O values. When your tuning efforts improve performance, this ratio should decrease.

Use Directory Maintenance to change the buffer pool parameters for user files, the System Log Files, or the Statistics Log file. Use the buffer pool parameters in the Create Environment Description utility to change the Directory files and the Task Log File.

Increase the number of buffers for high-use data, and decrease the number of buffers for low-use data. Change the value of the Direct Buffer Count parameter and leave the other parameters set at zero.

If you cannot increase the number of buffers due to set memory constraints, you can first decrease the size of the buffers to provide more space, and then increase the number of buffers. In this case, you must change the file's block size and reload the file, using the SUPRA DBA Utilities' Unload and Load functions. Another technique is to move extremely low-use files into buffer pools used by high-use files.

The following are some guidelines for allocating buffer pools and buffers for different types of files.

- ◆ For read-only KSDS related to a buffer pool, define enough buffers to support the activity. For update KSDS, define a single buffer pool with the buffer count as the number of concurrent updates. Also, when defining update KSDSs, use the smallest possible number of records per control interval, because any update activity forces the buffers to flush.
- ◆ Allocate one buffer pool for statistics files.



Allocate one buffer pool for the Task Log File. We recommend a buffer count of 4 as a starting value. Block size should be between 4K and 8K. You should adjust this buffer based on the systemwide wait-on-log statistic.

- ◆ You can improve task logging performance by reducing the number of forced flushes, which delay processing and increase the number of physical I/Os. Increasing the number of task log buffers provides more space for storing log records, and thereby reduces the need for a forced flush to the Task Log File. The ideal number of task log buffers increases with the number of users and decreases with the frequency with which their applications reach commit points.
- ◆ The number of data file buffers can also affect task logging performance. The flushing of a data record triggers the forced flushing of the Task Log File buffer containing that record's before image. The PDM forces the Task Log File buffer to be flushed first. Therefore, increase the number of buffers for high-use update data files to reduce forced flushes of the data buffers and, ultimately, forced flushes of the Task Log buffers.
- ◆ For System Log Files, use 2–5 buffers per log group. One buffer per log group is an absolute minimum and would degrade performance. (The number of files in a log group does not correspond to the number of buffers needed.) System Log Files must not share a buffer pool with data files. Adjust this count based on the systemwide wait-on-log statistic. Block size should be approximately 8K.

- ◆ When populating index files, you should reduce physical I/O as much as possible by creating a separate environment description with buffer pools for only the data file and the index file(s). If you are not using the Sorted-Populate utility, define the buffer pool for the data file with only 2–3 copies, and use all available space for the index file by eliminating unused pools. If you are using the Sorted-Populate utility, define 5 buffers for the low-level blocks and 1 buffer for each level of the index structure. For the average number of secondary keys in use, a buffer count of 8 is the recommended starting value.
- ◆ After the application is stabilized, review the buffer pool used by the user and system files. For user files, combining pools and sharing buffers may optimize the use of computer resources by the application. You can also look at high-activity files and very small files that can become memory resident. You may have to remove these from a common pool and create a special pool. In the statistics, an in-memory, hit rate percentage of 80% for related and 60% for primary should be minimum goals.

Tuning block size

Defining and tuning the block size of various types of files is a most-important effort in performance improvement. Use the following guidelines.

Defining block sizes

Use Directory Maintenance to define and change all file definitions, except the Directory files and the Task Log File. Use the Modify Schema utility for these files.

A suggested rule for block sizes is to restrict them to less than the virtual storage page size. Too large a block size increases device, string, and channel processing. This may slow or halt I/O accessing the defined file or any other files sharing the same hardware devices (channel, volume, etc.). As a general rule, set the block size to eliminate or reduce out-of-block synonyms. You can use the following formulas to calculate the block sizes for various file types:

- ◆ For BDAM or ESDS primary files, use this formula to assure minimum, physical I/O when traversing synonym chains:

`Record length * 5 (minimum 1K)`

- ◆ For related files, use this formula to assure minimum, physical I/O when traversing linkpaths:

`Record length * average number of records on the clustered
or primary linkpath`

- ◆ For the Statistics File, block size is not critical due to the low level of activity on that file.

- ◆ For the System Log File, block size, buffer size, buffer count, and device affect performance. The suggested initial block size is 4K.

Change the block size if the System Log File becomes a bottleneck (block not written before it fills up). To determine if the block-size requires a change, run LOGPRINT with LIST(NONE) and STATISTICS(ALL). Examine the block size displayed and the amount of free space. A high ratio of unused space in a block indicates that the block size is not a bottleneck. A low ratio of unused space in a block indicates that the block size could be a bottleneck (tasks slow down while waiting for I/O).

To change the block size:

- Use Directory Maintenance to change the System Log File(s).
 - If it is a BDAM or ESDS file, allocate and format the file.
 - Change any procedures such as operator procedures and JCL.
 - Cycle the PDM.
 - Verify the effect of the change (check for block-size bottleneck and monitor response time or elapsed time for DML). Remember that System Log Files of different block sizes cannot be processed in a single run by the Log Print and Recovery function. Watch the wait-on-log statistics.
- ◆ For the Task Log File, you need to find a balance between a large block size, which is good for logging large data records, and a small block size, which improves disk space utilization. A smaller block size is efficient during PDM secondary key maintenance. The PDM writes short records—often around 20 bytes—to the Task Log, using an entire block for these records. In a system with many secondary keys, a block size of perhaps 1K is recommended. Where there are fewer secondary keys, some performance improvement may occur with a larger block size, perhaps 4K, for the Task Log.
 - ◆ For index files, see the Index File Space Calculations in the “[Memory use and space calculations](#)” on page 107. The formula estimates the depth and total number of blocks required for a given set of attributes of a secondary key. For any secondary key, the key size, number of records indexed, and uniqueness can be considered fixed. Therefore, the only variables are block size and depth of tree. The depth of tree affects the number of buffers required in the pool to minimize I/O. In general, the block size should be 4096 bytes and may be as large as 8192 bytes.

Optimizing block size

When tuning, check block size and increase or decrease if necessary. Use Directory Maintenance to change the Blocks Per Track attribute on the file. Directory Maintenance automatically recalculates the value for Records Per Block/Control Interval.

Use the SUPRA DBA Utilities File Statistics function to determine block usage. For primary files, check the synonym statistics for the Records Not in Home Block and the corresponding percent of File Capacity statistic. If the file capacity percentage for those records increases, response time increases. Increase the block size to decrease the amount of I/O and increase disk space efficiency. However, this could decrease response time for single record retrieval since a longer block is read.

Use the SUPRA DBA Execution Statistics utility or the PDM and Interactive Services File Statistics to compare the ratio of logical I/Os against the physical I/Os. For primary files, set the block size at the point just prior to where the out-of-block-synonym rate begins to rise rapidly. For related files, the block size should equal the average chain length times the record size.

Tuning CSIPARM parameters

In the OS/390 environment, you can run in central or attached mode. To change modes, bring down the PDM and change the CSIPARM file. To run attached, use the ATTACH=DBMS parameter. To run central, remove ATTACH and use the DBM parameter.

Run in attached mode if you have a greater volume of online than batch access, and your PDM configuration is fairly static. In this mode, CICS, the PDM, RDM, and SPECTRA are in the same address space. Therefore, if CICS or the PDM abends, the other components are also down.

Run in central mode if you anticipate PDM changes (PDM not static) during peak operating hours. Thus, you can change the PDM environment independently from CICS.

In the VSE environment, the SUPRA PDM is able to work with other applications in different partitions within or outside the PDM address space. Within the address space, the SUPRA PDM uses XECB communication. Outside the address space, the SUPRA PDM uses a packet communication method based upon the XPCC services of VSE. To enable this feature in SUPRA Server, specify PATH=(XPCC=YES) in the CSIPARM file. For more information on the CSIPARM file, refer to the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250.

Although the performance of XECB communication is more efficient than XPCC communication, dynamic partitions in VSE/ESA no longer support XECB communication. The following is true only for VSE/ESA in static partitions. By placing an I/O-intensive batch program in the same address space as the SUPRA PDM, the batch program will perform better than when it is executed in a different address space. Due to the program addressability restriction of 16 MB and VSE limitations, entire systems probably cannot fit into a single address space. Place partitions for batch jobs into the same address space as the SUPRA PDM and place the online system in a different address space. Both the application and the SUPRA PDM must have the PATH=(XPCC=YES) option in their CSIPARM file. “[Performance considerations for VSE XPCC](#)” on page 144 has memory usage information concerning VSE XPCC support in SUPRA.

The MAXPACKET parameter in the CSIPARM file may affect performance. MAXPACKET provides VSE XPCC support. Avoid using an unnecessarily large value for MAXPACKET. The default of 15K should suffice for most applications.

The RELMAP and PRIMAP options in the CSIPARM file may affect performance tuning. These parameters enable or disable the file experience facility of SUPRA. [“Using the file experience table to reduce I/O”](#) on page 64 has more information concerning the file experience facility of SUPRA.

If you have the PDM Extended Storage Support option, which places PDM control blocks above the 16 MB line, use the XAMEM parameter of CSIPARM to activate it. This option provides more storage below the line for tuning the MEMORY parameter of the environment description. See [“Tuning the optional PDM Extended Storage Support”](#) on page 75 for more information about the PDM Extended Storage Support option.

For more information on the CSIPARM file, refer to the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250.

Using system logging options

The amount of information logged to the System Log affects overall throughput. The amount varies with your choice of System Log options (before image, after image, function, etc.) and with the amount of update activity.

Display the Log Options parameter in the environment description to determine which System Log options you are using. Consider revising them after reviewing the following points:

- ◆ You can activate options for tuning and testing as required, and you can deactivate them when tuning or testing is complete.
- ◆ When using before-image logging, the number of before images written to the System Log depends on whether task logging to the Task Log is active or inactive:
 - If task logging is not active, one before image is written to the System Log File each time a record is updated.
 - If task logging is active, only one before image is written to the System Log File per physical record regardless of how many times the application updates it in a logical unit of work.
- ◆ In most cases, you should limit your logging options for disaster recovery. In other words, if you experience infrequent device failures, do not use before image system logging when using task logging. The Task Log File contains all before images. However, if you log both before and after images to the System Log File, you ensure recovery if the Task Log File fails. If you do not use before image system logging and a Task Log File failure occurs, you must use disaster forward-recovery procedures. This is more costly during recovery but saves log I/O during normal processing.
- ◆ PDM KSDS and index files usually require additional System Log File resource consumption.
- ◆ Consider using the selective logging by file feature. This allows you to suppress system logging of before images, after images, or function (PDM command) images. It also allows suppression of task logging for the file. If you suppress task logging, system log before images and after images must also be suppressed. All files that are dependent on each other, such as a primary file, its related files, and their indexes, must opt for the same logging options. However, you should use this feature only for files that do not need recovery, such as work files.

Use the largest efficient block size possible for the System Log File (see “[Defining block sizes](#)” on page 54 to tune). To determine efficiency, use the Log Print function of the SUPRA DBA Utilities to get unused byte and spanning statistics for the System Log (see “[Obtaining statistics from the utilities Log Print function](#)” on page 44).

After tuning block size, buffering and choice of options, check the Execution Statistics for the amount of I/O to your System Log File. If you have quite low activity, use the IBM utility channel monitor to determine if the file is on a high-activity disk volume. If so, reassign it to a low-activity disk volume (see “[Placing files on devices](#)” on page 63).

For more information on logging, refer to the *SUPRA Server PDM Logging and Recovery Guide (OS/390 & VSE)*, P26-2223.

Calculating file capacity

Use Directory Maintenance File parameters to define and change capacity for all files except the Directory files and the Task Log File. Use the Modify Schema utility for those files. For information on Directory Maintenance, refer to the *SUPRA Server PDM Directory Online User's Guide (OS/390 & VSE)*, P26-1260, or the *SUPRA Server PDM Directory Batch User's Guide (OS/390 & VSE)*, P26-1261. For information on the Modify Schema utility, refer to the *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250.

Use the following guidelines to calculate file capacity for various file types:

- ◆ For BDAM or ESDS primary and related files:
 - Separate the files into volatile and nonvolatile categories.
 - For nonvolatile files (where adding and deleting records is minimal), estimate your projected number of records and multiply by 1.15. Use the result as input to Directory Maintenance for total logical records. For related files, the total records should be the average chain length along the clustered (or primary) linkpath times the number of records in the primary file with which it is clustered.
 - For volatile files (which require more space for SUPRA data maintenance algorithms to remain optimal), multiply the projected total records by 1.4 as a start. You can use a factor range of 1.4–10.0, limited by projected growth and your disk resources. Use the result as input to Directory Maintenance for total logical records.
- ◆ For index files, use the Index File Space Calculations in the “[Memory use and space calculations](#)” on page 107.

- ◆ For the Statistics File, file size is not critical due to the low level of activity on that file. However, the PDM abends if enough space is not available. The PDM writes approximately 350 bytes + (200 bytes * number of files in the system) at PDM initialization, also at termination, and each time any application issues an RSTAT DML command. Ensure that sufficient space is available. The space required is a combination of block size, the number of files, and the number of RSTAT DML commands applications issued to produce records, in addition to the automatic initialization and termination records the PDM writes.
- ◆ For the System Log File, several interdependent options are available:
 - During disk logging, the PDM can abend if it runs out of space unless you specified the Log Group Wrap option. If specified, the PDM reuses the data sets (upon operator confirmation). If you are performing disk logging, you should define 2–4 disk data sets and develop procedures to dump the data sets. Ensure that the data sets are large enough, the dumps are meaningful, sufficient space exists for the time necessary to dump the other data set, and minimum intervention is required from the operator. Inputs are also available to automatically submit jobs to dump log files.
 - For tape logging, size is not a tuning consideration except for the reel length. Use the longest tape possible to minimize the number of mounts. Parallel mount log tapes if possible. Do not use 3480 tape drives.
- ◆ For the Task Log File, there are no statistics and formulas to help with optimization of capacity. For general information about the Task Log File, refer to the *SUPRA Server PDM Logging and Recovery Guide (OS/390 & VSE)*, P26-2223.

Placing files on devices

Improperly placing files on volumes can affect response. Consider the number of channels, the number of disk volumes, and the number of files being used. You can use DASD manager utilities to spread files evenly across channels. Place high-activity files used at the same time daily on separate disk volumes. Ensure that disk volumes and/or channels are not over-used or under-used.

Due to high activity, place the Task Log and System Log Files on separate devices. Distribute other high-activity database files, teleprocessing files and operating system files on different channels and separate devices. Use a system monitor to detect contention.

Avoiding record contention

The PDM enqueues on a block of data only until the desired record is located, then it changes the enqueue to the specific record. The PDM releases a task's record enqueues at the task's commit points. If a task has too few commits, other tasks receive a HELD status until the commit is issued and the record is freed. See ["Handling concurrent application updates"](#) on page 72 and ["Coding commits in your application program"](#) on page 73 for instructions on using commits.

Tuning the number of concurrent PDM tasks

For a central PDM, you define the maximum number of PDM tasks that can be signed on via the environment description. You also define the maximum concurrently active (maximum concurrent threads). Make the threads' value equal to the number of CICS interface threads plus the number of concurrent batch tasks (each using one thread). To improve performance, you can tune each CICS Connector with the OPER CONNECT parameters (see ["Tuning OPER CONNECT parameters"](#) on page 82). See the illustration under ["Defining tasks"](#) on page 90 for the relationship of PDM, CICS Connector, and RDM tasks and threads.

Using the file experience table to reduce I/O

Use file experience tables (also known as bit maps) for related files if most of them are dense. Modify the CSIPARM file to turn on or turn off the file's experience table usage. There are two options:

- ◆ PRIMAP for primary files (Default=No).
- ◆ RELMAP for related files (Default=Yes).

The PDM ignores the option for Task Log File, System Log File, Statistics, or KSDS (primary or related). For detailed information on the CSIPARM file parameters, refer to the *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250.

When adding a record to a primary or related file, the PDM searches a physical block in a file. If the PDM finds no free space in that block and the experience option is turned on for the file type, the PDM updates the file experience table. The next time the PDM searches the file for space, it skips any block marked Full in the experience table. This reduces the number of I/O operations the PDM performs to locate space for new records.

In some instances where you must tolerate very dense files until a scheduled expansion occurs, you can write an application to help end users. You can execute a transaction during system initiation to add and delete against the dense file to prepare the experience table. This initial transaction will take some time to search for an available record area. During the search, the PDM updates the file's experience table. This built experience table allows subsequent end-user transactions against the file to execute in a shorter time because the PDM then finds free space quickly.

Experience tables are not recoverable. The experience table is rebuilt with each PDM execution.

Use the following formula to calculate the amount of storage required for an experience table:

$$(((\text{tot-logical-recs} + 1) / \text{recs-per-blk}) / 8) + 25$$

OS/390

If you are using the PDM Extended Storage Support option, and if you have specified enough memory above the line to build all possible control blocks, this table storage is above the 16 MB line.

Reorganizing files

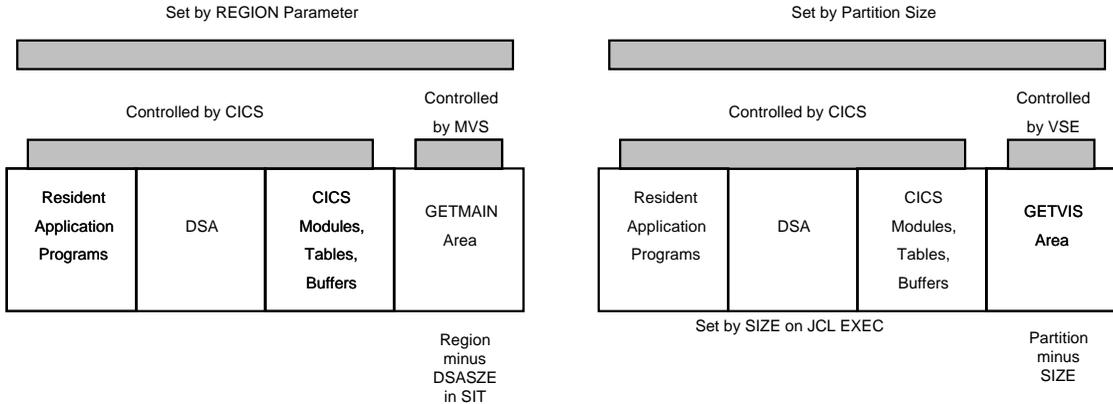
File reorganization can be valuable for primary files (to reduce out-of-block synonyms), related files and index files (to place chains in physical proximity). You should use the SUPRA DBA Utilities File Statistics function to check for too many out-of-block synonyms and for fragmented, primary-linkpath chains. After analyzing the statistics, you may want to reorganize the file and designate a new primary linkpath to correct fragmented chains. The following are reorganization strategies:

- ◆ Use the SUPRA DBA Load and Unload utilities to reorganize primary and related files. You can use the utilities to load a file onto one volume, if possible, and to increase the file size.
- ◆ Use the SUPRA DBA Reorganize utility to reorganize secondary keys on an index file. You may need to use the Sorted-Populate utility. Before and after secondary key reorganization, use the SHOWX command to list index file statistics (see “[Obtaining statistics from PDM DML commands](#)” on page 28). For the syntax and output fields of the SHOWX command, refer to the *SUPRA Server PDM DML Programming Guide (OS/390 & VSE)*, P26-4340.
- ◆ For KSDS, you may need to increase or reclaim free space. To avoid having a KSDS file on multiple volumes, decrease the amount of free space in the overflowing file during allocation, or increase the amount of free space in the overflowing file during allocation and then move it onto one volume.

Using PDM memory

Memory use can have a significant impact on performance.

In a CICS environment, memory consists of resident application programs; the Dynamic Storage Area (DSA) that contains working storage; CICS modules, tables and buffers; and GETMAIN or GETVIS storage (free space). The following figure shows space allocation for CICS Connector OS/390 and VSE environments.



