

Cincom

SUPRA SERVER PDM

Windows Client Support User's Guide

P26-7500-02



SUPRA Server PDM Windows Client Support User's Guide

Publication Number P26-7500-02

© 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This document contains unpublished, confidential, and proprietary information of Cincom.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 Windows Client Support User's Guide*, P26-7500-02, is dated January 15, 2002. This document supports Release 1.2 of SUPRA Server PDM Windows Client Support and requires the following releases be installed:

- ◆ Release 1.3.1 of SUPRA Server PDM with UNIX support
- ◆ Release 2.7 of SUPRA Server PDM with IBM support
- ◆ Release 2.4 of SUPRA Server PDM with VMS support



The Server portion of the Windows Client Support component already exists on the OpenVMS and UNIX platforms, and only needs to be installed on the OS/390 platform.

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 PDM Support
55 Merchant Street
Cincinnati, OH 45246-3732
U.S.A.



Contents

About this book	vii
Using this document.....	vii
Document organization	vii
Revisions to this manual	viii
Conventions	ix
SUPRA Server documentation series	xii
PDM support (OS/390 and VSE)	xii
PDM support (VMS and UNIX)	xiv
Windows Client Support	17
Overview	17
Getting started	19
Site requirements	19
Client site requirements	19
Server site requirements	19
Client component	20
Installing the client component.....	20
Tailoring the client component	23
Server components	30
Installing the OS/390 server component.....	30
Tailoring and starting the OS/390 server component	31
Tailoring and starting the VMS server.....	38
Tailoring and starting the UNIX server	43

Programming with Windows Client Support	45
Building C, C++, and COBOL programs	45
Building Visual Basic programs	46
Building Java programs	49
Data conversion	50
Manual data conversions	50
Automatic data conversions	51
Entry points	54
DML considerations	60
Sample coding techniques	61
Turning READAHEAD off/on	61
Using datbasConnectID to connect to the host (without using datbas.ini)	61
Using multiple datbasConnectID functions to SINON to multiple databases from the same application	61
Using datbasSetCurrentID	62
Using datbasGetCurrentID	62
Using datbasConvertToText	62
 Error messages	 63
SUPRA PDM Windows Client messages	63
SUPRA PDM Windows Server messages	68
 Troubleshooting	 79
Location of the datbas.ini file	79
Unable to connect to Unix server	79
Debugging SUPRA Server PDM Windows Client Support	80
 Translation tables	 81
EBCDIC to ASCII default translation table	81
ASCII to EBCDIC default translation table	82
 Index	 83

About this book

Using this document

Document organization

The information in this manual is organized as follows:

Chapter 1—Windows Client Support

Describes the key components in the Windows Client Support application.

Chapter 2—Getting started

Describes how to start and exit Windows Client Support.

Chapter 3—Programming with Windows Client Support

Discusses building programs in C, C++, COBOL, and Visual Basic, data conversions, entry points and DML considerations.

Chapter 4—Error messages

List the messages that can be returned by the SUPRA[®] PDM Windows Client components and the SUPRA PDM Windows Server components.

Chapter 5—Troubleshooting

Provides suggestions for resolving some error situations, if they occur.

Index

Revisions to this manual

The following changes have been made for this release:

- ◆ A new section on building Java programs has been added. See "[Building Java programs](#)" on page 49.
- ◆ New information about security codes has been added in chapter 2. See "[Step 8: Activating SUPRA Server PDM Windows Client Support](#)" on page 22.
- ◆ A new section was added regarding enabling Serial ReadAhead Cache in the section on tailoring the client. See "[Enabling Serial ReadAhead Cache](#)" on page 29.
- ◆ Several server init file parameters have been added. See "[Server init file parameters](#)" on page 33 for a complete list.
- ◆ The following entry points have been added:
 - [DATBAS\(\)](#) on page 55
 - [datbas_putenv\(\)](#) on page 56
 - [datbasGetConnectParams\(\)](#) on page 57
- ◆ The following error messages have been added:
 - [CSTJ111I](#) on page 66
 - [CSTJ112E](#) on page 66
 - [CSTJ113F](#) on page 66
 - [CSTJ114I](#) on page 66
 - [CSTJ114W](#) on page 67
 - [CSTJ115F](#) on page 67
 - [CSTJ116F](#) on page 67
 - [CSTJ117I](#) on page 67
 - [CSTJ118F](#) on page 67
- ◆ A new section was added to show sample coding techniques. See "[Sample coding techniques](#)" on page 61.
- ◆ A new section was added. "[Troubleshooting](#)" on page 79 describes several problems that may be encountered and gives the user an explanation of the situation.
- ◆ An appendix has been added, listing translation tables. See "[Translation tables](#)" on page 81.