You must provide for sufficient GETMAIN or GETVIS area for your non-CICS tasks (RDM, SPECTRA, and VSAM data buffers). For example, in VSE, if you set SIZE to 4 MB and the partition size is 7 MB, you will have a 3 MB GETVIS area for your RDM, SPECTRA, and VSAM control blocks.

In OS/390 attached mode, there must be enough GETMAIN space for the PDM. For OS/390 CICS systems up to CICS 2.1, the OSCOR parameter provides the space. Check the DSA space and the OSCOR and REGIONSIZE parameters for optimization. When processing, CICS acquires the entire address space and then releases the amount specified in the DFHSIT OSCOR parameter back to OS/390 control. For CICS higher than 2.1, the GETMAIN area is determined according to the preceding illustration.

Determining domain and dispatch priorities for central PDM

A central PDM should execute in a different performance group than its connected tasks and CICS. You assign dispatch priorities for the PDM to service more than one DML request each time the operating system returns control. If running in central mode, check the assignments for dispatch priorities for CICS, PDM, and batch jobs. The PDM should be immediately lower than CICS.

Check the run-time statistics for less than one command per cycle. If you assign the central PDM a lower priority, the processing tendency is to have several functions or services accomplished without losing control. The range of response time may vary more in this case, but the average is less than that achieved from other configurations. More importantly, the average amount of resources consumed per transaction is less. However, this assumes that the service units expended this interval do not exceed the units specified in the objective times the interval service (ISV, see [“Using interval service \(ISV\) in attached mode”](#) on page 70).

Assigning priorities for a single, central processor system

If you are using a single, central processor system, consider these items when assigning priorities:

- ◆ If the PDM is not tuned, a higher priority may be more beneficial. However, if the system is not busy, a higher priority may have no effect.
- ◆ If the PDM is tuned, run the PDM lower so requests can stack up from CICS before the PDM starts processing.
- ◆ Give the PDM or CICS every chance to complete processing before losing control (high ISV; see [“Using interval service \(ISV\) in attached mode”](#) on page 70).
- ◆ Reducing concurrent threads is more beneficial than attempting to process more threads than the physical-to-virtual ratio (memory) can reasonably handle.

If the PDM is assigned the highest priority, then the operating system gives control to the PDM as quickly as each task issues a DML request. Therefore, the PDM services only the waiting DML. If you want CICS tasks to have the highest priority during the regular business day, assign dispatch priorities as shown in the following figure. Batch jobs are usually not as important during the day.

Operating System	CICS			PDM	Batch Task 1	Batch Task 2	Power (VSE)
	Task 1	Task 2	Task 3				
	Task 4	Task 5	Task 6				
Domain (MVS)	A	A	A	A	A	A	N/A
Dispatch Priority MVS	Highest	Next Highest	Lower	Lowest			N/A
Dispatch Priority VSE	2	3	4	5			1

Using this example allows all CICS tasks that need PDM service to post the PDM before the operating system turns control to the PDM. When the PDM gets control, it uses DML function-overlap facilities for efficiency.

When the regular business day is over, you can change the assignments to give batch tasks higher priority. You can make CICS tasks either equal to or lower than the batch tasks, as shown in the following figure:

Operating System	CICS			PDM	Batch Task 1	Batch Task 2	Power (VSE)
	Task 1	Task 2	Task 3				
	Task 4	Task 5	Task 6				
Domain (MVS)	A			A	A	A	N/A
Dispatch Priority MVS	Lowest			Higher	Next Higher	Highest	N/A
Dispatch Priority VSE	3			2	3	3	1

Assigning priorities for a central multiprocessor system

Use job class and initiators to assign the CPUs. The PDM priority should equal CICS priority.

Using interval service (ISV) in attached mode

When running under OS/390 in attached mode, the PDM and CICS are in the same address space. Therefore, ensure the Interval Service (ISV) is sufficient to allow both the PDM and CICS to complete all processing without System Resource Manager (SRM) intervention. ISV is the number of service units allowed for the Performance Group Number before SRM interrupts and gives control to another address space.

To determine the value of ISV, check the System Management Facility statistics, run a heavy transaction, and check the statistics again to determine the service units expended by that transaction. Multiply these service units by the number of threads to obtain the ISV value for the performance group.

Optimizing your database design

The structure of your database and the way you access it has a great impact on overall performance. Database design includes files and their relationships, record layouts, file usage, file size and frequency of use, and physical system configuration. All of these factors affect performance. You use Directory Maintenance to design your database.

The design of database files and their relationships is important in logically grouping data to maintain data integrity and to facilitate data access and maintenance. Data integrity ensures that nonkey data is maintained in only one place and that redundant key data (foreign keys) is maintainable.

The definition of the control key (ORDERED or HASHED) and foreign keys (CLUSTERED, CHAINED, or INDEXED) determines the type of generated database. The control key is the primary key of the primary files and the base control key of the related files. An ORDERED control key indicates that control key sequential order is maintained by a secondary key. HASHED indicates that the control key is maintained in random order. CLUSTERED, CHAINED, and INDEXED generate (based on the relation type) primary, related, or index files, respectively.

When designing your database, consider the following:

- ◆ Type of access and maintenance use of the relations
- ◆ Type of relation in which the foreign key exists (essential, dependent, relationship, or extension)
- ◆ Frequency of maintenance and the percentage of daily maintenance to read access for each relation

Review your database design and optimize if possible. You may need multiple copies of the most active data for concurrent users.

Handling concurrent application updates

If you receive no response for a task, your task may have stalled due to concurrent updates from other applications. To correct the problem, try to reschedule job processing or redesign your applications. Turn on statistics for the PDM (see “[Obtaining PDM statistics](#)” on page 23) and check the system and file statistics with Interactive Services or SHOWX DML (see “[Obtaining statistics from interactive services](#)” on page 25 and “[Obtaining statistics from PDM DML commands](#)” on page 28). You can use the RSTAT DML command to record PDM execution statistics at various points in PDM operation, whether one task or multiple tasks are active.

Reschedule batch jobs that update the same records as your online jobs. Batch jobs should execute a commit before and after updating, and updating activity should be grouped together if possible. If a batch job abends, recovery is required only if the task was in update logic at the time of the abend.

If the Maximum Held Records (highest number held at any one time during the run) and the Current Held Records are frequently the same, you may need to increase record holding capacity. If the statistics’ Maximum Held Records value is frequently equal to the environment description’s Maximum Held Records parameter allowance, you need to increase record holding capability. Use Directory Maintenance to change the value of the Maximum Held Records parameter in the environment description. This may require a corresponding change to the environment description’s MEMORY parameter. If you are using the PDM Extended Storage Support, the corresponding change would be to the CSIPARM XAMEM parameter.

Coding commits in your application program

Code your applications to use small logical units of work. A COMMIT (RDML) or COMIT (PDML) defines a logical unit of work. In general, issue enough commits or sign-offs to release held records in a timely manner per any given task. You should use commits to allow recovery to a point where you or the computer operator (batch processing) can restart or reenter a request or job. For optimum performance, use a commit at the end of a logical unit of work where a task is restartable.

Each time SUPRA encounters a commit, it performs I/O operations against the PDM files and flushes the log buffers. If you have too many commits, you increase I/O to the Task Log File and System Log File. If you have too few commits, consider the following:

- ◆ Other tasks wishing to update the uncommitted records receive a HELD status until the commit is issued and the records are freed.
- ◆ Read-only tasks can access altered uncommitted records and could report information or update other records, based upon these uncommitted records. If the holding task abends, the PDM reverses any updates. A read-only task could report information or perform an update on other records, based on data that was subsequently reversed.

Optimizing user exits

When tuning, check your environment descriptions to determine if you are using user exits for the statistics or log files. Too many or inefficient user exit programs slow processing. Errors in a user exit can terminate processing. Check for optimal execution paths in user exits. Check user-exit coding to ensure the instructions are fast executing.

Nonoverlapped I/O in your exit can slow response time. For example, if you write a record to your own statistics file, and then perform a WRITE followed immediately by a CHECK, the CHECK causes the entire PDM to go into an operating system wait.

You can decrease CPU usage by eliminating unnecessary user exits in your environment descriptions. Use Directory Maintenance to display the exit names. Blank out any exits you do not need.

Defining dynamic indexes

In certain instances, using secondary keys can result in more efficient resource consumption. You should define secondary keys for data retrieval where feasible. Design your secondary keys so that read access for frequently read data is wholly contained in the second key. This eliminates additional I/O on the primary and related files in most cases. When you update a record with indexes defined on it, the maintenance of the index increases the number of logical I/Os the PDM requires to process the update request. Consider the impact of secondary key maintenance for indexes you define for high-activity files.

If you have large files (greater than 150,000 records), you should reduce physical I/O as much as possible. Therefore, create a separate environment description with buffer pools for only the data file and the index file(s) for single-task processing. If you are not using the Sorted-Populate utility, define the buffer pool for the user file with only 2–3 copies, and use all available space for the index file by eliminating unused pools. If you are using the Sorted-Populate utility, define 5 buffers for the low-level blocks and 1 buffer for each level of the index structure. See [“Calculating index space”](#) on page 148 about index structures.

Use multiple extents across 2 or more physical drives for the index file and/or multiple index files for secondary indexes built from the same data file.

If you are not using the Sorted-Populate utility, build the index during populate by serially sweeping the data file. Avoid specifying a secondary key that will have updates arrive in ascending order of the secondary key (a secondary key on the key to a primary file).

Tuning the optional PDM Extended Storage Support

If you have the option, which places many PDM control blocks above the 16 MB line, use the XAMEM parameter in the CSIPARM file to activate the option. (For all CSIPARM file parameters, refer to the *SUPRA Server PDM and Directory Administration Guide*, P26-2250.) The PDM Extended Storage Support option provides a pool of memory above the line and frees storage below the line in the existing MEMORY parameter allowance of the environment description. You can use this option along with the Buffer Cache Facility (see “[Tuning the optional Buffer Cache Facility](#)” on page 77) for using storage above the line. You are not required to use both.

By using PDM Extended Storage Support, you can add buffers, users, record holding entries, and so on, to your environment description using the same memory allowance, or reducing the allowance. You can keep the memory allowance the same but begin using the PRIMAP and RELMAP parameters of the CSIPARM file for experience tables or add to TRACESIZE, and so on.

To monitor this feature, analyze statistics gathered from two sources: OS/390 paging rates for the PDM and PDM execution statistics. For PDM execution statistics, see “[Obtaining statistics](#)” on page 23.

Realize that the extended storage memory pool might not be the only SUPRA resource or other resource, residing above the line. For example, if you have the RDM Extended Storage Support option, global views and heaps may be above the line. If you have the Buffer Cache Facility, it is above the line.

You can adjust several areas for tuning when using PDM Extended Storage Support:

- ◆ Page size in the CSIPARM file—In general, a smaller PDM-page size conserves storage resources while a larger PDM-page size conserves CPU resources. The CSIPARM-page size pertains to memory below and above the line. This should be a submultiple of or equal to the operating system's page size. If greater, this could impose additional work on the operating system paging facility.
- ◆ Memory pool size in the environment description—This specifies storage below the line for buffers and various control blocks.
- ◆ XAMEM pool size in the CSIPARM file—This specifies storage above the line for various control blocks.

To determine how much memory previously below the line is moved above at your site, run the PDM for 5 minutes without the XAMEM parameter, then run again for 5 minutes with it. Compare the CSTA417 and CSTA418 messages. If you do this while running scripts to simulate your production activity, it also reflects the size moved above the line for ACTV task table space.

Tuning the optional Buffer Cache Facility

If you have the Buffer Cache Facility, you can improve both memory constraints and PDM performance. You can use the Buffer Cache Facility along with PDM Extended Storage Support (see “[Tuning the optional PDM Extended Storage Support](#)” on page 75). You are not required to use both.

Buffer caching (for BDAM files only) provides additional file buffers above the 16 MB line. You can reduce some buffers below the line in the existing environment description MEMORY allowance. You can then use the lower memory allowance for other PDM needs, or even reduce the allowance.

The Buffer Cache Facility can improve PDM performance by reducing physical reads to BDAM PDM files. You assign the buffer pools in the cache and the files which belong to each pool (pools are similar to those below the line). A physical read stores the block in the buffer for the file (below the line) and in the buffer cache (above the line). The block stays in the buffer cache until it is the least recently used in its pool and a new block replaces it. If an update occurs to the buffer block, the cache buffer block is also updated.

To tune your buffer caching, consider these points:

- ◆ A tradeoff exists between PDM I/O and system paging I/O. To achieve an overall increase in performance, sufficient real- or extended-memory resources must exist to support paging.
- ◆ Use PDM and system statistics to determine which BDAM files to cache; the files with the highest physical read counts are candidates to mix into a pool with those with low counts. High-read files usually include the Directory files and indexes on high-access files. Also consider the performance of different channels, control units, and devices. Improper assignment of files to the cache could degrade overall system performance.
- ◆ Files that you often serially access for a large number of records should not share a cache pool with other files. You should assign a dedicated cache pool. (Remember that RDM uses serial access for some views.)
- ◆ Once you have assigned files, you can use the buffer cache's console-message statistics to further tune the sizes and assigned files. The messages occur when the PDM initializes, when a file closes, and at PDM termination.
- ◆ The file buffer pools that remain below the line must still be sufficient to support concurrent physical activity.

Refer to the *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250, for basic information about the Buffer Cache Facility.

4

Tuning the Directory

You can improve performance for the Directory by tuning with the methods described in this chapter.

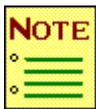
Using multiple load modules

Directory Maintenance provides multiple load module support. This helps you tune Directory Maintenance based on performance versus memory usage.

The installation tape contains the Directory Maintenance configuration that provides the most efficient use of memory:

- ◆ For OS/390, 1 load module for batch processing and 12 load modules for CICS processing
- ◆ For VSE, 6 load modules for batch processing and 12 load modules for CICS processing

However, you have the option of using 1, or 3–12 load modules for batch processing, and 2–12 load modules for CICS processing.



The fewer load modules specified, the more memory resources are used per transaction or function executed. Conversely, the more load modules specified, the fewer memory resources.

The Directory Maintenance load modules and memory requirements are listed in the “[Memory use and space calculations](#)” on page 107. To change the number of load modules, use the C\$MDMLNK macro (refer to the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250).

Placing directory files on disk

The Directory contains five files: NAME, DEF#, STRU, DATA, and TEXT. You should place the NAME, DEF#, and STRU files on separate physical drives or channels. These three files have high activity and are paramount to the performance of SUPRA Server.

You can control buffering of the Directory files in the boot environment description.

Processing a Directory

You should use batch Directory Maintenance to perform major tasks such as copying or deleting a schema, or checking the consistency of a schema. Otherwise, you must configure the production PDM for such tasks. For example, copying the Directory schema requires a Maximum Held Records parameter in the user environment description of at least 4600 entries. Normal user applications possibly do not need such a large value. Relegating such major tasks to a batch, non-Task Logging and Recovery environment relieves excessive requirements on the production PDM.

5

Tuning the CICS connector

To improve performance for CICS SUPRA, you can adjust the parameters of the OPER CONNECT command, which connects the CICS Connector to the Physical Data Manager (PDM). You use the CICS Connector to enable PDM access for both RDM and non-RDM online SUPRA tasks. For details on using the CICS Connector, refer to the *SUPRA Server PDM CICS Connector Systems Programming Guide (OS/390 & VSE)*, P26-7452.

Tuning OPER CONNECT parameters

The THREADS, TASKS, MEME, TFUL, CFUL, and ICOR values of the CONNECT command affect the performance of the CICS Connector. The default values are specified in the CSTXOPRM macro. You can override the defaults with the OPER CONNECT command. The Connector TASKS and THREADS values cannot exceed the PDM maximum tasks and threads values. See the illustration under “[Defining tasks](#)” on page 90.

To determine the appropriate balance of these values, monitor the CICS system under its actual load and examine the data. Use output from the OPER STATUS command to monitor performance. Also, consider the following factors:

- ◆ **Memory resource usage.** To save memory, you can decrease the TASKS parameter (see “[TASKS parameter](#)” on page 84).
- ◆ **Number of users.** You may want to increase or decrease the TASKS parameter to accommodate the number of users who need to be signed on to the PDM concurrently (see “[TASKS parameter](#)” on page 84). To accommodate more users during peak periods, you can also increase the CFUL retry parameter (see “[Retry parameters](#)” on page 86).
- ◆ **TFUL WAITS message.** If the OPER STATUS statistics are showing TFUL WAITS, you may want to increase your number of threads up to your CICS MAXTASKS or lower the CICS MAXTASKS parameter (MXT).

VSE

The CICS Connector no longer uses the CSTXOPRM macro MAXPACK parameter for calculating the space needed during CONNECT. You can use it merely for documentation purposes.

The following sections explain in detail how to adjust each OPER CONNECT parameter for optimum tuning. If you are using the CICS Connector, you are most likely also using RDM. See “[Tuning RDM](#)” on page 89 for information about tuning the CICS Connector parameters with RDM.

THREADS parameter

The CICS Connector allows you to adjust its number of concurrent threads for PDM requests. You use the THREADS parameter in the OPER CONNECT command.

You can effectively use threads and storage by considering the relationship of tasks to the Maximum Tasks Parameter (MXT) on the CICS System Initialization Table (DFHSIT). MXT specifies the actual maximum number of tasks that can be concurrently dispatched from the active queue. Use the following formula:

$$\text{THREADS} = (\% \text{ of tasks using SUPRA}) * \text{MXT}$$

To balance SUPRA usage in a system with a large mixture of different transaction types, you can also consider the Class Maximum Tasks Parameter (CMXT) in the System Initialization Table. This parameter specifies the maximum tasks of a given class allowed as part of the active queue (MXT). Assign a class for your SUPRA Server transactions and use the following formula:

$$\text{THREADS} = \text{CMXT for SUPRA class}$$

If task responses are poor, enough threads may not be available between the CICS Connector and the PDM to process all of the commands being issued. TFUL status codes indicate this condition. Increase the value of the THREADS parameter (up to the PDM maximum threads value, but not greater than the Connector TASKS) in the CONNECT command. Alternatively, you can decrease the MXT value in the CICS System Initialization Table. However, decreasing the MXT value could also adversely affect response time.

TASKS parameter

The OPER CONNECT TASKS parameter establishes the maximum number of CICS tasks that can be signed on to the PDM. TASKS determines the number of entries in the signed-on table (SOT). If you are receiving poor response for tasks, you may not have enough entries available. CFUL status codes indicate this condition.

Evaluate usage of the system. If there is a peak period when a maximum number of users want to access the PDM, you can increase the TASKS parameter (up to the PDM maximum tasks value) to accommodate those users' tasks. However, if you set this parameter too high for general use, memory could be wasted. If usage is generally stable except for certain peak periods, you can set the TASKS parameter for the average number of users and increase the CFUL RETRY parameter (see [“Retry parameters”](#) on page 86).

MEME parameter

The OPER CONNECT MEME parameter places a limit on the number of 65,504-byte blocks the CICS Connector can obtain before DISCONNECT. In prior releases, the Connector obtained memory only at connect time, according to your estimate of probable needs (MEMORY parameter and VSE MAXPACK parameter). With releases 2.1.6 and higher of SUPRA Server, the Connector dynamically obtains memory only as needed.

A request for memory causes the Connector to acquire one 65K block (the first set of requests occur at CONNECT). Subsequent requests use areas of those blocks until they are filled and another is acquired. Requests might be from DBAID, the PDM interface, or RDM. The memory is used for the Connector's hash table, TASKS, THREADS, MAXPACKET, and miscellaneous storage. See "[Memory requirements for the CICS Connector](#)" on page 142 for a formula.



This parameter does not include the dynamic 5K per task that occurs at task attach and log-on to PDM (see "[Memory requirements for the CICS Connector](#)" on page 142).

You can set this parameter to reflect the highest or standard production needs in the CSTXOPRM macro and set it higher or lower with the CONNECT command.

Retry parameters

Use the OPER CONNECT retry parameters to reduce the number of task abends during peak usage times. These parameters specify the maximum number of retries if the interface encounters the following status codes:

- ◆ CFUL parameter for CFUL status—No task SOT entries are available
- ◆ TFUL parameter for TFUL status—No threads are available
- ◆ ICOR parameter for ICOR status—Not enough memory is available



ICOR problems indicate that you need to perform additional tuning to balance the memory load in your system. You may need to adjust the region, partition, or address space size. Also see the tuning information for the various SUPRA Server components.

If a high number of your tasks fail, you can increase resources: TASKS (see “[TASKS parameter](#)” on page 84) or THREADS (see “[THREADS parameter](#)” on page 83). However, if you set these resources for maximum usage levels, you waste resources during average usage periods and improve response time only during peak usage periods.

The OPER CONNECT retry parameters allow you to use your resources more effectively. Set tasks and threads for your average level of usage. At peak times, tasks receiving a TFUL or CFUL status are automatically retried. You can raise the OPER CONNECT retry parameters to increase the rate of successful transactions. However, this can also increase response time, but only for those tasks waiting for the retry on CFUL. If you have a THREADS value that is too small and a TFUL retry value that is high, you will experience a general decrease in response time.

Turning off tracing and Sync for performance

CICS Connector tracing and RDM tracing use CPU time and I/O. Use them only in test systems or to help debug a specific production problem. Turn them off for normal production. The sample CSTXOTBL distributed with SUPRA has CICS Connector tracing turned off, and RDM is distributed with tracing disabled.

If you want to activate CICS Connector tracing, specify TRACE1=trace-id (TRACE1=5) in the CSTXOPRM macro. The other two TRACE parameters automatically turn on when TRACE1 is on. If TRACE1 is a 5, TRACE2 is a 6 and TRACE3 is a 7. Refer to the *SUPRA Server PDM CICS Connector Systems Programming Guide (OS/390 & VSE)*, P26-7452, for the macro.



Cincom Support may request that you turn on CICS and/or RDM tracing at your site.

Cincom tracing does not use CICS global exits as in prior releases. It now uses CICS Task Related User Exit (T.R.U.E.) routines. They are active whether tracing is on or off. When Cincom product tracing is off, the CICS Connector issues the following calls:

- ◆ For an RDML request, a DFHRMCAL to the T.R.U.E. that manages the RDM resource.
- ◆ For a PDML request (generated by an RDML request or a PDML task), a DFHRMCAL to the T.R.U.E. that manages the PDM MTMT interface resource.
- ◆ During PDM processing of the PDML, an EXEC CICS WAIT,EVENT,ECADDR.

For each DFHRMCAL above, CICS creates approximately four TRACE entries. They do not affect response time, but they do affect CPU usage. You can modify these entries with the CETR transaction. You can eliminate these entries by using the DFHSIT parameter INTTR, or more specifically, parameter STNTRUE (UE refers to the CICS USER EXIT component).

When CICS Connector tracing is on, the exit routine creates trace entries at two points with an EXEC CICS ENTER,TRACENUM command. One is when PDML enters the CICS Connector's T.R.U.E., and one is when it exits. When RDM tracing is on, the RDM resource T.R.U.E. routine creates several trace entries for each RDML request. Other trace entries occur at some points (not at each RDML or PDML), but these have little effect on CPU usage. You can use the DFHSIT USERTR parameter override to limit tracing for a particular CICS execution. The DFHRMCALs still cause CICS to generate entries.

Synchronization also uses valuable resources. Code SYNC as N on CSTXOPRM (or let default to N) unless you have a real need to maintain SUPRA updates in synchronization with CICS protected resource updates, such as a VSAM file.

6

Tuning RDM

You can improve RDM performance with the methods discussed in the following sections. This includes installation tips, task, and task storage definition with the RDM C\$VOOPTM macro parameters, view design, and placement, and tuning applications.

Installing RDM

Part of tuning RDM concerns how you install the modules. OS/390/XA and OS/390/ESA sites can install a part of RDM in the Link Pack Area or cause it to be loaded above the 16 MB line at execution time. VSE sites can install a part of RDM in the Shared Virtual Area (SVA). This conserves main memory and reduces the paging rate. See “[Resident Program requirements for SUPRA Server in a CICS environment](#)” on page 143 for sizes.

For further installation information, refer to the *[SUPRA Server PDM RDM Administration Guide \(OS/390 & VSE\)](#)*, P26-8220.

Defining tasks

The number of RDM tasks allowed depends on many factors. To begin with, the active environment description determines how many tasks and threads the PDM allows from all CICS systems and all batch jobs. Each CICS Connector defines the number of tasks and threads from that CICS system to the PDM. These can be PDML applications, Directory Maintenance users, and RDM users. RDM users include MANTIS, SPECTRA, DBAID, and precompiled applications. RDM determines how many users it can sign on with the RDMUSR# parameter of the C\$VOOPTM macro.

RDM response is poor if RDM, the CICS Connector, or the PDM allow too few tasks for the number of RDM users. For the CICS Connector, ensure that the value of the THREADS and TASKS parameters for the CONNECT command are large enough (see “[THREADS parameter](#)” on page 83 and “[TASKS parameter](#)” on page 84). For the PDM, ensure that enough PDM tasks and threads are defined in the environment description on the Directory for all CICS systems and all batch jobs. Also see “[Tuning the number of concurrent PDM tasks](#)” on page 63 about concurrent PDM tasks.

The CICS MXT value is a factor in tuning the various TASKS and THREADS parameters. Much of your tuning depends on whether your CICS system is strictly for RDM transactions, is available to other SUPRA transactions, or is also available to non-SUPRA transactions.

Another factor is that RDM uses the HEAP# parameter of the C\$VOOPTM macro to determine how many task heaps to acquire for the number of RDM users (RDMUSR#). You usually make RDMUSR# and HEAP# the same number. Having HEAP# greater than RDMUSR# is a waste of memory. You can make HEAP# less than RDMUSR#, usually when you have pseudoconversational tasks. When less, any pseudoconversational task heaps are subject to RDM rollout to CICS Temporary Storage. This can occur when they are inactive and RDM needs the heap for another task.

You set these parameters to accommodate your number and mix of transactions. The following figure illustrates the concepts and relationships of this task discussion. Use it to help you determine the optimum settings for the PDM, CICS, CICS Connector, and RDM at your site.

For a complete discussion of the interaction of RDM parameters with other SUPRA and CICS parameters, refer to the *SUPRA Server PDM RDM Administration Guide (OS/390 & VSE)*, P26-8220. Also refer to that guide for the C\$VOOPTM macro description.

<p>Central PDM</p> <p>The CSIPARM file names the REALM Environment Description</p>	<p>Batch job (Batch Dir. Maint.)</p> <p>1 interface 1 thread 1 task</p>		
	<p>Batch job (Batch RDM)</p> <p>1 interface 1 thread 1 task</p>		
	<p>Batch job (PDML Appl.)</p> <p>1 interface 1 thread 1 task</p>		
<p>Directory</p> <p>Environment Description</p>	<p>CICS Connector</p> <p>1 interface</p>		<p>CICS System</p> <p>THREADS = 3 TASKS = 10</p>
	<p>CICS Connector</p> <p>1 interface</p>		<p>CICS System</p> <p>THREADS = 3 TASKS = 10</p>
		<p>Max. Connected Interfaces = 6</p>	<p>Max. Connected Threads = 9</p>

An interface cannot connect if the PDM does not have sufficient unassigned resources. In this figure, another batch or CICS job cannot connect because all TASKS and THREADS are reserved, even though an Interface is available.

CICS Connector

CSTXOPRM macro
 CSTXOTBL module
 OPER CONNECT command

← THREADS - Simultaneous in-flight PDML.
 - TFUL when exceeded.
 - Equally available to all SUPRA tasks.
 (PDML Appls., RDM signed-on tasks,
 Directory Maintenance tasks).

← TASKS - Simultaneous signed-on tasks.
 - CFUL when exceeded.
 - Equally available to all SUPRA tasks.
 (PDML Appls., RDM signed-on tasks,
 Directory Maintenance tasks).

CICS System, SIT

MXT - Initiated CICS tasks
 (attached).

AMXT - Active CICS tasks
 (dispatchable).
 - Include the Connector
 THREADS plus any
 non-SUPRA activity.

<p>PDML Application Always 1 task; 1 thread when needed</p>
<p>Directory Maintenance Always 1 task; 1 thread when needed</p>

<p>RDM (RDML Appls., DBAID, SPECTRA) C\$VOOPTM macro CSVOOPTM module</p>	
RDMUSR#	<ul style="list-style-type: none"> - Simultaneous signed-on tasks. Each uses a Connector task and a PDM task for sign-on duration. - INSUFFICIENT RESOURCES when exceeded. - If greater than TASKS, excess receive CFUL.
HEAP#	<ul style="list-style-type: none"> - Opened view storage for each RDM user. - Assigned at RDML sign-on, remains until sign-off. - If greater than RDMUSR#, wasted space. - When RDML/PDML in-flight, uses a thread. - Same number as RDMUSR# if you have sufficient space; can be less when psuedoconversational tasks (lowest at least THREADS). - If pseudoconversational, RDM can roll out the heap when the task detaches, to be assigned to a resuming task or a new sign-on. - INSUFFICIENT RESOURCES if no heap is available at sign-on.

A thread uses more resource than a task log-on area. THREADS need never be as high as tasks, even if all tasks are conversational.

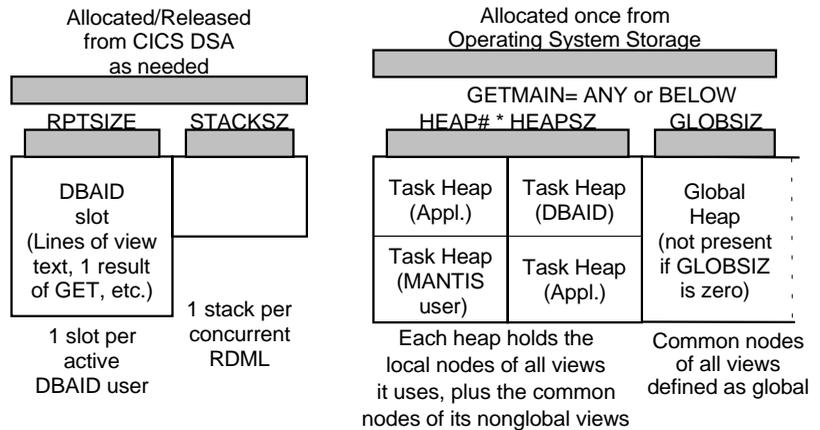
Defining task and global view storage

Online RDM is distributed with arbitrary values for the storage it will acquire for task- and global-view context areas. You tune these values for your test and production systems by using the RDM C\$VOOPTM macro. The following figure shows the C\$VOOPTM parameters involved with storage, and the areas they produce.

Batch RDM does not use the C\$VOOPTM macro. Batch RDM acquires a fixed task-memory amount of 512K (variable via patch obtained from Cincom Support) and does not use global views. Batch DBAID acquires a fixed slot-memory amount of 512K (variable via patch obtained from Cincom Support).

As shown in the following figure, each DBAID task that signs on allocates a CICS DSA RPTSIZE slot that remains until the user signs off; then DBAID releases it. (A DBAID task also uses a stack and a task heap during its session.) For each RDML command from any task, RDM allocates a stack in CICS DSA and releases it upon command completion.

The C\$VOOPTM macro definition of RDM dynamic storage areas:



RDM allocates the task heaps and the global heap at initialization, and they remain allocated throughout RDM execution. RDM cannot initialize if there is not enough GETMAIN space (OS/390) or GETVIS space (VSE) according to the parameters. For OS/390, if you have the RDM Extended Storage option, you can allocate this storage above the 16 MB line by using GETMAIN=ANY in the C\$VOOPTM macro. Otherwise, the storage comes from OSCOR (CICS 2.1 and lower) or from the available storage determined by CICS region size minus DSASZE in SIT (CICS above release 2.1). In VSE, the storage is taken from the GETVIS area of the CICS system using RDM.

After successful initialization, every task that signs on to RDM is assigned an available task heap (see “[Defining tasks](#)” on page 90 about number of tasks and heaps). At task sign-off, the heap is available to be reassigned.

For a conversational task, the heap remains assigned until the task signs off (DBAID is conversational). For a pseudoconversational task, the heap is subject to rollout/rollin to/from CICS Temporary Storage if there are fewer heaps than RDM users. (SPECTRA tasks are pseudoconversational; MANTIS tasks could be either.) The C\$VOOPTM macro contains parameters for coordinating your heap size, the rollout increment, and the CICS Temporary Storage file size.

RDM uses the task heaps and global heap to contain information about the views the task is using. The illustration at the end of this section shows this usage.

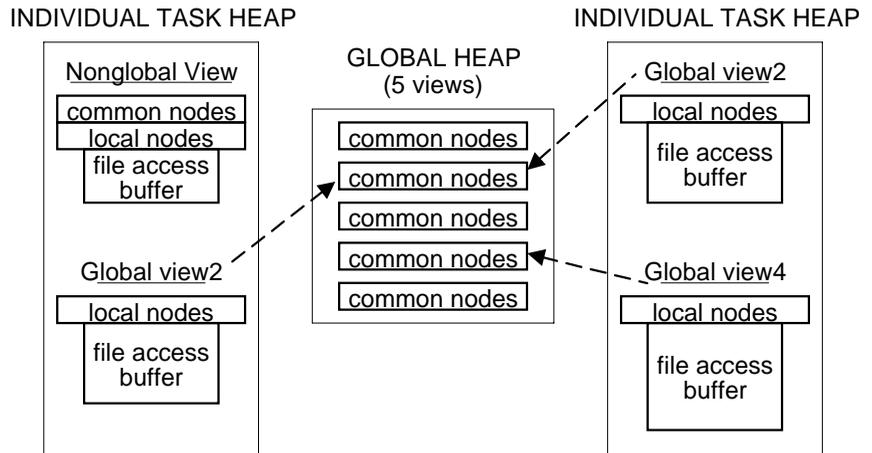
The sets of common nodes and local nodes in the figure represent open views. RDM allocates them when the view is opened, and they remain until a view is released or the task signs off. A node contains part of the internal representation of an open view. Common nodes contain the information that does not change from RDML to RDML (e.g, file names, external field names). Local nodes contain the information that does change with each RDML. For example, ASI returns and positions references within accessed files. The common and local nodes for one view have the same view-name unless you used the User Name option. If so, the user-name is assigned to the local nodes. You specify the User Name option in an application with INCLUDE user-name=view-name or in DBAID with OPEN user-name=view-name. See “[Determining view size](#)” on page 98 and “[Tuning DBAID and applications](#)” on page 102 for more about nodes.

The file access buffer contains a record from each file named in the view's access definition. If a file is named twice in the access definition, there are two records from it in the file access buffer.

The individual task heaps are all the same size (see the illustration under "Defining task and global view storage" on page 93) although each task may need different amounts (RDM requires a fixed amount of 8K in each task heap). A target of 64–96K heap size is reasonable for many production applications. You need to experiment with different values for HEAP# and HEAPSZ to determine the optimum values for your environment.

The heap size must be large enough to hold the nodes and file access buffers for the application (or DBAID session) that simultaneously uses the most and the largest views (global and nonglobal). If the heap size becomes too small for a task's view needs, a 2816 task abend occurs. Determining view size is discussed in "Determining view size" on page 98. Tuning DBAID and application usage of views is presented in "Tuning DBAID and applications" on page 102.

If you change the C\$VOOPTM HEAPSZ (or any parameter) for online RDM, refer to the *SUPRA Server PDM CICS Connector Systems Programming Guide (OS/390 & VSE)*, P26-7452, to incorporate the changed module. Any changed counts or sizes, or a change to the GETMAIN option, impact RDM memory allocations.



Defining efficient views

You use DBAID to experiment, define, and save views. You can also use Directory Maintenance to define views directly but DBAID is recommended.

Defining data and navigation

Design views in such a way as to minimize the number of views, RDML calls, and database navigation required for each application. Design views and their navigation as simply as possible. Include only the data the application is actively working on and a minimum of overhead data that must be accessed to maintain integrity.

In general, you should have specialized views for each application. It is better to have many small- or medium-sized specialized views rather than one large view that all applications use. The larger the view, the more heap storage is used. The space needed on the Directory for many views is more efficient than having a large view that requires more heap storage for opening by applications. However, a specialized view should access as many of an application's files as possible. The view should include only those fields from the files that are necessary for navigation and program logic.

Navigation defines how to get from file to file within a logical view using an access definition. You can use three classes of navigation: direct keyed, sweep, and secondary key (on an index file). You can write the access definition in the following ways:

- ◆ It defines the physical navigation through your files.
- ◆ It uses the generalized form so that RDM determines the navigation.

When you use the generalized form, ensure that RDM is not scanning the file when you do not want it scanned. You can use the DBAID SHOW-NAVIGATION command to see how RDM is accessing the files in a view. If you are using RDM DEBUG, inspect the output for navigation paths. For more information on these tools, refer to the *SUPRA Server PDM RDM Administration Guide (OS/390 & VSE)*, P26-8220.

Use RDM statistics to determine the efficiency of file access on a particular view. Check the access definition and the keys and files used. Also check any external fields you used. If you have defined an external field as a unique key in your view, it must be a unique key in the PDM. If not, whenever that view is used to insert records, an extra check is done on the database to verify that the added records are unique. In the case of related file linkpaths, response time increases in proportion to the number of records in the linkpath chain.

You should use unique physical keys for files in a view if possible. For example, if you define a view using a single, uniquely keyed file, and use a data field instead of the control key as a logical key, RDM serially scans the file to retrieve your data.

When defining an access definition with a control key having multiple fields (compound key), you must enclose all of those fields within parentheses, in the order defined in the Directory. If not, RDM does not use the control key and sweeps the file to retrieve your data.

It is more efficient for RDM to search a file than it is for an application program to do the search. Therefore, if your applications need to search for records on values that are not physical keys, you should define those values as logical keys and let RDM do the searching.

The efficient navigation methods to retrieve records from a keyed file involve using a logical key that maps to a physical key in the record (control key or secondary key). If the fields in a WHERE clause in an access definition map to either a physical key, RDM accesses the records directly; otherwise, RDM sweeps the file.

Determining view size

View size is important for determining how large to make the task heap size (see “[Defining task and global view storage](#)” on page 93). You can use DBAID and RDM reports to find the size of new and existing views, whether or not they are global.

To use RDM DBA reports, execute the RDM Reporter utility DBA Report. This shows the total bytes to open, including common and local nodes and file access buffers.

To use DBAID, issue 2 DBAID OPENS to determine the memory requirement of each view. Open the same view twice with different user names as follows (when you do not specify a user-name, it is the same as view-name):

```
(user-name) (view-name)
> OPEN          CSI-SP-ATTRIBUTE
FSI: *  VSI: =   MSG:          8000 BYTES USED IN OPENING VIEW
> OPEN ELEMENTS=CSI-SP-ATTRIBUTE
FSI: *  VSI: =   MSG:          2168 BYTES USED IN OPENING VIEW
```

This is a multiple open of the view because there are two user names for the same view name. DBAID always generates a message for an open that indicates the memory requirement for the common and local nodes (see the illustration at the end of “[Defining task and global view storage](#)” beginning on page 93) of that open. (When you use this method, add 5 percent to each of these figures because the calculations for the message do not account for double word alignment.)

In the example, the common nodes are shared by the second open and do not require further memory. The arithmetic difference (5832 bytes) is the one-time amount needed for the common nodes. The memory required for the local nodes and file access buffer is the second figure (2168 bytes). There are two occurrences of the local nodes in this multiple open (see “[Tuning DBAID and applications](#)” on page 102 for more on this subject).

If the view is global, the common node amount goes into the global heap once, and the local node amounts go into the task heap of each task that opens the view.