Conventions

The following table describes the conventions used in this document. These conventions will help you identify statements, commands, and references within the text and software.

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

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>[<u>(WAIT)</u>] [<u>(NOWAIT)</u>]</p> <hr/> <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><i>INTO :host-variable [:ind-variable],...</i></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
	In the UNIX operating environment, keywords are case-sensitive, and you must enter them exactly as shown.	cp *.QAR /backup
<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;">UNIX</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">VMS</div>	Information specific to a certain operating system is flagged by a symbol in a shadowed box (UNIX) 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;">UNIX</div> To delete these files, return to the shell and use the rm command. <div style="border: 1px solid black; padding: 2px; display: inline-block;">VMS</div> To delete these files, return to the command level and use the DELETE command.

SUPRA Server documentation series

SUPRA Server is the advanced relational database management system for high-volume, update-oriented production processing. The following list shows the manuals and tools used to fulfill the data management and retrieval requirements for various tasks. Some of these tools are optional. Therefore, you may not have all the manuals listed. For a brief synopsis of each manual, refer to the *SUPRA Server PDM Digest (OS/390 & VSE)*, P26-9062, or the *SUPRA Server PDM Digest for VMS Systems*, P25-9062.

PDM support (OS/390 and VSE)

Overview

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

Getting started

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

General use

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

Database administration tasks

- ◆ *SUPRA Server PDM and Directory Administration Guide (OS/390 & VSE)*, P26-2250
- ◆ *SUPRA Server PDM Directory Online User's Guide (OS/390 & VSE)*, P26-1260
- ◆ *SUPRA Server PDM Directory Batch User's Guide (OS/390 & VSE)*, P26-1261
- ◆ *SUPRA Server PDM DBA Utilities User's Guide (OS/390 & VSE)*, P26-6260
- ◆ *SUPRA Server PDM Logging and Recovery (OS/390 & VSE)*, P26-2223

- ◆ *SUPRA Server PDM Tuning Guide (OS/390 & VSE)*, P26-0225
- ◆ *SUPRA Server PDM RDM Administration Guide (OS/390 & VSE)*, P26-8220
- ◆ *SUPRA Server PDM RDM PDM Support Supplement (OS/390 & VSE)*, P26-8221
- ◆ *SUPRA Server PDM RDM VSAM Support Supplement (OS/390 & VSE)*, P26-8222
- ◆ *SUPRA Server PDM Migration Guide (OS/390 & VSE)*, P26-0550*
- ◆ *SUPRA Server PDM Windows Client Support User's Guide*, P26-7500*
- ◆ *SPECTRA Administrator's Guide*, P26-9220

Application programming tasks

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

Report tasks

- ◆ *SPECTRA User's Guide*, P26-9561

PDM support (VMS and UNIX)

Overview

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

Getting started

- ◆ *SUPRA Server PDM UNIX Installation Guide*, P25-1008
- ◆ *SUPRA Server PDM VMS Installation Guide*, P25-0147
- ◆ *SUPRA Server PDM UNIX Tutorial*, T25-2262
- ◆ *SUPRA Server PDM VMS Tutorial*, T25-2263