Making views bound

View binding improves performance on the initial access to a view (RDM opening of the view on initialization or task request). When you bind a view, RDM stores the open view on the Directory as an internal form of the view's common nodes. The stored item is referred to as "a view binding" or simply "a binding." A binding greatly reduces processing time for RDM view opening during application or DBAID execution. Bound views are most useful in a production environment where a view is unlikely to change frequently. Do not bind a view you are developing. Binding a view has no effect on memory requirements.

You bind views by issuing the DBAID BIND command or DBAID SAVE command with the BIND option. If a view is bound, RDM uses the binding for an application, but the text of the view is still available to the DBA for maintenance. RDM automatically chooses bound views if found at run time, even if the view text or files have been changed since the view was most recently bound. This allows you to test newer versions with DBAID while applications use the bound copy. If the newer version is to be used in production, you must rebind the view.

Any time you change a view, you must bind it again if it was previously bound. Binding a view requires that RDM parse the view text to open it, and RDM reads a number of metadata items during this process. Some of the metadata items may not be part of the view itself, so if you change any of them during other maintenance, you may need to rebind one or more affected views. You do not always know when a view is affected. Therefore, it is a good practice to periodically rebind all bound views to ensure that all bindings are current with the state of your database. Use the DBAID command BIND BOUND to rebind bound views. Also use this command after Cincom product maintenance procedures.



Use bound views that are also global if possible. Global bound views allow faster RDM initialization (see ["Making views global"](#) on page 100).

Making views global

You can improve performance and reduce the size needed for task heaps by making views global. Global views exist only for CICS RDM (not batch RDM). CICS RDM opens global views only once, during RDM initialization. Global views remain in the global heap (see the illustrations under “[Defining task and global view storage](#)” on page 93), available to all valid users for the duration of RDM execution. These global views remain opened, even if individual tasks release them. Making a view global does not change its overall size. Views used by multiple applications or by multiple users of a single application should be made global.

When you make a view global, the following improvements occur:

- ◆ RDM reads the view definition from the Directory at initialization, rather than each time an application causes a view open (see “[Tuning DBAID and applications](#)” on page 102). Since some I/O is shifted to initialization time, initialization becomes slower. However, total I/O may be reduced; the more times applications open this view, the more I/O is saved.
- ◆ RDM constructs the view common nodes at initialization, rather than each time an application causes a view open. Like I/O, since some CPU use is shifted to initialization time, initialization becomes slower. Total CPU use may be reduced; the more times applications open this view, the more CPU use is saved.
- ◆ RDM builds the view common nodes only once in the global heap rather than multiple times in application task heaps (see “[Defining task and global view storage](#)” on page 93). Some memory consumption is shifted from the task heaps to the global heap. A global view may increase region-size requirements for the online job. However, if you define commonly used views as global, you can then decrease task-heap size. If you have a large number of task heaps, the saving in task heap memory is usually greater than the increase in global heap size (see the illustration at the end of “[Defining task and global view storage](#),” which begins on page 93).

- ◆ If you can place all production views in the global heap, you can significantly reduce the task-heap size and use the gained memory to increase the number of task heaps. If you have the RDM Extended Storage option, you can place the global heap (and the task heaps) above the 16 MB line (see the illustration under “[Defining task and global view storage](#)” on page 93). If you cannot place all production views in the global heap, define at least the largest and most frequently used views as global views.
- ◆ You define global views by using long text records associated with your environment description on the Directory. These constitute the global view list. New entries do not take effect until you shut down and reinitialize RDM. You can use the CICS Connector OPER commands to effect changes immediately. Refer to the [SUPRA Server PDM CICS Connector Systems Programming Guide \(OS/390 & VSE\)](#), P26-7452.
- ◆ Global views are most useful in a production environment where the global view list is not likely to change frequently. Do not use global views when you are developing views.

For details on defining global views, refer to the [SUPRA Server PDM RDM Administration Guide \(OS/390 & VSE\)](#), P26-8220, and the [SUPRA Server PDM Directory Online User's Guide \(OS/390 & VSE\)](#), P26-1260.

Tuning DBAID and applications

You can use some application programming and DBAID methods to improve performance and reduce memory usage and program size. Understanding how RDM uses the task heaps and view opening is necessary for tuning.

See the illustration at the end of “[Defining task and global view storage](#),” which begins on page 93 about common and local nodes in conjunction with the following discussion. Repeating the DBAID example in “[Determining view size](#)” on page 98 that demonstrates a “multiple open”:

```
(user-name) (view-name)
> OPEN          CSI-SP-ATTRIBUTE
FSI: *   VSI: =   MSG:          8000 BYTES USED IN OPENING VIEW
> OPEN ELEMENTS=CSI-SP-ATTRIBUTE
FSI: *   VSI: =   MSG:          2168 BYTES USED IN OPENING VIEW
```

With this multiple open, the DBAID task heap now has one occurrence of the common nodes and two occurrences of the local nodes (if the view is global, the common nodes are in the global heap instead of the task heap):

Common	Local
CSI-SP-ATTRIBUTE, 5832 bytes	CSI-SP-ATTRIBUTE, 2168 bytes
	ELEMENTS, 2168 bytes

However, if you use a “reopen,” RDM releases the local nodes and then reallocates them, so they occur only once. A reopen happens when you open a view with the same view name and user view name. Use one of the following sequences to perform a reopen of a view.

A reopen with a user-name specified:

```
(user-name) (view-name)
> OPEN ELEMENTS=CSI-SP-ATTRIBUTE
FSI: * VSI: = MSG:          8000 BYTES USED IN OPENING VIEW
> OPEN ELEMENTS=CSI-SP-ATTRIBUTE
FSI: * VSI: = MSG:          8000 BYTES USED IN OPENING VIEW
```

Common	Local
ELEMENTS, 5832 bytes	ELEMENTS, 2168 bytes

or, a reopen with user-name being the same as view-name:

```
(view-name)
> OPEN CSI-SP-ATTRIBUTE
FSI: * VSI: = MSG:          8000 BYTES USED IN OPENING VIEW
> OPEN CSI-SP-ATTRIBUTE
FSI: * VSI: = MSG:          8000 BYTES USED IN OPENING VIEW
```

Common	Local
CSI-SP-ATTRIBUTE, 5832 bytes	CSI-SP-ATTRIBUTE, 2168 bytes

Either set of the above DBAID statements reopens the view. The common nodes occur once as in a multiple open (either in the task heap or in the global heap if the view is global). Unlike a multiple open, with a reopen the task heap contains only one occurrence of the local nodes.

User applications do not have an OPEN command, but an application can do multiple opens or reopens. An application accomplishes a multiple open by using 2 INCLUDEs for the same view with differing user-names. An application accomplishes a reopen by using 2 INCLUDEs with the same user-view names (or omitting both times), or by using a RELEASE for a view and later using a GET for the same view. This might be in 2 executions of a pseudoconversational program. Pseudoconversational tasks must always use the same view-naming convention (for a particular view) throughout the application.

You should try to keep processing and task-heap usage low by opening a view only once per task or pseudoconversation, and releasing it when it is no longer needed. The RELEASE command is available in DBAID and in applications. This releases space in the task heap to open other views.

Subsetting views

Subsetting a view allows the task to use less program data area. You subset a view by specifying a column list after the view name:

```

Program {
    INCLUDE view-name (column, column, column, column)
    or
    INCLUDE user-view-name=view-name (column, column, column, column)
DBAID {
    OPEN view-name column, column, column, column
    or
    OPEN user-view-name=view-name column, column, column, column

```

Be careful when subsetting views. Subsetting a view can cause a projection on the view where the data for fast PDM access is not available, causing extended PDM activity. You can subset view attributes including keys in your applications. This may cause RDM to generate a nonkeyed access where a keyed access is possible. Never subset intermediate parts of physical keys in compound keyed views.

Subsetting views can eliminate unnecessary accesses. However, you should consider the overhead when defining subset views. For example, if a view accesses 10 files, but you need attributes from only 1 file, subsetting definitely saves access time. If more than half the files in a view are accessed, subsetting the view would not save access time, but it would save program storage space. You should define new views on the Directory for some of these situations, rather than subsetting.

Using views

Use global views when possible (see “[Making views global](#)” on page 100). Check to see if you have global views defined in your environment descriptions. Studies prove that using nonglobal views causes substantial CPU overhead each time a nonglobal view is opened.

Make sure your logical views are efficient and used. Check your high-usage programs to determine if all included views are actually used. Eliminate any unused views.

To gain a slight performance improvement, applications can include the least-used views first and the most-used last.

In some cases, a view used in the main program or in an external routine is a subset of a view also used in the same application program. Combine these views into one and remove the superfluous view.

If you use global views and tasks use the `RELEASE` command with specified view names (selective release), you can also decrease task heap size, which reduces the time for heap rolling for pseudoconversational tasks. An application can release an individual view without releasing all other views. Views used only occasionally in an application can be released immediately after their last use. This frees storage for the view without losing position or marks for other views.

Releasing one or several views does not reduce the size of a task heap although it does reduce the space used in the heap. The purpose of carefully using the `RELEASE` command is perhaps to allow a lower, instantaneous demand for allocated nodes and file-access buffers within a heap. This allows the task to open other views. If all tasks use it wisely, this could allow you to reduce the `HEAPSZ` parameter allowance for all heaps.

Using the online DBAID slot (RPTSIZE)

Use the RPTSIZE parameter in the C\$VOOPTM macro to define the slot storage RDM acquires for each online DBAID session. Batch DBAID does not use the C\$VOOPTM macro. The batch DBAID slot size is a fixed 256K (variable via patch obtainable from Cincom Support).

A DBAID slot holds the statement lines of views you listed or defined (the task heap holds the view nodes). See “[Defining task and global view storage](#)” on page 93 about online DBAID slot storage allocation and duration.

If the slot is too small, inadequate space is available to service DBAID requests, which results in a 2808 or a 2816 abend. Set the RPTSIZE value to accommodate the size required for the DBAID session that will use the most lines of view listings. When you increase RPTSIZE, you increase the total storage allocated to RDM DBAID sessions, and you decrease the total storage available to the rest of CICS.

To gain space when needed *during* a DBAID session, UNDEFINE any views you no longer need in the DBAID session (after saving, if desired). This frees up slot space.

A

Memory use and space calculations

This appendix provides memory requirements for SUPRA Server components and shows the calculations used to determine those requirements. The components are the **Physical Data Manager (PDM)**, **VSAM files**, **Directory Maintenance**, the **CICS Connector**, **SUPRA in a CICS environment**, and **index files**.

These values are intended only as guidelines. Every site has different variables and requirements affecting these values.

MEMORY parameter calculation for the PDM

The PDM needs to acquire storage from the operating system's available space during bootstrap initialization and during realm initialization. The PDM uses this memory for various control blocks. The MEMORY parameters in your boot environment description and your environment descriptions on the Directory tell the PDM how much storage to acquire (memory pool).

The distributed boot environment descriptions provide a sufficient amount for initial installation. You need to estimate or calculate the amount you need for the boot environment descriptions as your site grows, and for your various user environment descriptions. The following tables and notes detail how to calculate the amount of space. Subsequent sections show sample values derived from the calculations.

PDM system memory requirements

Entity	Size as dec (hex)	Number of entities	Additional explanation
Interface	4944(1350)	Max. connected interfaces value	The environment description parameters control the PDM system requirements, except for MAXPACKET below.
Thread	4136(1028)	Max. connected threads value	
Task	304(130)	Max. signed-on tasks value + 1	See "Notes" following the table about CICS Connector tasks.
Record holding entry	96(60)	Maximum held records value	
Shared records read entry	32(20)	Max. signed on tasks value * number of files (minimum of 2)	
Wait management entry	36(24)	Number of interfaces + number of threads + 10	
MAXPACKET	Default or CSIPARM user value	One	This pertains to central environments other than OS/390.

File memory requirements

(See [Notes](#) following these tables for system file sizes.)

Entity	Size as dec (hex)	Number of entities	Additional explanation
File	1064(428) + ACB/DCB/DTF length	One per file	Repeat the remaining file calculations for each file.
Record code	106(6A)	Number of record codes in the file + 1	
Data item or linkpath	32(20)	Number of data items and linkpaths in the file + 2	The PDM needs 1 extra for data items and 1 extra for linkpaths.
Secondary key	352(160)	Number of secondary keys in the file	
Secondary key code	48(30)	One per key code in the file	
Secondary key element	108(6C)	Number of elements in the secondary key	
File experience table (bit map). Non-KSDS files only.	0 if disabled. Otherwise, varies by file size.	One per non-KSDS file	Enable/disable for all files with CSIPARM PRIMAP or RELMAP. The formula is: $((TLR+1) / Recs-per-block) / 8) + 25$.

Buffer memory requirements

Entity	Size as dec (hex)	Number of entities	Additional explanation
Buffer pool	192(C0)	One per buffer pool	Repeat the remaining buffer calculations for each pool. KSDS buffers are not allocated from this memory pool (see "VSAM control-block allocation" on page 133)
BDAM, BSAM, ESDS buffer control	704(2C0)	$DB + (ST * SB)$	From the Buffer parameter of the environment description. DB= Direct buffers ST = Serial threads SB = Serial buffers
BDAM, BSAM, ESDS Direct buffer	Largest block or control interval size among files sharing this pool	$DB + (ST * SB)$	
BDAM, BSAM, ESDS block control entry	48(30)	$DB + (ST * SB) +$ Number of threads + Number of interfaces + 5	
BDAM, ESDS serial thread and serial buffer	Serial threads and buffers are not used as such. If specified for the buffer pool, they are converted to Direct buffers (the last 3 entries above).		
VSAM file management	OS/390, 24(18) VSE, 98(62)	One per VSAM file	In addition, RPL blocks are obtained. See Notes following these tables about RPLs.
Log synch entry	32(20)	$((TLF \text{ buffers} * \text{blksz}) / 200) + ((SLF \text{ buffers} * \text{blksz}) / 200) + 28$	TLF buffers = buffers allocated for the Task Log. SLF buffers = sum of buffers allocated for all system log groups.

User-variable memory requirements

Entity	Size as dec (hex)	Number of entities	Additional explanation
ACTV entry	272(110) + control key length	One per file * tasks using the file	If 2 tasks use 1 file, there are 2 ACTV entries; if A tasks use B files, there are A*B=C ACTV entries. The PDM releases all entries for one task either when it issues a commit and the environment description's threshold value is exceeded, or when the task signs off.
PDML READX		One per each unfinished command	The PDM allocates 1 per initiated command. It remains allocated until RDM or the PDML application issues the same command with a qualifier of END., until end of index file is reached, or until the task signs off.
PDML FINDX and RDNXT (KSDS only)	32(20) + control key length	One per each unfinished command	The PDM allocates 1 per initiated command. It remains allocated until RDM or the PDML application issues the same command with a qualifier of ENDS, until end of file is reached, or until the task signs off.
KSCB for KSDS RPL control on FINDX, RDNXT	24(18) + VSAM RPL length	One per string per each unfinished command	The PDM allocates these per initiated command. They remain allocated until RDM or the PDML application issues the same command with a qualifier of ENDS, until end of file is reached, or until the task signs off.
Bound data list	Varies by longest data list length	Number of unique lists in all applications	The PDM releases each unique data list when the last application using it signs off.
Command-related requirements	Varies by command type and affected file(s)	Number of concurrent commands (threads maximum)	This memory is acquired and released with each command. It pertains to any PDML command. See Notes following the table about calculations.

Notes on the preceding tables

- ◆ An example for calculating the File Memory Requirements category is as follows (this does not include the buffer or other categories). This file has 4 elements, 2 linkpaths, the base record code, and no secondary keys:

Entity	Size	Number	Totals
File	1064(428) + ACB/DCB/DTF	1	1064(428) + ACB/DCB/DTF
Record code	106(6A)	1	106(6A)
Data items and linkpaths	32(20)	6+2=8	256(100) 1426(592) + ACB/DCB/DTF

- ◆ The File Memory Requirements category values for the distributed boot module files are as follows (this does not include the buffer or other categories).

Def. Num. (DEF#)	19560	(4C68)
Name (NAME)	1490	(5D2)
Text (TEXT)	1830	(726)
Structure (STRU)	1862	(746)
Data (DATA)	7038	(187E)
Task Log (TLOG)	1076	(434)
System Log (SLOG)	1076	(434)
Statistics Log (STAT)	1076	(434)
Total Bootmod size	35008	(88C0)

- ◆ You should add 2%–3% to your final memory sum. This allows for memory fragmentation and minor dynamic allocation for command processing.
- ◆ Each CICS Connector adds 1 to its TASKS parameter when it sends the CONNECT request to the PDM. This ensures a slot for the DISCONNECT task. Account for these in your environment description tasks parameter for the PDM. You do not need to account for the DDI and termination tasks; the PDM leaves space for them.

- ◆ The PDM allocates RPLs (request parameter lists) during initialization for ESDS and at open time for KSDS. The VSAM RPL for OS/390 is approximately 76(4C) bytes; for VSE approximately 52(34) bytes (this can change with releases). There is one RPL per VSAM string per file (VSAM string = (Direct buffers + (Threads * Serial buffers)). For ESDS, these RPLs are shared; for KSDS, they are not. For example, if 20 ESDS share a buffer pool of 11 buffers, only 11 RPLs are allocated. If 20 KSDS share a buffer pool of 11 buffers, 220 RPLs are allocated. The KSDS buffers themselves are outside the memory pool (see “[Memory requirements for VSAM PDM data sets](#)” on page 132).
- ◆ The Command-related memory item in the User Variable category depends on your files and the maximum amount of update and read activity at any one moment. Use the following procedure to determine the maximum amount needed for maximum concurrent command activity (threads value). The PDM acquires the memory as needed during command processing and releases it for reuse upon command completion.

The PDM uses various control blocks, depending on command type:

- MRWA—Maximum Record Work Area control block. This contains the image of a record as it was at the beginning of command processing.
- RWA—Record Work Area control block. This is a work area for some commands. Length varies by record length.
- SR—Structure Resource logging control block. You calculate this according to PDM type (primary, related) and access type (BDAM, ESDS, KSDS).
- ICB—Index maintenance control block. You calculate this according to access type (BDAM, ESDS, KSDS).

Calculating SR (Structure resource) control block size

The calculation for determining the size of a structure resource is different for primary files and related files.

$$\text{Primary file SR} = (M + 44) (25) + 32 (2S + 8) + 2L + 16$$

where:

M = Maximum length of any secondary key in the file

S = Number of secondary keys in the file

L = Logical record length of the file

$$\text{Related file SR} = (M + 44) (S) + (((((B + C) + 1) * 2) + S) * 32) + L + 32$$

where:

M = Maximum length of any secondary key in the file

S = Number of secondary keys in the file

B = Maximum number of base linkpaths

C = Maximum number of coded linkpaths

L = Logical record length of the file

Calculating ICB (index maintenance) control block size

The ICB size depends on whether the base file is a BDAM, ESDS or KSDS file, and is calculated as follows.

- ◆ For each secondary key on a BDAM or ESDS file:

$$\text{Secondary key} = (2 * \text{key-length}) + 32$$

- ◆ For each secondary key on a KSDS file:

$$\text{Secondary key} = (4 * \text{key-length}) + 20$$

- ◆ Using the above value, you then calculate ICB:

$$\text{ICB} = \text{sum of all secondary key values} + \text{index file block size} + 252$$

Determine the amount needed for the largest primary or related file plus its indexes. Multiply this by the maximum number of concurrent updates you expect (updates, inserts, deletes). To tune this value, you can monitor with Statistics on and add the 2 PDM statistics S4.02 and S4.03 using the Execution Statistics report or Interactive Services (see “[Obtaining statistics](#)” on page 23). These show the total but not the maximum concurrent. You can use the S5.02 statistic to determine the highest number of concurrent commands (these include read commands).

Sample PDM memory requirements

The following sample tables use the PDM module sizes and the calculations from the table under “**MEMORY parameter calculation for the PDM**” on page 107 to provide sample values for PDM control block sizes, counts, and the memory required. You can estimate your needs from these samples. Sample values are provided for small, medium, and large user sites based on the following figures:

	Small	Medium	Large
Number of interfaces	3	5	8
Number of threads	5	10	20
Number of tasks	25	70	150
Active update tasks	2	5	8
Number of files	50	150	250

Definitions of the terms interfaces, threads and tasks are as follows:

- ◆ **Interface.** An internal program used to handle communication between SUPRA system programs. For example, the PDM interface handles communication between user applications and the PDM.
- ◆ **Thread.** A line of access through which work is performed. A thread can exist between an interface and the PDM, within an interface, or within the PDM. The number of requests in progress is limited by the number of threads.
- ◆ **Task.** The processing of an application program from log-on through log-off in either online or batch mode.

PDM memory requirements—small sites

The following table provides the sample values for small sites. In the table, you see items followed by a number within parentheses. These numbers refer to explanatory notes in “[Sample PDM memory requirements notes](#)” on page 128, such as (1) means see Note 1.

Item	Size in K	Count	Result K, rounded
Multitask PDM Code			480
Multitask Initialization Fixed Overhead (1)			30
DDI			
CSMYDDI (code)			32
CSMYDDI (context)			10
CSTESTMT (code)			124
CSTESTMT (context)			<u>32</u>
Total K:			198
Nonfile-related Memory			
PDM Fixed Overhead (2)			10
Interface Control Blocks (3)	4.80	3	15
Thread Control Blocks (4)	4.03	5	21
Task Control Blocks (5)	0.30	25	8
Log Group Control Blocks	0.16	1	1
System Log Buffer Pool (6)	4.00	2	8
Task Log Buffer Pool (7)	4.00	2	8
Log Synch. Control Blocks (8)			3
Trace Table (9)			<u>8</u>
Total K:			82

Item	Size in K	Count	Result K, rounded
User Files Memory			
File Control Blocks	1.04	50	52
Internal Record Control Blocks (10)	0.11	150	17
Physical Field Control Blocks (11)	0.04	3750	150
Record Holding Entry Control Blocks (12)	0.13	150	20
Dynamic Index Control Blocks (13)	0.32	100	33
Code Control Blocks (14)	0.05	100	6
Ix. Physical Field Control Blocks (15)	0.08	200	17
Buffer Pool Control Blocks	0.18	1	1
Blocks Per File (16)		75	
Bit Map (17)			5
Average Memory for Buffer Pools (24)			<u>236</u>
Total K:			537

Pertaining to the Average Buffer Pool entry above:

Buffers Per Buffer Pool (18)		50	
Buffer Pool (BDAM) (19)	4.67	50	233
Additional Memory for ESDS Files (20)	0.06+	50	3+
Additional Memory for KSDS Files (21)	0.06+	50	3+
Minimum Memory for Buffer Pools (22)			233
Maximum Memory for Buffer Pools (23)			239

Item	Size in K	Count	Result K, rounded
Directory Files Memory			
File Control Blocks	1.04	5	6
Internal Record Control Blocks	0.11	71	8
Physical Field Control Blocks	0.04	595	24
Record Holding Entry Control Blocks (12)	0.13	500	65
Dynamic Index Control Blocks (13)	0.32		1
Code Control Blocks (14)	0.05		1
Ix. Physical Field Control Blocks (15)	0.08		1
Buffer Pool Control Blocks	0.18	1	1
Blocks Per File (16)		60	
Bit Map (17)			1
Average Memory for Buffer Pools (24)			<u>13</u>
Total K:			121

Pertaining to the Average Buffer Pool entry above:

Buffers Per Buffer Pool (18)		2	
Buffer Pool (BDAM) (19)	5.30	2	12
Additional Memory for ESDS Files (20)	0.06+	2	1+
Additional Memory for KSDS Files (21)	0.06+	2	1+
Minimum Memory for Buffer Pools (22)			12
Maximum Memory for Buffer Pools (23)			14

Item	Size in K	Count	Result K, rounded
NORMAL Scratch Pad Memory			
File Control Blocks	1.04	16	17
Internal Record Control Blocks (10)	0.11	16	2
Physical Field Control Blocks (11)	0.04	149	6
Record Holding Entry Control Blocks (12)	0.13	50	7
Dynamic Index Control Blocks (13)	0.32		1
Code Control Blocks (14)	0.05		1
Ix. Physical Field Control Blocks (15)	0.08		1
Buffer Pool Control Blocks	0.18	1	1
Blocks Per File (16)		75	
Bit Map (17)			2
Average Memory for Buffer Pools (24)			<u>20</u>
Total K:			58
Pertaining to the Average Buffer Pool entry above:			
Buffers Per Buffer Pool (18)		4	
Buffer Pool (BDAM) (19)	4.67	4	19
Additional Memory for ESDS Files (20)	0.06+	4	1+
Additional Memory for KSDS Files (21)	0.06+	4	1+
Minimum Memory for Buffer Pools (22)			19
Maximum Memory for Buffer Pools (23)			21
Command-related Memory (25)			15
Grand Total for Small Sites:			1323

PDM memory requirements—medium sites

The following table provides the sample values for medium sites. In the table, you see items followed by a number within parentheses. These numbers refer to explanatory notes in “[Sample PDM memory requirements notes](#)” on page 128, such as (1) means see Note 1.

Item	Size in K	Count	Result K, rounded
Multitask PDM Code			480
Multitask Initialization Fixed Overhead (1)			30
DDI			
CSMYDDI (code)			32
CSMYDDI (context)			10
CSTESTMT (code)			124
CSTESTMT (context)			<u>32</u>
Total K:			198
Nonfile-related Memory			
PDM Fixed Overhead (2)			10
Interface Control Blocks (3)	4.80	5	24
Thread Control Blocks (4)	4.03	10	41
Task Control Blocks (5)	0.30	70	21
Log Group Control Blocks	0.16	2	1
System Log Buffer Pool (6)	4.00	6	24
Task Log Buffer Pool (7)	4.00	15	60
Log Synch. Control Blocks (8)			12
Trace Table (9)			<u>8</u>
Total K:			201

Item	Size in K	Count	Result K, rounded
User Files Memory			
File Control Blocks	1.04	150	156
Internal Record Control Blocks (10)	0.11	450	50
Physical Field Control Blocks (11)	0.04	11250	450
Record Holding Entry Control Blocks (12)	0.13	450	59
Dynamic Index Control Blocks (13)	0.32	300	97
Code Control Blocks (14)	0.05	300	16
Ix. Physical Field Control Blocks (15)	0.08	600	49
Buffer Pool Control Blocks	0.18	5	1
Blocks Per File (16)		500	
Bit Map (17)			21
Average Memory for Buffer Pools (24)			<u>946</u>
Total K:			1845

Pertaining to the Average Buffer Pool entry above:

Buffers Per Buffer Pool (18)		15	
Buffer Pool (BDAM) (19)	4.67	150	700
Additional Memory for ESDS Files (20)	0.06+	150	9+
Additional Memory for KSDS Files (21)	0.06+	150	9+
Minimum Memory for Buffer Pools (22)			934
Maximum Memory for Buffer Pools (23)			958

Item	Size in K	Count	Result K, rounded
Directory Files Memory			
File Control Blocks	1.04	5	6
Internal Record Control Blocks	0.11	71	8
Physical Field Control Blocks	0.04	595	24
Record Holding Entry Control Blocks (12)	0.13	1000	130
Dynamic Index Control Blocks (13)	0.32		1
Code Control Blocks (14)	0.05		1
Ix. Physical Field Control Blocks (15)	0.08		1
Buffer Pool Control Blocks	0.18	3	1
Blocks Per File (16)		250	
Bit Map (17)			1
Average Memory for Buffer Pools (24)			<u>73</u>
Total K:			246

Pertaining to the Average Buffer Pool entry above:

Buffers Per Buffer Pool (18)		4	
Buffer Pool (BDAM) (19)	5.30	12	72
Additional Memory for ESDS Files (20)	0.06+	12	1+
Additional Memory for KSDS Files (21)	0.06+	12	1+
Minimum Memory for Buffer Pools (22)			72
Maximum Memory for Buffer Pools (23)			74

Item	Size in K	Count	Result K, rounded
NORMAL Scratch Pad Memory			
File Control Blocks	1.04	16	17
Internal Record Control Blocks (10)	0.11	16	2
Physical Field Control Blocks (11)	0.04	149	6
Record Holding Entry Control Blocks (12)	0.13	100	13
Dynamic Index Control Blocks (13)	0.32		1
Code Control Blocks (14)	0.05		1
Ix. Physical Field Control Blocks (15)	0.08		1
Buffer Pool Control Blocks	0.18	1	1
Blocks Per File (16)		75	
Bit Map (17)			2
Average Memory for Buffer Pools (24)			<u>48</u>
Total K:			92
Pertaining to the Average Buffer Pool entry above:			
Buffers Per Buffer Pool (18)		10	
Buffer Pool (BDAM) (19)	4.67	10	47
Additional Memory for ESDS Files (20)	0.06+	10	1+
Additional Memory for KSDS Files (21)	0.06+	10	1+
Minimum Memory for Buffer Pools (22)			47
Maximum Memory for Buffer Pools (23)			49
Command-related Memory (25)			45
Grand Total for Medium Sites:			3137

PDM memory requirements—large sites

The following table provides the sample values for large sites. In the table, you see items followed by a number within parentheses. These numbers refer to explanatory notes in “[Sample PDM memory requirements notes](#)” on page 128, such as (1) means see Note 1.

Item	Size in K	Count	Result K, rounded
Multitask PDM Code			480
Multitask Initialization Fixed Overhead (1)			30
DDI			
CSMYDDI (code)			32
CSMYDDI (context)			10
CSTESTMT (code)			124
CSTESTMT (context)			<u>32</u>
Total K:			198
Nonfile-related Memory			
PDM Fixed Overhead (2)			10
Interface Control Blocks (3)	4.80	8	39
Thread Control Blocks (4)	4.03	20	81
Task Control Blocks (5)	0.30	150	45
Log Group Control Blocks	0.16	2	1
System Log Buffer Pool (6)	4.00	18	72
Task Log Buffer Pool (7)	4.00	30	120
Log Synch. Control Blocks (8)			27
Trace Table (9)			<u>8</u>
Total K:			403

Item	Size in K	Count	Result K, rounded
User Files Memory			
File Control Blocks	1.04	250	260
Internal Record Control Blocks (10)	0.11	750	83
Physical Field Control Blocks (11)	0.04	18750	563
Record Holding Entry Control Blocks (12)	0.13	750	98
Dynamic Index Control Blocks (13)	0.32	500	160
Code Control Blocks (14)	0.05	500	25
Ix. Physical Field Control Blocks (15)	0.08	1000	80
Buffer Pool Control Blocks	0.18	10	2
Blocks Per File (16)		1200	
Bit Map (17)			102
Average Memory for Buffer Pools (24)			<u>2820</u>
Total K:			4193

Pertaining to the Average Buffer Pool entry above:

Buffers Per Buffer Pool (18)		25	
Buffer Pool (BDAM) (19)	4.67	250	1168
Additional Memory for ESDS Files (20)	0.06+	250	15+
Additional Memory for KSDS Files (21)	0.06+	250	15+
Minimum Memory for Buffer Pools (22)			2802
Maximum Memory for Buffer Pools (23)			2838

Item	Size in K	Count	Result K, rounded
Directory Files Memory			
File Control Blocks	1.04	5	6
Internal Record Control Blocks	0.11	71	8
Physical Field Control Blocks	0.04	595	24
Record Holding Entry Control Blocks (12)	0.13	1500	195
Dynamic Index Control Blocks (13)	0.32		1
Code Control Blocks (14)	0.05		1
Ix. Physical Field Control Blocks (15)	0.08		1
Buffer Pool Control Blocks	0.18	5	1
Blocks Per File (16)		400	
Bit Map (17)			1
Average Memory for Buffer Pools (24)			<u>272</u>
Total K:			510
Pertaining to the Average Buffer Pool entry above:			
Buffers Per Buffer Pool (18)		9	
Buffer Pool (BDAM) (19)	5.30	45	269
Additional Memory for ESDS Files (20)	0.06+	45	3+
Additional Memory for KSDS Files (21)	0.06+	45	3+
Minimum Memory for Buffer Pools (22)			269
Maximum Memory for Buffer Pools (23)			275

Item	Size in K	Count	Result K, rounded
NORMAL Scratch Pad Memory			
File Control Blocks	1.04	16	17
Internal Record Control Blocks (10)	0.11	16	2
Physical Field Control Blocks (11)	0.04	149	6
Record Holding Entry Control Blocks (12)	0.13	150	13
Dynamic Index Control Blocks (13)	0.32		1
Code Control Blocks (14)	0.05		1
Ix. Physical Field Control Blocks (15)	0.08		1
Buffer Pool Control Blocks	0.18	1	1
Blocks Per File (16)		75	
Bit Map (17)			2
Average Memory for Buffer Pools (24)			<u>71</u>
Total K:			115
Pertaining to the Average Buffer Pool entry above:			
Buffers Per Buffer Pool (18)		15	
Buffer Pool (BDAM) (19)	4.67	15	70
Additional Memory for ESDS Files (20)	0.06+	15	1+
Additional Memory for KSDS Files (21)	0.06+	15	1+
Minimum Memory for Buffer Pools (22)			70
Maximum Memory for Buffer Pools (23)			71
Command-related Memory (25)			135
Grand Total for Large Sites:			6064

Sample PDM memory requirements notes

This section provides the notes for the tables under “PDM memory requirements—small sites” on page 116, “PDM memory requirements—medium sites” on page 120, and “PDM memory requirements—large sites” on page 124. The number of each note corresponds to a number in parentheses following an item title; for example, PDM Fixed Overhead (2).

1. Multitask initialization fixed overhead is the memory required for context during system initialization. Most of this context is used during a warm start.
2. PDM fixed overhead is the memory required for ENDTO and DDI control blocks.
3. Interface is the assumed maximum number of interfaces communicating with the multitask PDM.
4. Threads are the assumed maximum number of threads for all interfaces.
5. Tasks are the assumed maximum number of tasks for all interfaces.
6. Size is the assumed block size of the system log and count is the number of system log buffers.
7. Size is the block size of the task log and count is the number of task log buffers.
8. Log synchronization control blocks are required for synchronization of the task and system log. The calculation of Result K for log synchronization is:

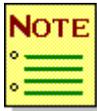
$$\text{Result K} = ((\text{TBUF} * \text{TBLK} / 200) + (\text{SBUF} * \text{SBLK} / 200)) * 28$$

bytes
9. Trace table is the amount of memory required for the PDM in-core trace table. You can specify this size in the CSIPARM file.
10. The count of internal records assumes an average of 3 internal records per file (multiplied by File Control Blocks).

11. The count of physical fields assumes 25 physical fields per internal record (multiplied by Internal Record Control Blocks).
 12. The count of record holding entries assumes an average of 25 record holding entries for each task during a single PDM transaction (multiplied by Task Control Blocks). The memory for held records is actually calculated as Maximum Held Records times the control block size. All memory for record holding entries is allocated when the PDM initializes (it is not dynamic).
 13. The count of dynamic indices assumes an average of 2 dynamic indices per file (multiplied by File Control Blocks).
 14. The count of index codes assumes an average of 2 index codes per dynamic index (multiplied by Dynamic Index Control Blocks).
 15. The count of index fields assumes an average of 1 index field per index code (multiplied by Code Control Blocks).
 16. Blocks per file is the average number of physical blocks per file.
 17. Bit map is the amount of memory required for the bit map for an average file, calculated in bytes as follows:
- $\text{Result K} = (\text{Blocks} / 8) + 25$
- where *Blocks* is the value of Blocks Per File.
18. Count is the average number of buffers per buffer pool for the indicated files.
 19. Size is the assumed average block size for the indicated files. Result K is the amount of memory required for buffers assuming that all files sharing the buffer are BDAM files. Result K is calculated in bytes as follows:

$$\text{Result K} = (\text{Buffers} * (688 + \text{Block Size}))$$

where *Buffers* is the value of Buffers Per Buffer Pool.



If only a KSDS file is using the buffer pool, this value is zero.

20. Result K is the extra memory required for RPLs if any ESDS files share the buffer pool, calculated in bytes as follows:

$$\text{Result K} = (\text{Buffers} * 64)$$

Additional memory is required and allocated by VSAM directly from the operating system. See “[Memory requirements for VSAM PDM data sets](#)” on page 132 for these memory requirements.

21. Result K is the extra memory required for RPLs if any KSDS files share the buffer, calculated in bytes as follows:

$$\text{Result K} = (\text{Buffers} * 64)$$

Additional memory is required and allocated by VSAM directly from the operating system. See “[Memory requirements for VSAM PDM data sets](#)” on page 132 for these memory requirements.

22. Result K is the amount of memory required for BDAM buffers (minimum). This minimum is used in number 19.
23. Result K is the amount of memory required assuming that each buffer pool contains BDAM, ESDS, and KSDS files (maximum).
24. Result K is the average of the minimum and maximum requirements for buffers (22 and 23). This average value is used in calculating the sample amounts.
25. Command-related memory is required to process each PDM DML command received from a PDML application or from SUPRA components (RDM, Directory Maintenance, etc.). This memory is acquired as needed during the processing of the command and released upon completion. This memory includes control blocks for structure resources (SR), index maintenance (ICB), a maximum record work area (MRWA), and a record work area (RWA) as described in the notes for the table under “[MEMORY parameter calculation for the PDM](#)” on page 107.

The following table shows the amount of memory required for the control blocks, which were then used to calculate the average amount according to site size. The memory in this summary table assumes several factors for the formulas in THE TABLE UNDER “MEMORY parameter calculation for the PDM” on page 107:

BDAM or ESDS files

M (Maximum length of any secondary key) = 64

S (Number of secondary keys in the file)= 2

B (Maximum number of base linkpaths) = 2

C (number of coded linkpaths) = 2

L (Logical record length of the file) = 256

Block size of an index file = 4096

Type of command	MRWA	RWA	SR	ICB	Total
UPDATE command	1K	-	2K	5K	8K
READ commands	-	1K	-	-	1K
READX command	1K	-	-	5K	6K
Other commands	-	1K	-	-	1K

Assuming a mix of 15% UPDATE commands, 80% READ commands, 3% READX commands, and 2% other commands, the average command requires 3K of memory for the control blocks. Assuming the threads value (at the beginning of the tables under “MEMORY parameter calculation for the PDM” on page 107, “PDM memory requirements—small sites” on page 116, and “PDM memory requirements—medium sites” on page 120) represents the average number of concurrent commands, the Result K values for command-related memory (at the end of the tables) were calculated as follows: small sites: 5 * 3K = 15K; medium sites: 15 * 3K = 45K; large sites: 45 * 3K = 135K.

Memory requirements for VSAM PDM data sets

This section provides buffer memory requirements and VSAM control-block allocation for VSAM PDM data sets. ESDS buffer memory comes from the environment description memory allocation; KSDS buffer memory does not, but they both use GETMAIN or GETVIS space. For information on memory pool calculations, see [“MEMORY parameter calculation for the PDM”](#) on page 107.

Buffer-size calculation for KSDS

Use the following formula to calculate the buffer size for each KSDS. Storage for KSDS buffers is not allocated from the memory pool. It is an additional allocation in the GETMAIN portion of the REGION (OS/390) or the GETVIS portion of the partition (VSE). See [“Using PDM memory”](#) on page 66 about memory usage.

$$(MI * (D + 1)) + (MD * (D + 1))$$

where:

MI = Index control interval size

MD = Data control interval size

D = Direct buffers

VSAM control-block allocation

The following tables provide VSAM control-block allocation per file within GETMAIN/GETVIS storage. See the [notes](#) following the table for more information. The following expressions are used in the table:

p	= per
BUF	= VSAM buffer
KL	= VSAM key length
EXT	= VSAM extent
STR	= VSAM string
STRNO	= VSAM string number

The table does not include VSAM control blocks used for alternate indexes.