General use

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

Database administration tasks

- ◆ *SUPRA Server PDM Database Administration Guide (UNIX & VMS)*, P25-2260
- ◆ *SUPRA Server PDM System Administration Guide (VMS)*, P25-0130
- ◆ *SUPRA Server PDM System Administration Guide (UNIX)*, P25-0132*
- ◆ *SUPRA Server PDM Utilities Reference Manual (UNIX & VMS)*, P25-6220
- ◆ *SUPRA Server PDM Directory Views (VMS)*, P25-1120
- ◆ *SUPRA Server PDM Windows Client Support User's Guide*, P26-7500*
- ◆ *SPECTRA Administrator's Guide*, P26-9220

Application programming tasks

- ◆ *SUPRA Server PDM Programming Guide (UNIX & VMS)*, P25-0240
- ◆ *SUPRA Server PDM System Administration Guide (VMS)*, P25-0130
- ◆ *SUPRA Server PDM System Administration Guide (UNIX)*, P25-0132*
- ◆ *SUPRA Server PDM RDM Administration Guide (VMS)*, P25-8220
- ◆ *SUPRA Server PDM Windows Client Support User's Guide*, P26-7500*
- ◆ *MANTIS Planning Guide*, P25-1315

Report tasks

- ◆ *SPECTRA User's Guide*, P26-9561



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



SUPRA Server educational material is available from your regional Cincom education department.

1

Windows Client Support

Overview

SUPRA Server PDM Windows Client Support is a Windows DLL that provides a `datbas()` function call interface like that provided on the IBM OS/390 & VSE, DEC OpenVMS, and various UNIX SUPRA PDM server platforms. For details on calling the `datbas()` function from application programs, refer to the *SUPRA Server PDM DML Programming Guide (OS/390 & VSE)*, P26-4340, and the *SUPRA Server PDM Programming Guide (UNIX & VMS)*, P25-0240. These programming details remain unchanged when porting application code from the server platform to the PC Windows platform and are not discussed in this document.

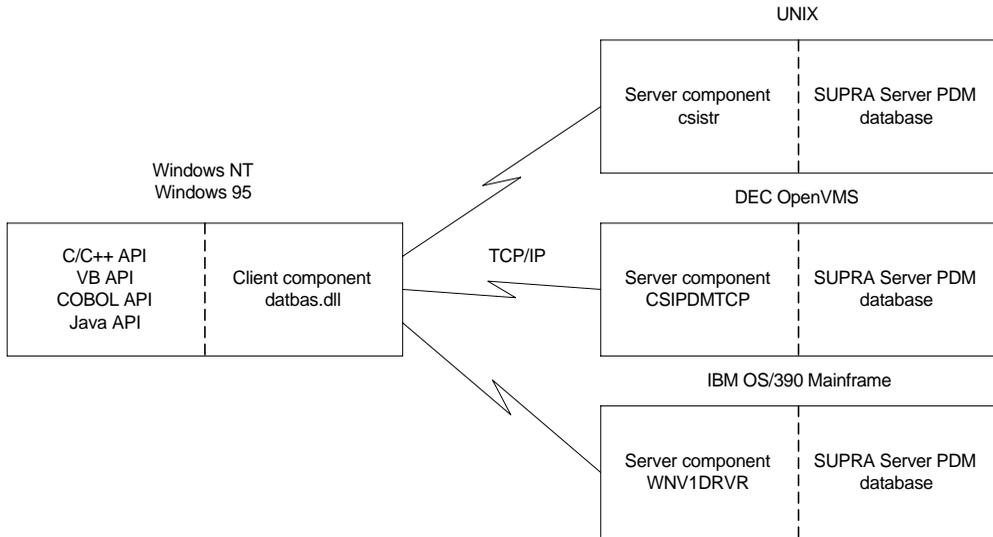
Windows Client Support allows access to PDM data for Windows applications. This access can be obtained by migrating existing host applications to the PC, or by creating new GUI applications on the PC.