For OS/390, some of the control blocks are allocated not within the PDM region but within SQA (subpool 245). An approximate relation of memory allocation is: CSA/PRIV to SQA is 100 to 15. Thus, if 100K is allocated within CSA, about 15K is allocated within SQA.

OS/390 allocations

Cont. blk. Name	Length (bytes)	Number allocated		Storage used		Notes
		ESDS	KSDS	ESDS	KSDS	
AMB	188	1	2	188	376	1
AMBL	68	1	1	68	68	-
AMBXN	60	1pSTR	2pSTR	60*STR	120*STR	2,3
AMDSB	96	1	2	96	192	1
ARDB	28+KL*2	-	4 (min)	0	128 (min)	4
BIB	120	1	1	120	120	-
BUFC	84	1pBUF	2pBUF	84pBUF	168*BUF	2,5
BUFFER		0		0		6
CLW	30	1	1	30	30	-
CMB	76	1	1	76	76	-
DEB	28	(1pEXT)+ 84	(1pEXT)+ 84	(28*EXT)+84	(28*EXT)+84	2,7,8
DIWA	96	1	1	96	96	-
ESL/EDB	30	1pEXT	1pEXT	30*EXT	30*EXT	2,7
HEB	332	1	1	332	332	-
ICWA	60+KL*3	-	1 (min)	0	66 (min)	9
IMWA	79+KL*3	-	1 (min)	0	82 (min)	10
IOMB	148	1pSTR	2pSTR	148*STR	296*STR	2,3
IOSB	108	1pSTR	2pSTR	108*STR	216*STR	2,3
IQE	24	1pSTR	2pSTR	24*STR	48*STR	2,3
IRB	164	1pSTR	2pSTR	164*STR	328*STR	2,3
LPMB	28	1	2	28	56	1
OPW	816	1	1	816	816	-
PFL	16	1pSTR	2pSTR	16*STR	32*STR	2,3
PLH	308+KL	-	1pSTR	0	310*STR (min)	2,11
SRB	44	1pSTR	2pSTR	44*STR	88*STR	2,3
SSL	148	1	1	148	148	-

VSE allocations

Cont. blk.	Length	Number allocated		Storage used		
Name	(bytes)	ESDS	KSDS	ESDS	KSDS	Notes
AMBL	100	1	1	100	100	-
AMDSB	200	1	2	200	400	1
ARDB	256	0	4 (min)	0	1024 (min)	4
BCB	72	1pBUF	2pBUF	72*BUF	144*BUF	2,5
BHD	52	1pSTR	2pSTR	52*STRNO	104*STRNO	2,12
BUFFS	?	0	?	0	?	6
CPWA	2048	1	1	2048	2048	-
EDB	40	1pEXT	1pEXT	40*EXT	40*EXT	7
LPMB	24	1	1	24	24	-
PLH	500	1pSTR	1pSTR	500*STRNO	500*STRNO	2,3

Notes for the preceding tables

1. ESDSs: 1 per file.
KSDSs: 1 each for data and index per file.
2. The PDM performs the following calculation for VSAM strings and buffers:
no. strings (STR) = direct buffers + (threads * serial buffers)
no. index buffers (KSDS) = direct buffers + 1
no. data buffers (KSDS) = direct buffers + 1
no. data buffers (ESDS) = direct buffers + (threads * serial buffers) + 1
3. One per VSAM string, whether or not buffers within the PDM are shared.
4. Keeps track of key ranges. A simple KSDS has 4. Each key range adds 1 or more.
5. ESDSs: 1 per VSAM buffer.
KSDSs: 2 per VSAM buffer (1 each for data and index).
6. ESDSs: No additional storage required (satisfied in PDM memory pool).
KSDSs: Calculate memory using the formula described in “[Buffer-size calculation for KSDS](#)” on page 132.
7. 1 per VSAM extent (use IDCAMS LISTCAT function to verify).
At least 2 for KSDS (1 each for data and index).
8. For the DEB, add 84 bytes more per file regardless of the number of extents.
9. At least 1 per KSDS. Keeps track of index levels per index.
10. At least 1 per KSDS. Keeps track of index inserts.
11. The formula for PLH is: **20 + (STR * (308 + KL))**.
12. ESDS files: 1 per VSAM string.
KSDS files: 2 per VSAM string (1 each for data and index).

Minimum storage requirements for PDM VSAM files

This section summarizes the control block requirements in the previous section and describes further requirements.

OS/390 requirements

Calculate the following to obtain the minimum storage requirements for VSAM PDM files in OS/390:

- ◆ Calculate the following for each ESDS:

2082

+ (564 * (direct buffers + (threads * serial buffers)))

+ (58 * VSAM extents)

+ (84 * (direct buffers + (threads * serial buffers)) + 1)

- ◆ Calculate the following for each KSDS:

2642

+ (1128 * direct buffers)

+ ((308 + KL) * direct buffers)

+ (168 * (direct buffers + 1))

+ (CI/size index * (direct buffers + 1))

+ (CI/size data * (direct buffers + 1))

+ (14 * KL)

- ◆ Calculate the following for other GETMAIN requirements:

(40K * catalogs), or double if both master and user catalogs

+ 400K for Directory interface, bootmod, and other GETMAIN requests (non-VSAM, OPEN, etc.)

+ boot memory pool

+ user memory pool

VSE requirements

Calculate the following to obtain the minimum storage requirements for VSAM PDM files in VSE:

- ◆ Calculate the following for each ESDS file:

2372

+ (552 * (direct buffers + (threads * serial buffers)))

+ (40 * VSAM extents)

+ (72 * ((direct buffers + (threads * serial buffers)) + 1))

- ◆ Calculate the following for each KSDS file:

3596

+ (604 * direct buffers)

+ (40 * VSAM extents data)

+ (40 * VSAM extents index)

+ (144 * (direct buffers + 1))

+ (CI/size index * (direct buffers + 1))

+ (CI/size data * (direct buffers + 1))

- ◆ For other GETVIS requirements, calculate:

(40K * catalogs), or double if both master and user catalogs

+ 310K for Directory interface, bootmod, and other GETVIS requests (non-VSAM OPEN, etc.) If any VSAM modules are loaded into the partition, add the size for each module.

+ boot memory pool

+ user memory pool

Minimum size of PDM with VSAM files

To calculate the minimum partition size of the PDM with VSAM files, add the size of the PDM modules to the result of the calculations in “[Minimum storage requirements for PDM VSAM files](#)” on page 137.

Memory requirements for Directory Maintenance

For Directory Maintenance, you have the option of having only certain modules in memory (CICS user DSA) at any one time. You can use 3–12 load modules for batch processing or only 1 module. You can use 2–12 load modules for CICS processing. The following tables describe the sizes of each Directory Maintenance load module and the minimum storage needed according to which configuration you choose. To change the number of load modules, use the C\$MDMLNK macro. For the macro description, refer to the *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250.

Batch configurations

Name	Number of load modules/module size in K					
	1	2	3	4	5	6
CSMBDIRM	762	n/a	272	283	283	283
CSMBDIR1		n/a	211	189	189	189
CSMBDIR2		n/a	321	114	114	81
CSMBDIR3		n/a		235	185	130
CSMBDIR4		n/a			68	83
CSMBDIR5		n/a				91
Minimum Storages						
Non-TLOG	781	n/a	634	559	513	513
With TLOG	800	n/a	653	578	532	532
CSMBDIRM	216	216	216	216	216	116
CSMBDIR1	179	32	32	32	32	32
CSMBDIR2	59	59	59	59	59	59
CSMBDIR3	120	120	120	63	63	63
CSMBDIR4	73	73	59	59	37	37
CSMBDIR5	81	81	81	26	26	26
CSMBDIR6	87	104	92	92	92	92
CSMBDIR7		137	137	137	137	137
CSMBDIR8			35	35	35	35
CSMBDIR9				120	120	120
CSMBDIRA					30	30
CSMBDIRB						109
Minimum Storages						
Non-TLOG	436	394	394	394	394	294
With TLOG	455	413	413	413	413	313

CICS configurations

Name	Number of load modules/module size in K					
	1	2	3	4	5	6
CSMCDIRM	n/a	277	277	287	287	281
CSMCDIR1	n/a	508	214	189	189	184
CSMCDIR2	n/a		322	118	118	80
CSMCDIR3	n/a			236	186	129
CSMCDIR4	n/a				68	85
CSMCDIR5	n/a					91
Minimum Storages (includes 3K for CSMQCMP):						
Add 61K per thread for each configuration						
	n/a	788	602	526	479	468
CSMCDIRM	221	221	221	221	221	118
CSMCDIR1	179	32	32	32	32	32
CSMCDIR2	59	59	59	59	59	59
CSMCDIR3	120	120	120	64	64	64
CSMCDIR4	76	76	59	59	37	37
CSMCDIR5	83	83	83	27	27	27
CSMCDIR6	87	104	92	92	92	92
CSMCDIR7		137	137	137	137	137
CSMCDIR8			38	38	38	38
CSMCDIR9				120	120	120
CSMCDIRA					30	30
CSMCDIRB						111
Minimum Storages (includes 3K for CSMQCMP):						
Add 61K per thread for each configuration						
	403	361	361	361	361	258

Memory requirements for the CICS Connector

The CICS Connector requires about 333K memory for its modules; 177K of this memory is resident. Some of the dynamic memory required comes from its memory pool and some comes from outside the pool.

Memory pool requirements

The Connector obtains the memory pool requirements from the CICS shared DSA (CICS GETMAIN SHARED). The memory pool can grow as needed, up to the maximum indicated by the MEME parameter of the CONNECT or the CSTXOPRM macro default (see “MEME parameter” on page 85).

- ◆ SOTM (Signed on table) = $96 \text{ bytes} * (\text{number of tasks} + 1)$
- ◆ Hash header table = 2K
- ◆ PDM interface control blocks:
 - $(\text{MAXPACKET} + 4\text{K}) * \text{THREADS}$ (For VSE XPCC)
 - $256 * \text{TASKS}$
 - $4\text{K} * \text{THREADS}$ (for OS/390)
- ◆ Active Directory information requests = Additional 4K during request to the PDM.

The exact amount of memory required depends in part on the combination of RDML requests issued by the application programs and the activities of Directory Maintenance.

Outside of memory pool requirements

The memory outside the pool is obtained from CICS DSA.

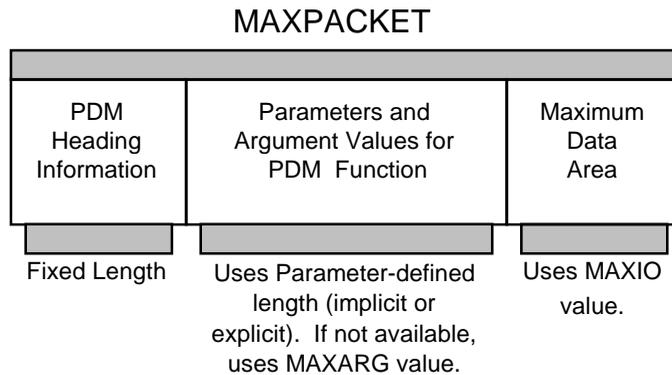
- ◆ DTC (Dynamic task context) =
 - 2560 bytes reserved for a global area for T.R.U.E. usage acquired at connect and release at disconnect.
- $(4\text{K} + 920 \text{ bytes}) * \text{number of active tasks that issue calls to the PDM}$

Resident Program requirements for SUPRA Server in a CICS environment

Component	Entity	Resident
CICS Connector	CSTXCSMT	146K
	CSTXRMDK	42K
	CSTXRMOS	21K
	CSTXUSER	3K
		212K
DBAID (RDM)	CSVLVDBA	145
	CSVODBA	58
		203
Directory Maint. (see the table under “Memory requirements for Directory Maintenance” on page 139 about minimum module sizes)		Transaction DIRM is conversational and therefore resident while active.
RDM	CSVNPLVS	45
	CSNVRES	565
	CSNVRUN	35
	CSVOPLVS	2
		651

Performance considerations for VSE XPCC

When using VSE/SP2, XPCC is the method of communication for the PDM when the PDM and the application are in different address spaces (see “[Tuning CSIPARM parameters](#)” on page 57). XPCC theoretically allows you to use the full 40 MB of virtual storage provided with SP2 systems. This method uses packet switching. When configuring your system, you should account for additional memory for packet switching. The CSIPARM file contains the MAXIO, MAXPACKET, and MAXARG parameters for XPCC support. The following figure shows the use of these parameters.



The PDM uses these parameters to allocate areas (packets) based on the number of threads. You can use the following suggested values:

- ◆ Set MAXARG equal to or greater than the maximum PDM DML FINDX argument length (RDM uses FINDX).
- ◆ Set MAXIO equal to or greater than the maximum implicit data area length.
- ◆ Set MAXPACKET equal to or greater than 2K + MAXARG + the greater of MAXIO or largest explicit data area.

Refer to the [SUPRA Server PDM and Directory Administration Guide \(OS/390 & VSE\)](#), P26-2250, for specifying these parameters on the CSIPARM file. Use the following tuning information.

If you do not use the suggested values, consider the following when determining the size of MAXPACKET:

- ◆ The PDM heading information, which is a fixed-length, contiguous area preallocated by the interface.
- ◆ The maximum length specified within a user-defined parameter. The data length varies by DML statement.
- ◆ The maximum length of your system data and user DML parameters.
- ◆ Any undefined parameter lengths. Some DML parameter lengths are not predefined. In that case, the values of MAXARG and MAXIO determine the maximum length.

XPCC memory usage

Using XPCC increases memory used by the PDM interface. If the values of CSIPARM file MAXIO, MAXARG, and MAXPACKET are too high for actual program use, excessive memory requirements can result. The memory requirements for packet areas are based on the MAXPACKET specification and the number of threads. (MAXPACKET times the number of threads equals the size of the packet pool.)

In CICS environments, the Dynamic Storage Area (DSA) is taken at connect time and later, when needed by the MEME parameter in the OPER CONNECT command. Ensure the value of MEME includes enough memory for the CSIPARM MAXPACKET times the number of CICS Connector threads. You may need to increase the SIZE parameter on the EXEC statement of DFHSIP. This may also require increasing the partition.

For batch applications running against a central PDM, this memory is acquired from the GETVIS area.

Within the PDM, memory is allocated from the definition in the user- and boot-environment descriptions. With XPCC you may need to increase the memory specified in the environment description depending on the excess that existed in the pool and the size of the packets. The memory allocated is based on the actual size of the information sent rather than on MAXPACKET.

XPCC CPU usage

If you set the values of MAXIO or MAXARG significantly larger than necessary, CPU use by the PDM interface and the XPCC facility can be adversely affected.

If you set the value of MAXIO too small, an error status is returned.

Application memory usage with XPCC

MANTIS operates from a base program of approximately 300K that is resident in CICS. A MANTIS transaction processes from 2 data areas that could reach 32K each. If the transaction executes a DO group, this DO group may also use 64K. Therefore, external DO group usage and the level of DO groups are essential in estimating the use of memory in a MANTIS application.

Estimating the average MANTIS transaction memory usage and the average COBOL size for a given transaction may provide a reasonable estimate on the amount of DSA to have available above the requirements for the SUPRA connection. For example, if the average COBOL program is 64K and the average MANTIS program uses one external DO (64K plus 64K), and you have 20 active users (10 COBOL and 10 MANTIS), you will need up to 1.9 MB of DSA. This example provides a rough estimate from which you can begin your tuning efforts.

Index disk space calculations

Most of this appendix has dealt with the execution memory requirements of SUPRA Server. This section provides a way for you to calculate the amount of disk space (in blocks) you need for a secondary key, also known as an index. When you populate the secondary key, SUPRA builds the blocks as a tree, unlike primary and related file structure or sequential log files.

An index file can contain more than one index (secondary key). These calculations help you determine the number of blocks and optimum block size to use for one key based on its associated file's characteristics. Secondary keys on the same physical file must use the same block size.

“[Calculating index space](#)” on page 148 describes the steps and the formulas to determine the space needed for the blocks. “[Sample program for index space calculation](#)” on page 152 shows an example program to perform the steps and calculations using your variable input.

Calculating index space

Perform the following outlined steps to calculate your index space requirements for one secondary key:



Within the steps, some items are followed by a parenthetical expression such as (1.B). This shows that you are to use the result of a previous step in the current formula or calculation. For example, the calculation for uniqueness factor in Step 2.a uses the value of maximum records. Maximum records is followed by (1.a) indicating it is the result of Step 1.a.

1. Analyze your data and Directory information to determine the following values:
 - a. The capacity of the PDM data file (maximum records)
 - b. What fields comprise the secondary key (key length). Add 1 byte for each field that is data sensitive.
 - c. The number of unique key values ($< =$ maximum records). For example:
 - If no 2 values for this secondary key have duplicate data, the number of unique key values = maximum records (value 1.1).
 - If the secondary key contains yes/no type data, the number of unique key values is 2.
 - d. The index file's blocksize (suggested value $> =$ 4096 bytes)
 - e. The index file's packing density (suggested value of .5)
 - f. If PDM data file is KSDS, REFPTR = size of primary key. If PDM data file is not KSDS and has direct pointers, REFPTR = 4. If PDM data file is not KSDS and has indirect pointers, REFPTR = size of primary key.

2. Solve the following to determine the number of low-level (sequence set) blocks:

a. Solve for the uniqueness factor (UF), X:1

$$UF = \frac{\text{Maximum Records(1.a)}}{\text{Unique Keys (1.c)}}$$

b. Solve for the average size of 1 low-level block entry (LLBYTES):

$$\text{LLBYTES} = \text{Key Length(1.b)} + 6 + (\text{REFPTR(1.f)} * \text{Uniqueness Factor (2.a)})$$

c. Solve for the span factor and average entries per low-level block (ENTRIESLLB):

$$\text{IF } (\text{Blocksize(1.d)} - 36) \leq \text{LLBYTES(2.b)}$$

THEN

$$\text{Span Factor} = \frac{\text{LLBYTES} - (\text{Key Length(1.B)} + 6)}{(\text{Blocksize} - 36) - (\text{Key Length} + 6)}$$

and

$$\text{Average entries per low-level block} = 1$$

ELSE

$$\text{Span factor} = 1$$

and

$$\text{Avg. entries per low-level block} = \text{INT} \frac{\text{Blocksize} - 36}{\text{LLBYTES}}$$

d. Solve for the estimated number of low-level blocks (QTYLLB):

$$\text{QTYLLB} = \text{INT} \left(\frac{\text{Unique keys(1.c)} * \text{LLBYTES}}{\frac{\text{Blocksize} - 36}{\text{Packing Density(1.e)}}} \right) + 1$$

3. Solve the following to determine the number of high-level blocks:

- a. Solve for the maximum entries per high-level block (ENTRIESHLB):

$$\text{ENTRIESHLB} = \text{INT} \frac{\text{Blocksize} - 36}{\text{Key Length} + 8}$$

If this ratio is less than 8 entries per block, the block size is too small. To improve performance, increase the block size.

- b. Solve for the maximum number of lowest level, high-level entries needed (MLLHE) (shown as WORKIT in sample program):

$$\text{MLLHE} = \left(\frac{\text{QTYLLB (2.d)}}{\text{Span Factor (2.c)}} \right) + 1$$

- c. Solve for the average number of entries (spread) used per high-level block that results during a block split:

$$\text{Spread} = \text{ENTRIESHLB}(3.a) * \text{Packing Density}(1.e)$$

- d. Solve for the index tree's structure depth (STRUDEPTH) and the estimated number of high-level blocks (HLBLKS):

- IF MLLHE (3.B) <= ENTRIESHLB (3.A)

THEN Depth = 2 and high-level blocks = 1

ELSE Depth = 1 and high-level blocks = 0

- MLLHE = INT $\frac{\text{MLLHE}(\text{latest value})}{\text{Spread}(3.c)}$ (Y = next level)

$$\text{High-level blocks} = \text{HLBLKS} + \text{MLLHEY}$$

Depth = Depth +1

$$\text{MLLHE} = \text{INT} \frac{\text{MLLHEY}(\text{latest value})}{\text{ENTRIESHLB}(3.a)} + 1$$

IF MLLHE > ENTRIESHLB(3.a)

THEN Perform 3.d.ii again

ELSE Proceed to 4

4. Analyze the preliminary results:
 - a. High-level blocks must be ≥ 1 .
If not, high-level blocks = 1.
 - b. If depth = 3, high-level blocks must be ≥ 3 .
If not, high-level blocks = 3.
 - c. High-level blocks must be \leq low-level blocks.
If not, high-level blocks = low-level blocks.

5. Solve for estimated total number of blocks:

Total blocks = total high-level blocks + total low-level blocks.
Expressed as a formula, this is (TOTALBLKS = QTYHLB + QTYLLB).

Sample program for index space calculation

This sample MANTIS program will automatically calculate your index file space requirements. Calculations are done for maximum values and active values.

87/01/12 - 11:02:28

MANTIS PRINT FACILITY

USER - DALLAS

MANTIS PROGRAM

NAME : B_INDEX_BLK_CALC

DESCRIPTION : CALCULATE REQ'D.

BLKS. FOR SECKEY

PROGR BOUND : NO

```

1  ENTRY B_INDEX_BLK
2  .| THIS PROGRAM COURTESY OF CINCOM
3  .TEXT DATAFILENAME(4),INDXFILENAME(4),SECKEYNAME(8),MSG(72)
4  .ACTIVERCD=ZERO:DENSITY=ZERO:ENTRIESHLB=ZERO:ENTRIESLLB=ZERO
5  .KSDSKEYLGTH=ZERO:MAXRCD=ZERO:QTYHLB=ZERO:QTYLLB=ZERO:SECKEYLGTH=ZERO
6  .STRUDEPTH=ZERO:TOTALBLKS=ZERO:UNIQUEFACTOR=ZERO:UNIQUEKEY=ZERO
7  .SCREEN MAP("DALLAS:B_INDEX_BLK")
8  .DATAFILENAME="      ":INDXFILENAME="      ":SECKEYNAME="      "
9  .BLKSIZE=4096:PACKING=.5:DISPLAYNAME="THIS"
10 .CONVERSE MAP
11 .WHILE KEY<>"CANCEL"
12 ..IF" "=INDXFILENAME(1,1)OR SIZE(INDXFILENAME)<1
13 ...DISPLAYNAME="THIS"
14 ..ELSE
15 ...DISPLAYNAME=INDXFILENAME
16 ..END
17 ..ERRORCD=ZERO
18 ..MSG=""
19 ..IF PACKING>1 OR PACKING=ZERO
20 ...PACKING=.5
21 ...ATTRIBUTE(MAP,PACKING)="BRI"
22 ..END
23 ..DATAFILENAME=SECKEYNAME(1,4)
24 ..IF BLKSIZE<256

```

```
25 ...MSG="PLEASE PROVIDE BLOCKSIZE > 255 BYTES -- >=4096 RECOMMENDED"
26 ...ATTRIBUTE(MAP,BLKSIZE)="BRI,CUR"
27 ...CONVERSE MAP
28 ...MSG=" ":ERRORCD=1
29 ..ELSE
30 ...NOOFSECKEY=INT((BLKSIZE-128)/128)
31 ..END
32 ..IF MAXRCD=ZERO
33 ...MSG="PLEASE PROVIDE MAXIMUM RECORDS IN PDM DATA FILE"
34 ...ATTRIBUTE(MAP,MAXRCD)="BRI,CUR"
35 ...CONVERSE MAP
36 ...MSG=" ":ERRORCD=1
37 ..END
38 ..IF MAXRCD<ACTIVERCDS
39 ...MSG="ACTIVE RECORDS MAY NOT EXCEED MAXIMUM RECORDS"
40 ...ATTRIBUTE(MAP,ACTIVERCDS)="BRI,CUR"
41 ...CONVERSE MAP
42 ...MSG=" ":ERRORCD=1
43 ..END
44 ..IF ACTIVERCDS=ZERO
45 ...ACTIVERCDS=1
46 ..END
47 ..IF UNIQUEKEY=ZERO
48 ...IF ACTIVERCDS>1
49 ...UNIQUEKEY=ACTIVERCDS
50 ...ELSE
51 ...UNIQUEKEY=MAXRCD
52 ...END
53 ...ATTRIBUTE(MAP,UNIQUEKEY)="BRI"
54 ..END
55 ..IF UNIQUEKEY>MAXRCD
56 ...UNIQUEKEY=MAXRCD
57 ...ATTRIBUTE(MAP,UNIQUEKEY)="BRI"
58 ..END
59 ..IF SECKEYLGTH=ZERO
60 ...MSG="PLEASE PROVIDE THE SECONDARY KEY'S LENGTH"
61 ...ATTRIBUTE(MAP,SECKEYLGTH)="BRI,CUR"
62 ...CONVERSE MAP
63 ...MSG=" ":ERRORCD=1
```

```

64  ..END
65  ..UNIQUEFACTOR=MAXRCD/UNIQUEKEY
66  ..DENSITY=ACTIVERCD/MAXRCD
67  ..IF KSDSKEYLGTH=ZERO
68  ...FACTOR=4
69  ..ELSE
70  ...FACTOR=KSDSKEYLGTH
71  ..END
72  ..LLBYTES=SECKEYLGTH+6+(FACTOR*UNIQUEFACTOR)
73  ..IF BLKSIZE-36<(SECKEYLGTH+6+FACTOR)
74  ...MSG="PLEASE INCR. BLOCKSIZE. 1 ENTRY WILL NOT RESIDE WITHIN "
75  ...'+TXT(BLKSIZE)+" BYTES!!!"
76  ...ATTRIBUTE(MAP,BLKSIZE)="BRI,CUR"
77  ...CONVERSE MAP
78  ...MSG="":ERRORCD=1
79  ..END
80  ..IF(BLKSIZE-36)<=LLBYTES
81  ...SPANFACTOR=((LLBYTES-(SECKEYLGTH+6))/((BLKSIZE-36)-(SECKEYLGTH+6)))
82  ...ENTRIESLLB=1
83  ..ELSE
84  ...SPANFACTOR=1
85  ...ENTRIESLLB<5
86  ...IF ENTRIESLLB<5
87  ...MSG="INDEX BLOCK SIZE TOO SMALL FOR EFFECTIVE PERFORMANCE"
88  ...END
89  ..END
90  ..QTYLLB=INT(((UNIQUEKEY*LLBYTES)/(BLKSIZE-36))/PACKING)+1
91  ..ENTRIESHLB=INT((BLKSIZE-36)/(SECKEYLGTH+8))
92  ..STRUDEPTH=1:WORKIT=(QTYLLB/SPANFACTOR)+1
93  ..ADDHLBLKS=ZERO:WORKX=WORKIT
94  ..SPREAD=ENTRIESHLB*PACKING
95  ..IF ERRORCD=ZERO
96  ...IF WORKIT<=ENTRIESHLB
97  ....QTYHLB=1:STRUDEPTH=STRUDEPTH+1
98  ...ELSE
99  ....WHILE WORKIT>ENTRIESHLB+1
100  .....STRUDEPTH=STRUDEPTH+1
101  .....WORKX=INT(WORKIT/SPREAD)
102  .....ADDHLBLKS=ADDHLBLKS+WORKX

```

```

103     . . . . .WORKX=( (WORKIT/ENTRIESHLB)+1)
104     . . . . .WORKIT=WORKX
105     . . . . .END
106     . . . . .QTYHLB=ADDHLBLKS+1:STRUDEPTH=STRUDEPTH+1
107     . . . . .END
108     . . . . .TOTALBLKS=QTYLLB+QTYHLB
109     . . . . .IF  STRUDEPTH<2
110     . . . . .STRUDEPTH=2
111     . . . . .END
112     . . . . .IF  ACTIVERCDS<UNIQUEKEY
113     . . . . .AUNIQUEKEY=ACTIVERCDS
114     . . . . .ELSE
115     . . . . .AUNIQUEKEY=UNIQUEKEY
116     . . . . .END
117     . . . . .AUNIQUEFACTOR=INT (ACTIVERCDS/AUNIQUEKEY)
118     . . . . .ALLBYTES=SECKEYLGTH+6+(FACTOR*AUNIQUEFACTOR)
119     . . . . .IF  (BLKSIZE-36)<=ALLBYTES
120     . . . . .ASPANFACTOR=INT( (ALLBYTES-SECKEYLGTH-6)/((BLKSIZE-36)-(SECKEYLGTH+6)))
121     . . . . .AEPLLB=1
122     . . . . .ELSE
123     . . . . .ASPANFACTOR=1
124     . . . . .AEPLLB=INT( (BLKSIZE-36)/ALLBYTES)
125     . . . . .END
126     . . . . .AQTYLLB=INT( ((AUNIQUEKEY*ALLBYTES)/(BLKSIZE-36))/PACKING)+1
127     . . . . .AEPHLB=INT( (BLKSIZE-36)/(SECKEYLGTH+8))
128     . . . . .ASTRUDEPTH=1:WORKIT=(AQTYLLB/SPANFACTOR)+1
129     . . . . .ADDHLBLKS=ZERO:WORKX=WORKIT
130     . . . . .SPREAD=AEPHLB*PACKING
131     . . . . .IF  AQTYLLB+1<=AEPHLB
132     . . . . .AQTYHLB=1:ASTRUDEPTH=ASTRUDEPTH+1
133     . . . . .ELSE
134     . . . . .WHILE  WORKIT>AEPHLB+1
135     . . . . .ASTRUDEPTH=ASTRUDEPTH+1
136     . . . . .WORKX=INT(WORKIT/SPREAD)
137     . . . . .ADDHLBLKS=ADDHLBLKS+WORKX
138     . . . . .WORKX=( (WORKIT/AEPHLB)+1)
139     . . . . .WORKIT=WORKX
140     . . . . .END

```

```
141 ...AQTYHLB=ADDHLBLKS+1:ASTRUDEPTH=ASTRUDEPTH+1
142 ...END
143 ...ATOTALBLKS=AQTYLLB+AQTYHLB
144 ..END
145 ..IF ASTRUDEPTH<2
146 ...ASTRUDEPTH=2
147 ..END
148 ..IF STRUDEPTH=3 AND QTYHLB<3
149 ...QTYHLB=3:TOTALBLKS=QTYLLB+QTYHLB
150 ..END
151 ..IF ASTRUDEPTH=3 AND AQTYHLB<3
152 ...AQTYHLB=3:ATOTALBLKS=AQTYLLB+AQTYHLB
153 ..END
154 ..IF ERRORCD=ZERO
155 ...IF STRUDEPTH>10 OR SPANFACTOR+STRUDEPTH>55
156 ...MSG="INDEX BLOCK SIZE MAY BE TOO SMALL FOR EFFECTIVE PERFORMANCE"
157 ...END
158 ...IF AQTYLLB<AEPHLB
159 ...AQTYHLB=1:ASTRUDEPTH=2:ATOTALBLKS=AQTYHLB+AQTYLLB
160 ...END
161 ...|IF AQTYHLB>AQTYLLB
162 ...|AQTYHLB=AQTYLLB/AEPHLB
163 ...|IF AQTYHLB<1
164 ...|AQTYHLB=1
165 ...|ENDPHLB
166 ...|ATOTALBLKS=AQTYHLB+AQTYLLB
167 ...|END
168 ...IF QTYLLB<ENTRIESHLB
169 ...QTYHLB=1:STRUDEPTH=2:TOTALBLKS=QTYHLB+QTYLLB
170 ...END
171 ...|IF QTYHLB>QTYLLB
172 ...|QTYHLB=QTYLLB/ENTRIESHLB
173 ...|IF QTYHLB<1
174 ...|QTYHLB=1
175 ...|END
176 ...|TOTALBLKS=QTYHLB+QTYLLB
177 ...|END
178 ...CONVERSE MAP
179 ...STRUDEPTH=ZERO:ASTRUDEPTH=ZERO
```

```
180   ...TOTALBLKS=ZERO:ATOTALBLKS=ZERO
181   ...UNIQUEFACTOR=ZERO:AUNIQUEFACTOR=ZERO:LLBYTES=ZERO:ALLBYTES=ZERO
182   ...SPANFACTOR=ZERO:ASPANFACTOR=ZERO:ENTRIESLLB=ZERO:AEPPLB=ZERO
183   ...ENTRIESHLB=ZERO:AEPHLB=ZERO
184   ...QTYLLB=ZERO:AQTYLLB=ZERO:QTYHLB=ZERO:AQTYHLB=ZERO
185   ...ATTRIBUTE(MAP)="RESET"
186   ..ELSE
187   ...ERRORCD=ZERO
188   ..END
189   .END
190   .IF DOLEVEL=ZERO
191   ..STOP
192   .END
193   EXIT
```

The following two screens show the MANTIS screen layout plan for the preceding program.

MANTIS SCREEN																					
NAME : B-INDEX-BLKS	DESCRIPTION : CALCULATE INDEX BLOCKS REQ'D. FOR SECONDARY KEY										PASSWORD : BEEZ										
FRMT : NEW	FULL DISPLAY: NO		PROT BOT LINE : NO		ALARM : NO		SCREEN DOMAIN : (22,80)														
DATA	FLD	POS	FLD	VER	REP	HOR	REP	PRO	AUT	INS	MOD	PEN	REV	--BOX--							
-----FIELD NAME-----	TYPE---	ROW	COL		LEN	OCC	DIS	OCC	DIS	TCT	SKP	CUR	TAG	DET	VID	BLI	HIG	INTENS	COLOR	O	L
R U																					
"D Y N A M I C I N D E X"	HEADING	128	25					YES							BRIGHT			NO			
"ESTIMATE BLOCKS NEEDED ON IN	HEADING	213	55					YES							NORMAL			NO			
"INDEX FILE BLKSIZE"	HEADING	42	18					YES							NORMAL			NO			
BLKSIZE	NUMERIC	421	5											NORMAL				NO			
"ACTIVE RCD. ON PDM USER DATA	HEADING	427	33					YES							NORMAL			NO			
ACTIVERCD	NUMERIC	461	9					YES							NORMAL			NO			
DENSITY	NUMERIC	471	6				YES	YES							NORMAL			NO			
"MAX. RECDS. " " " "	HEADING	527	31					YES							NORMAL			NO			
MAXRCD	NUMERIC	561	9					YES							NORMAL			NO			
" / DENSITY"	HEADING	571	9				YES	YES							NORMAL			NO			
"MAX. UNIQUE KEY VALUES IN THE	HEADING	62	58					YES							NORMAL			NO			
UNIQUEKEY	NUMERIC	661	9					YES							NORMAL			NO			
"	HEADING	671	1					YES							NORMAL			NO			
MESG	TEXT	7	279				YES							BRIGHT				NO			
"SECONDARY KEY LGTH"	HEADING	82	18					YES							NORMAL			NO			
SECKEYLGTH	NUMERIC	821	5					YES							NORMAL			NO			
"PACKING DENSITY"	HEADING	827	15					YES							NORMAL			NO			
PACKING	NUMERIC	843	6					YES							NORMAL			NO			
" "	HEADING	850	1				YES	YES							NORMAL			NO			
"MAXIMUM ACTIVE"	HEADING	862	18					YES							BRIGHT			NO			
"IF PDM USER DATA FILE IS KSD	HEADING	92	55					YES							NORMAL			NO			
UNIQUEFACTOR	NUMERIC	960	11					YES	YES						NORMAL			NO			
AUNIQUEFACTOR	NUMERIC	972	9				YES	YES	YES						NORMAL			NO			
"WHAT IS PRIMARY KEY LGTH?"	HEADING	10	225					YES							NORMAL			NO			
KSDSKEYLGTH	NUMERIC	10	29	4											NORMAL			NO			
" SPANFACTOR"	HEADING	10	34	16				YES							NORMAL			NO			
SPANFACTOR	NUMERIC	10	60	11				YES	YES						NORMAL			NO			
ASSPANFACTOR	NUMERIC	10	72	9				YES	YES						NORMAL			NO			
"AVG. LGTH LOW LEVEL ENTRY"	HEADING	11	32	25				YES	YES						NORMAL			NO			
LLBYTES	NUMERIC	11	60	10				YES	YES						NORMAL			NO			
ALLBYTES	NUMERIC	11	71	10				YES	YES						NORMAL			NO			
"SECONDARY KEY NAME"	HEADING	12	218					YES							NORMAL			NO			
" SECKEYNAME	TEXT	12	21	8										NORMAL				NO			
" "	HEADING	12	30	1				YES							NORMAL			NO			
"MAX."	HEADING	12	32	4				YES							BRIGHT			NO			
"ENTRIES/HIGH LEVEL BLK.-"	HEADING	12	37	25				YES	YES						NORMAL			NO			
ENTRIESHLB	NUMERIC	12	63	7				YES	YES						BRIGHT			NO			
AEPHLB	NUMERIC	12	74	7				YES	YES						NORMAL			NO			
"AVG."	HEADING	13	32	4				YES	YES						BRIGHT			NO			
" /LOW " " --"	HEADING	13	37	25				YES	YES						NORMAL			NO			
ENTRIESLLB	NUMERIC	13	63	7				YES	YES						BRIGHT			NO			
AEPLLB	NUMERIC	13	74	7				YES	YES						NORMAL			NO			
"INDEX FILE NAME"	HEADING	14	215					YES							NORMAL			NO			
INDXFILENAME	TEXT	14	21	4										NORMAL				NO			
" "	HEADING	14	26	1				YES							NORMAL			NO			
"DEPTH"	HEADING	14	32	5				YES							BRIGHT			NO			
"OF INDEX TREE STRUCTURE-"	HEADING	14	38	25				YES	YES						NORMAL			NO			
STRUDEPTH	NUMERIC	14	64	3				YES	YES						BRIGHT			NO			
ASTRUDEPTH	NUMERIC	14	74	3				YES	YES						BRIGHT			NO			

MANTIS SCREEN																				
NAME : B-INDEXT-BLKS		DESCRIPTION : CALCULATE INDEX BLOCKS REQ'D. FOR SECONDARY KEY										PASSWORD : BEEZ								
FRMT : NEW		FULL DISPLAY: NO		PROT BOT LINE : NO		ALARM : NO		SCREEN DOMAIN : (22,80)												
-----FIELD NAME-----	DATA TYPE---	FLD ROW	POS COL	FLD LEN	VER OCC	REP DIS	HOR OCC	REP DIS	PRO TCT	AUT SKP	INS CUR	MOD TAG	PEN DET	REV VID	B LI	HIG	INTENS	--BOX--	COLOR	O L
R U																				
"MAX. INDICES"	HEADING	15 2	12					YES										NORMAL	NO	
DISPLAYNAME	TEXT 15	15 4					YES											NORMAL	NO	
"FILE-"	HEADING	15 20	6					YES										NORMAL	NO	
NOOFSECKEY	NUMERIC	15 27	4				YES											BRIGHT	NO	
"NUMBER OF HIGH LEVEL BLKS.-"	HEADING	15 32	28					YES										NORMAL	NO	
QTYHLB	NUMERIC	15 62	8				YES	YES										NORMAL	NO	
AQTYHLB	NUMERIC	15 73	8				YES	YES										NORMAL	NO	
"PDM USER DATA FILE"	HEADING	16 2	18					YES										NORMAL	NO	
DATAFILENAME	TEXT 16	21 4					YES											NORMAL	NO	
" " " LOW " " --"	HEADING	16 34	26					YES										NORMAL	NO	
QTYLLB	NUMERIC	16 61	9				YES	YES										NORMAL	NO	
AQTYLLB	NUMERIC	16 72	9				YES	YES										NORMAL	NO	
"SECONDARY KEY STRUCTURE"	HEADING	17 2	23					YES										NORMAL	NO	
"INT. FLD. NAMES BELOW 8 MAX."	HEADING	18 2	28					YES										NORMAL	NO	
"TOTAL INDEX"	HEADING	18 32	11					YES										BRIGHT	NO	
"FILE"	HEADING	18 44	4					YES										NORMAL	NO	
"BLOCKS---"	HEADING	18 49	10					YES										BRIGHT	NO	
TOTALBLKS	NUMERIC	18 60	10					YES	YES									BRIGHT	NO	
ATOTALBLKS	NUMERIC	18 71	10					YES	YES									BRIGHT	NO	
FLD1	TEXT 19	2 8																NORMAL	NO	
FLD2	TEXT 19	11 8																NORMAL	NO	
FLD3	TEXT 19	20 8																NORMAL	NO	
FLD4	TEXT 19	29 8																NORMAL	NO	
FLD5	TEXT 19	38 8																NORMAL	NO	
FLD6	TEXT 19	47 8																NORMAL	NO	
FLD7	TEXT 19	56 8																NORMAL	NO	
FLD8	TEXT 19	65 8																NORMAL	NO	
" "	HEADING	19 74	1				YES											NORMAL	NO	
"PF1 TO INSERT OR MODIFY"	HEADING	20 2	23					YES										NORMAL	NO	
"PF2 TO DELETE"	HEADING	21 2	40					YES										NORMAL	NO	