SUPRA PDM Windows Client Support provides the following:

- ◆ Support for 32-bit Windows platforms
- ◆ Can be used with UNIX SUPRA, OpenVMS SUPRA, or IBM SUPRA
- ◆ Support for all DML currently supported by UNIX, OpenVMS, and IBM SUPRA `datbas()` interfaces and compatibility with current DML formats
- ◆ Support for TCP/IP to communicate with UNIX, OpenVMS, and OS/390 SUPRA servers
- ◆ Can be used with COBOL, C, C++, Visual Basic, or Java

SUPRA PDM Windows Client Support communicates with OpenVMS and UNIX servers using the TCP/IP formats and protocols currently used by the servers. In the IBM environment, a TCP/IP server is provided.

The following illustration shows the configuration and components supporting SUPRA Windows Client Support:



Applications communicate with the client component. This is implemented as a Windows DLL. The client component provides all data conversion and maintains the socket to communicate with the server component. The server component issues DML to SUPRA PDM and maintains sockets to communicate with client components.

2

Getting started

Site requirements

Client site requirements

The SUPRA PDM Windows Client component is currently supported in following environments:

- ◆ Windows NT Release 4 or higher
- ◆ Windows 95/98

Server site requirements

The SUPRA PDM Windows Server component is currently supported in following environments:

- ◆ OS/390 1.0 and higher
- ◆ OS/390/ESA 4.2 and higher
- ◆ OS/390/TCPIP
- ◆ AIX 4.3 and higher
- ◆ Digital UNIX 4.0 and higher
- ◆ HPUX 10.0 and higher
- ◆ OpenVMS 6.2 and higher

The following releases of SUPRA Server are required:

- ◆ Release 1.3.1 of SUPRA Server PDM with UNIX support
- ◆ Release 2.7 of SUPRA Server PDM with IBM support
- ◆ Release 2.4 of SUPRA Server PDM with VMS support

Client component

Installing the client component



This installation process requires Microsoft® Windows™ 95/98 or Windows NT® 4.0(Intel) and assumes that the installer has Windowing knowledge and capabilities.

The complete SUPRA Server PDM Windows Client Support installation process consists of the following general steps:

1. Starting the installation
2. Destination location
3. Setup type
4. Program folder
5. Installation summary
6. Configuring the datbas.ini initialization file
7. Activating SUPRA Server PDM Windows Client Support

This section also discusses:

- ◆ Uninstalling SUPRA Server PDM Windows Client Support
- ◆ Re-installing SUPRA Server PDM Windows Client Support

Step 1: Starting the installation

From the Start Menu select Run and type `<drive:>\disk1\setup.exe`, where `<drive:>` is the drive letter where you placed the SUPRA Server PDM Windows Client Support media.

Step 2: Destination location

PCDatbas is the default.

Step 3: Setup type

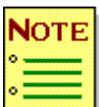
Typical	Installs all components. (Recommended for most PCs.)
Compact	Installs all components excluding Debug and Example files components. (To save disk space.)
Custom	Any combination of files can be selected for installation. Also clicking on a component gives its description. (For advanced users and system administrators only.)



Program and Library files components have Non-Debug and Debug Files as their subcomponents.

Step 4: Program folder

SUPRA PDM Client is the default.

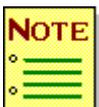


If the platform is Windows NT 4.0 and the administrator or a user with administrator privileges is logged in, then the program folder is located in the Common group, otherwise the program folder is located in the Personal group.

Step 5: Installation summary

Displays Setup type and its components, Destination directory and Program folder. The installation will copy program files, example files, library files, help files, and updates the registry. It will create three environment variables:

- ◆ CSIPDM_PATCH (See "[Step 8: Activating SUPRA Server PDM Windows Client Support](#)" on page 22)
- ◆ CSI_READAHEAD (See "[Enabling Serial ReadAhead Cache](#)" on page 29)
- ◆ CSI_READAHEAD_STATISTICS (See "[Enabling Serial ReadAhead Cache](#)" on page 29)



Any component with an asterisk (*) preceding its name is not installed for the selected Setup type.