The following shows the result of the previous screen plan.

```

.....1.....2.....3.....4.....5.....6.....7.....8
.
.           D Y N A M I C   I N D E X
. ESTIMATE BLOCKS NEEDED ON INDEX FILE FOR SECONDARY KEY
.
. INDEX FILE BLKSIZE ##### ACTIVE RCD. ON PDM USER DATA FILE #####Z #.####
+           MAX. RECDS. " " " " " " #####Z / DENSITY +
. MAX. UNIQUE KEY VALUES IN THE INDEX .....#####Z
. #####
. SECONDARY KEY LGTH #####Z PACKING DENSITY #.####          MAXIMUM      ACTIVE
. IF PDM USER DATA FILE IS KSDS,          UNIQUENESS FACTOR #####.# #####Z
1 WHAT IS PRIMARY KEY LGTH?  #####          SPANFACTOR          #####.# #####Z 1
.
.           AVG. LGTH LOW LEVEL ENTRY #####Z #####Z
. SECONDARY KEY NAME #####Z          MAX. ENTRIES/HIGH LEVEL BLK.-> #####Z #####Z
.
.           AVG. " /LOW " " " --> #####Z #####Z
. INDEX FILE NAME      #####          DEPTH OF INDEX TREE STRUCTURE-> ###      ###
+ MAX. INDICES ##### FILE-> ##### NUMBER OF HIGH LEVEL BLKS.-> #####Z #####Z +
. PDM USER DATA FILE #####          " " LOW " " " --> #####Z #####Z
. SECONDARY KEY STRUCTURE
. INT. FLD. NAMES BELOW 8 MAX. TOTAL INDEX FILE BLOCKS---> #####Z #####Z
. #####
2 PF1 TO INSERT OR MODIFY
. PF2 TO DELETE          PA2 TO EXIT
.
.....1.....2.....3.....4.....5.....6.....7.....8

```

Index

A

- abends
 - PDM 62
 - PDM or CICS 57
 - task 2816 95, 106
 - task recovery 72, 73
- access definition, view 95, 96
- ACTV task table, PDM 76, 111
- allocating buffer pools 51
- applications
 - RDM, statistics 46
 - sample for index space 152
 - tuning
 - block size 54
 - buffers 51
 - CSIPARM parameters 57
 - DBAID 102
 - using statistics 21
 - using RSTAT and SHOWX 62
- attached or central PDM 57, 66, 70
- Audit Trail logs, CICS Connector 47

B

- Basic File Information report 34
- batch
 - DBAID 93, 106
 - RDM 93
- BDAM. *See* files
- benefits of tuning 17
- binding views 99
- bit maps, primary and related
 - files 58, 64, 109
- block size, tuning 54
- bootmod memory pool, files
 - category 112
- bound data list, memory pool 111
- bound views 99

- Buffer Cache Facility, OS/390 77
- buffer pools, allocating 51
- buffers
 - allocation 51
 - counts 50
 - tuning 50
- Buffers
 - tuning 77

C

- C\$MDMLNK macro for Directory Maintenance 139
- C\$VOOPTM macro for RDM 93, 95
- capacity of files 61
- CFUL status code
 - defining tasks 92
 - OPER CONNECT retry parameters 86
 - STATUS command 47
 - TASKS parameter 82
- Chain Length Statistics report
 - for primary files 37
 - for related files 38
- Chain Migration Statistics report
 - for primary files 39
 - for related files 40
- CHAINED foreign key 71
- chained records, reorganizing 65
- channel usage
 - block size 54
 - buffer caching 78
 - device placement 63
 - System Log File 60
- CICS Connector
 - and CMXT 83
 - and DFHDCT 47
 - and DFHRMCAL 87
 - and DFHSIP 146
 - and DFHSIT 83, 87
 - and MXT 82, 83, 92
 - and PDM 66, 81, 90
 - and T.R.U.E. 87
- CFUL status code
 - defining tasks 92
 - retry parameters 86
 - STATUS command 47
 - TASKS parameter 82

- CICS Connector (*cont.*)
 - CONNECT command
 - defining tasks 90
 - MEME parameter 85
 - obtaining statistics 47
 - tuning OPER CONNECT
 - parameters 82
 - CSTXOPRM macro 47, 82, 87
 - ICOR status code
 - retry parameters 86
 - STATUS command 47
 - MEME parameter 85, 146
 - messages 47
 - OPER CONNECT retry
 - parameters 86
 - statistics 47, 48
 - status automatically retried 47
 - STATUS command 47, 82
 - system tables 47, 83
 - tasks 90, 91, 112
 - TASKS parameter 84
 - TFUL status code
 - defining tasks 92
 - MXT 82
 - retry parameters 86
 - STATUS command 47
 - threads 83, 90, 91
 - tracing and synch 87
 - CICS Connector Activity Audit
 - Trail logs 47
 - CLUSTERED foreign key 71
 - CMXT, and CICS Connector 83
 - command-related memory
 - requirements, PDM 113, 130
 - commits 52, 63, 72, 73
 - common and local nodes
 - of multiple open views 98
 - of open views 94
 - of views 100
 - of views bound 99
 - concurrent tasks and threads 63, 82, 90, 112
 - concurrent updates, handling 72
 - CONNECT command
 - defining tasks 90
 - MEME parameter 85
 - obtaining CICS Connector statistics 47
 - tuning OPER CONNECT
 - parameters 82
 - contention, record, avoiding 63
 - control block memory, PDM 107
 - CPU usage
 - global views 100
 - goals 18
 - XPCC 146
 - Create Environment Description utility 51
 - CSIPARM file
 - and XPCC 144
 - parameter tuning 57, 64, 76
 - CSTX messages, CICS Connector 47
 - CSTXOPRM macro for CICS Connector 47, 82, 87
 - Current File Size report 35
- D**
- database design 71, 96
 - DBAID usage
 - BIND 99
 - BIND BOUND 99
 - defining views 96
 - finding view size 98
 - RDM task 90
 - SAVE 99
 - SHOW-NAVIGATION 96
 - slot and heap 93, 95, 106
 - statistics 46
 - tuning 102
 - UNDEFINE 106
 - density of files 64, 148
 - DFHDCT, and CICS Connector 47
 - DFHRMCAL, and CICS Connector 87
 - DFHSIP, and CICS Connector 146
 - DFHSIT, and CICS Connector 83, 87
 - direct buffer count 110
 - Directory files
 - boot memory pool
 - requirements 112
 - buffering 51
 - file placement 80

Directory Maintenance
 C\$MDMLNK macro 139
 changing buffer pool
 parameters 51
 changing file capacity 61
 changing file definitions 54
 changing Maximum Held
 Records parameter value
 72
 changing System Log File(s) 55
 database design 71
 defining file capacity 61
 defining file definitions 54
 displaying exit names 73
 file placement 80
 installing 79
 memory requirements 139
 multiple load modules 79, 139
 optimizing block size 56
 using batch 80
 dispatch priority 67
 domain and dispatch priority 67
 DSA memory. *See* memory
 allocation

E

environment description,
 MEMORY parameter 107
 ESDS *See also* files
 memory requirements 132
 RPL space 113, 130
 EXEC statement, SIZE 146
 Execution Statistics utility
 description 23, 29
 uses 51, 56
 exit. *See* user exits
 Extended Storage Support
 PDM 58, 72, 75
 RDM 94, 101

F

file experience table 58, 64, 109
 File Statistics function, DBA
 utilities
 Basic File Information report 34

File Statistics function, DBA
 Utilities
 Chain Length Statistics report
 37, 38
 Chain Migration Statistics
 report 39, 40
 Current File Size report 35
 description 23, 33
 Linkpath Statistics report 36
 Record Code Statistics report
 43
 Synonym Statistics report 42
 uses 56, 65
 files
 bit map 58, 64, 109
 block size 54
 buffering 50
 capacity 61
 control blocks 107
 density of 64, 148
 ESDS
 memory requirements 132
 RPL space 113
 experience table 58, 64, 109
 KSDS
 and MEMORY parameter 111
 buffering 52
 memory requirements 132
 reorganizing 65
 RPL space 113, 130
 statistics notes 34, 35, 36
 memory pool 107
 navigating with views 96
 placement 63
 reorganizing 65
 foreign keys. *See* keys
 formulas
 block sizes 54
 buffer counts 50
 CICS Connector tasks and
 threads 83
 command-related memory 114,
 130
 experience table 64
 file capacities 61
 memory pool 107
 VSAM control blocks 132
 free space, KSDS 65

G

- general tuning considerations 20
- GETMAIN parameter, of
 - C\$VOOPTM macro 94
- global heap, RDM
 - allocating 94
 - building 100
 - common node amount 98
 - DBAID statements 103
 - using 94
- global views, RDM
 - bound 99
 - determining view size 98
 - global 100
 - storage 93
- GLOBSIZ parameter, of
 - C\$VOOPTM macro 94
- goals of tuning 18
- guidelines for tuning 17

H

- HASHED control key 71
- heap rolling, RDM 90, 105
- Heap rolling, RDM 94
- HEAP# parameter, of
 - C\$VOOPTM macro 94
- HEAPSZ parameter, of
 - C\$VOOPTM macro 94, 95, 105
- held records 63, 72, 73
- hit rate percentage
 - obtaining file reports 26, 31, 32
 - statistics 53

I

- I/O usage
 - block size 54
 - buffers 51, 52
 - file experience table 64
 - global views 100
 - goals 18
 - index files 74
 - optimizing logical and physical
 - 53, 56
 - overlapped 73
 - with buffer caching 78
- ICOR status code
 - retry parameters 86
 - STATUS command 47

- index files
 - block size 55
 - buffering 50, 53
 - calculating space 147
 - defining 74
- Index files
 - reorganizing 65
- INDEXED foreign key 71
- in-memory hit percentage
 - obtaining file reports 26, 31, 32
 - statistics 53
- installing
 - Directory Maintenance 79
 - RDM 89
- INSUFFICIENT RESOURCES
 - message, RDM 92
- interactive services
 - description 23
 - obtaining statistics 25
 - tuning 51, 72
- Interactive Services File Statistics 56
- interval service (ISV) 67
- Interval Service (ISV) 70
- ISV. *See* interval service (ISV)

K

- keys
 - foreign
 - and design 71
 - in RDM views 96, 104
 - optimizing design 71
 - secondary
 - block size 55
 - buffering 50, 53
 - calculating space 147
 - defining 74
 - reorganizing 65
- KSDS *See also* files
 - and MEMORY parameter 111
 - buffering 52
 - memory requirements 132
 - reorganizing 65
 - RPL space 113, 130
 - statistics notes 34, 35, 36

L

Linkpath Statistics report 36
 linkpath tuning
 calculating buffer count value
 50
 calculating file capacity 61
 correcting fragmented chains
 65
 defining block sizes 54
 Load function, DBA Utilities 51,
 65
 local and common nodes
 of multiple open views 98
 of open views 94
 of views 99
 Log Print function, DBA Utilities
 44, 55, 60
 logging options 59
 logical unit of work 21, 59, 73
 long text records, global views
 101

M

macros
 C\$MDMLNK macro, Directory
 Maintenance 139
 C\$VOOPTM macro, RDM 90,
 93
 CSTXOPRM macro, CICS
 Connector 47, 82, 87
 MAXARG, for XPCC 144
 MAXIO, for XPCC 144
 MAXPACK and CICS Connector
 82, 85
 MAXPACKET
 and CICS Connector 146
 for XPCC 58, 144
 MEME parameter, CICS
 Connector 85, 146
 memory allocation
 CICS and PDM 66
 RDM heaps 93
 RDM slots 93
 RDM stacks 93
 VSAM files 132
 XPCC 146

MEMORY parameter
 acquiring storage 107
 freeing storage 75
 pool of memory 75
 reducing buffers 77
 requirements 110
 memory pool 107, 132
 memory requirements
 CICS Connector 142
 Directory Maintenance 139
 PDM
 command-related 113, 130
 memory pool calculations 107
 with VSAM files 138
 VSE using XPCC 146, 147
 Memory requirements
 VSAM files 132
 messages, CICS Connector 47
 Modify Schema utility 54, 61
 multiple load modules, Directory
 Maintenance 79, 139
 multiple open and reopen, views
 102
 MXT, and CICS Connector 82,
 83, 92
 defining tasks 92

N

naming a view 94, 102
 navigation, view 96
 nodes, common and local
 of multiple open views 98
 of open views 94
 of views 100
 of views bound 99

O

OPEN command 103
 opening a view 99
 ORDERED control key 71
 out-of-block synonyms
 checking for 65
 monitoring 42
 setting size 54, 56
 overlapping
 DML functions 68
 I/O 73

P

page-size parameter, CSIPARM
76

paging rate 75, 78, 89

PDM Extended Storage Support,
OS/390 75

PDM statistics. *See* statistics

PDM. *See* Physical Data
Manager (PDM)

PDML commands. *See* commits;
RSTAT; SHOWX

Physical Data Manager (PDM)
160

- memory pool calculations 107
- tasks 63, 91, 112
- threads 63, 91
- tuning 49

populating indexes 53, 65, 74

priority settings 67

pseudoconversational tasks

- CICS Temporary Storage 94
- HEAP# parameter 90
- RELEASE command 105
- view-naming convention 103

R

RDM Extended Storage option
94, 101

RDM Extended Storage Support
option 75

RDM. *See* Relational Data
Manager (RDM)

RDML commands

- COMMIT 73
- stacks for 93

RDMUSR# parameter, of
C\$VOOPTM macro 90

Record Code Statistics report 43

record contention, avoiding 63

Recover function, DBA Utilities
55

Relational Data Manager (RDM)

- application tuning 102
- C\$VOOPTM macro 90, 93
- defining views 96
- global heap 93
- installing 89
- number of task heaps 90, 93
- number of tasks 90

- stack memory 93
- statistics 46
- tasks 92

RELEASE command 103, 105

releasing a view 94, 100, 103,
105

reopen and multiple open, views
102

Reorganize function, DBA
Utilities 65

reorganizing files 65

resource allocation 22

response time

- and priority 67
- and user exits 73
- CICS Connector 83, 86
- goals 18
- stalls 72

rolling task heaps, RDM 90, 94,
105

RPL space, VSAM files 113, 130

RPTSIZE parameter, of
C\$VOOPTM macro 94, 106

RSTAT DML command

- description 23, 28
- effect on reports 27, 29, 30
- obtaining file reports 26
- statistics 31
- uses 62, 72

S

secondary keys. *See* keys

selective logging 59

serial buffer count 110

serial thread count 110

Shared Virtual Area (SVA) 89

SHOW-NAVIGATION, DBAID 96

SHOWX DML command

- description 23, 28
- uses 65, 72

size

- CICS memory 66, 94
- DBAID slot 106
- file blocksize 54
- files 61
- RDM global heap 94
- RDM task heaps 94
- views 98

size of memory 146

Sorted-Populate utility 53, 65, 74

spanning, on system log 44, 60
 stack, RDML command 93
 STACKSZ parameter, of
 C\$VOOPTM macro 94
 stalls 72, 73
 statistics
 obtaining
 CICS 48
 CICS Connector 47
 Execution Statistics utility 29
 file reports 26, 31, 32, 33
 Interactive Services 25
 PDM 23
 PDM system reports 27, 30
 RDM 46
 RSTAT and SHOWX 23, 28
 tuning with
 block usage 56, 60
 buffer caching 78
 buffers 51
 commands per cycle 67
 concurrent updates 72
 extended storage 75
 hits 53
 log waits 52, 55
 monitoring 23
 reorganizing 65
 status codes, CICS Connector
 86, 92
 STATUS command, CICS
 Connector 47, 82
 subsetting a view 104
 SVA. *See* Shared Virtual Area
 (SVA)
 synchronization, CICS Connector
 88
 Synonym Statistics report 42
 System Log File
 block size 55
 buffering 52
 capacity 62
 printing 44
 selective logging 59
 System Log Print function, DBA
 Utilities. *See* Log Print
 function

T

T.R.U.E., and CICS Connector
 87
 task heaps, RDM 98
 allocating 94
 common node amount 98
 DBAID statements 103
 using 94
 Task Log File
 block size 55
 buffering 51, 52
 capacity 62
 selective logging 59
 tasks
 CICS Connector
 concurrent 112
 defining 90
 reserved 91
 PDM
 concurrent 112
 defining 90
 reserved 91
 tuning number of 63
 RDM 90, 92
 TASKS parameter, CICS
 Connector 84
 Temporary Storage File, CICS 94
 TFUL status code
 CICS Connector and MXT 82
 defining tasks 92
 retry parameters 86
 STATUS command 47
 threads
 CICS Connector 83
 concurrent 112
 defining 90
 reserved 91
 PDM 108
 concurrent 112
 defining 90
 reserved 91
 tuning number of 63
 trace table, PDM 128
 tracing and synch, CICS
 Connector 87

tuning

- allocating resources 22
- applications
 - block size 54
 - buffers 51
 - CSIPARM parameters 57
 - DBAID 102
 - using statistics 21
- benefits and guidelines 17
- block size 54
- buffer pools 50
- buffers 77
- design considerations 71
- file placement 63
- individual components
 - CICS Connector 81
 - Directory 79
 - PDM 49
 - RDM 89
- planning 21
- using statistics 23
- tuning considerations, general 20

U

- uncommitted records 73
- UNDEFINE, DBAID 106
- Unload function, DBA Utilities 51, 65
- user exits
 - from Log Print function 44
 - from system file I/O 73
- User exits
 - from CICS 87
- user view names 102
- User view names 94
- utilities
 - Create Environment
 - Description 51
 - DBA Utilities, functions
 - File Statistics
 - description 23, 33
 - uses 56, 65
 - Load 51, 65
 - Log Print 55, 60
 - Recover 55
 - Reorganize 65
 - Sorted-Populate 53, 65
 - Unload 51, 65

Execution Statistics

- description 23, 29
- uses 51, 56
- Interactive Services
 - description 23
 - uses 51, 56, 72
- Modify Schema 54, 61

V

views

- binding 99
- common and local nodes 100
 - bound 99
 - multiple open 98
 - open 94
- defining 96
- global 100
- naming 94, 102
- navigation 96
- releasing 103, 105
- size 98
- subsetting 104
- UNDEFINE, DBAID 106
- using 105
- VSAM files 132

W

- waits 52, 55, 73

X

- XAMEM parameter, CSIPARM
 - 58, 72, 75
- XECB, for VSE 57
- XPCC, for VSE 57, 144