Step 6: Select Options

Displays three options:

- ◆ Read the README file
- ◆ Read the online documentation
- ◆ Create the datbas.ini file

Step 7: Configuring the datbas.ini initialization file

See “[Tailoring the client component](#)” on page 23.

Step 8: Activating SUPRA Server PDM Windows Client Support

Before SUPRA Server PDM Windows Client Support can be used, you must define the environment variable CSIPDM_PATCH. This environment variable defines the security patch that controls the expiration date. Normally the installation process defines it for you. You can verify this by checking the environment variable settings with the Windows Control Panel or by using the DOS command "SET CSIPDM_PATCH". The value should appear as a string of five 8-character hexadecimal numbers as follows: "nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn nnnnnnnn". When the expiration date is within 30 days, the product will warn you with a CSTJ114W message advising you to contact your account representative to acquire a new security patch. The latest security message can be found in the datbas.log file contained in your current working directory.

Before accessing the SUPRA Server PDM Windows Client Support from your application, reboot your PC.

Uninstalling SUPRA Server PDM Windows Client Support

Select SUPRA PDM Client uninstall from the SUPRA PDM Client program folder.



If the platform is Windows NT 4.0 and the administrator or a user with administrator privileges performs the install, then only the administrator or a user with administrator privileges can perform the uninstall.

Re-installing SUPRA Server PDM Windows Client Support

Re-installing SUPRA Server PDM Windows Client Support on top of a previous install, with overwrite, does not remove files but overwrites them. However, if the `datbas.ini` initialization file already exists, it will not be overwritten unless you specify that a new file be created. If the `datbas.ini` initialization file does not exist, it will be created with keys but no key values unless configured while creating the `datbas.ini` file in [Step 6](#).

In addition, to overwrite a Typical or Custom Setup Type install with a Compact install, uninstallation of SUPRA Server PDM Windows Client Support is required to recover disk space.

Tailoring the client component

There are two parts to tailoring the client component:

- ◆ Creating the `datbas.ini` file
- ◆ Enabling Serial ReadAhead Cache

Creating the datbas.ini file

Datbas.ini is a text file containing 12 lines in the following format:

[datbas]

DoMessageBox =

TRUE/YES
FALSE/NO

ServerType =

IBM
MVS
VMS
OpenVMS
UNIX

ServerName = *host-name*

Transport =

TCPIP
TCP/IP
DECnet
APPC
ORB

PortInfo = *nnnn*

Csipdmid = *pdm-name*

Scope =

SYSTEM
GROUP
MULTIPLE
MULTIPLESYSTEM
MULTI_SYSTEM
MULTIPLESYSTEMWIDE
MSW

Group = *nnnnnn*

User = *nnnnnn*

Prefix = *xxx*

Transaction = *xxxx*

[datbas]

Description *Required.* The init file section name must be entered exactly as shown.

DoMessageBox =

TRUE/YES
FALSE/NO

Description *Optional.* When Windows Client encounters an exception condition, such as a communication error with the server, this setting determines whether a Windows message box is presented to the end user. In either case, a message is written to the datbas.log file.

Default TRUE

Options Values are TRUE, YES, FALSE, and NO. Any other value is treated as FALSE.

Format Value is not case sensitive and may be either upper or lower or mixed case.

ServerType =

IBM
MVS
VMS
OpenVMS
UNIX

Description *Required.* Determines one of three server types to be accessed: IBM, VMS, or UNIX.

Options Values are IBM and MVS (which are both equivalent to IBM), VMS and OpenVMS (which are both equivalent to VMS), and UNIX. Any other value causes an error condition.

Format Value is not case sensitive and may be either upper or lower or mixed case.

Consideration Results from specifying an incorrect server type are unpredictable.

ServerName = *host-name*

Description *Required.* Specifies the host name of the server machine that is running the SUPRA PDM server.

Transport =

TCPIP
TCP/IP
DECnet
APPC
ORB

- Description** *Optional.* Determines one of four transports to be used, TCPIP, DECnet, APPC, or ORB. In this release of Windows Client, only TCPIP or TCP/IP is supported.
- Default** TCPIP
- Options** Values are TCPIP and TCP/IP (which are both equivalent to TCPIP), DECnet, APPC, and ORB. Any other value causes an error condition.
- Format** Value is not case sensitive and may be either upper or lower or mixed case.
-

PortInfo = *nnnn*

- Description** *Optional.* For TCPIP transport the portinfo is the port number to be used for socket communication on the remote host, and is required. If this entry is omitted, the default value will be used, possibly resulting in subsequent errors.
- Default** 0000
- Format** Numeric
-

Csipdmid = *pdm-name*

- Description** *Optional.* This is the name of the PDM to be accessed on the remote server. On IBM servers, this is the name of the running PDM server. On VMS and UNIX servers, this is the translation of the logical name CSI_PDMID or CSI_SYSPDMID.
- Default** MISSING
- Format** Usually 8 characters

Scope =

SYSTEM
GROUP
MULTIPLE
MULTIPLESYSTEM
MULTI_SYSTEM
MULTIPLESYSTEMWIDE
MSW

Description *Optional.* For IBM servers, this is not applicable. For VMS and UNIX servers, this indicates the scope of the PDM to be accessed on the remote server.

Default SYSTEM

Options Acceptable values are SYSTEM, GROUP, MULTIPLE, MULTIPLESYSTEM, MULTI_SYSTEM, MULTIPLESYSTEMWIDE, and MSW. Any other value causes an error condition.

Format Value is not case sensitive and may be either upper or lower or mixed case.

Group = nnnnnn

Description *Optional.* For IBM servers, this is not applicable. For VMS and UNIX servers, this indicates the group number of a group wide PDM server.

Default 000000

Format Numeric decimal

Consideration OpenVMS systems use group numbers in numeric octal format. Therefore, you must convert your OpenVMS octal group number to decimal. For example, when using an OpenVMS UIC group of [000202], you must enter the following:

Group=000130

User = nnnnnn

Description	<i>Required.</i> Specifies the user id for connection to the remote host. In the initial release of Windows Client the value is unimportant and can be 0, but must be present.
Format	Numeric

Prefix = xxx

Description	<i>Optional.</i> For VMS and UNIX servers, this indicates the translation of the logical name CSI_PREFIX on the host server machine.
Default	"" (a null string)
Format	0–3 characters

Transaction = xxxx

Description	<i>Optional.</i> For IBM servers this indicates the CICS transaction id to be used.
Default	"" (a null string)
Format	1–4 characters. Only permitted for IBM servers

Enabling Serial ReadAhead Cache

If your host server platform supports the Serial ReadAhead Cache (SRAC) feature, SUPRA Server PDM Windows Client Support can take advantage of this by defining the environment variable `CSI_READAHEAD` as `TRUE`. In addition, SRAC statistics display can be enabled by defining the environment variable `CSI_READAHEAD_STATISTICS` as `TRUE` which will enable the output of the CSTJ1111 message.

SRAC retrieves a block of records from the host PDM server and caches them locally in the client for faster subsequent access. This occurs when serially reading the same dataset with no record holding. The performance benefits when sweeping a large number of records can be substantial. Refer to your host server documentation for particulars regarding use with that host platform.

- ◆ On IBM host platforms the SRAC feature is not yet supported for TCP/IP clients, and `CSI_READAHEAD` should always be set to `FALSE`. During the installation of Windows Client, the environment variable `CSI_READAHEAD` will be set to `TRUE` by default. It will be set to `FALSE` if you specify `YES` to "Create `datbas.ini` file" and specify `IBM/CICS/MVS` as the server type.
- ◆ On UNIX and VMS host platforms refer to the *SUPRA Server PDM System Administration Guide (VMS)*, P25-0130, or the *SUPRA Server PDM System Administration Guide (UNIX)*, P25-0132. Look for the sections titled, "Tuning your database," "Designing application programs," "Understanding client read-ahead buffering," "Context position considerations," and "Application programming considerations."

Server components

Installing the OS/390 server component

Locate the JCL member (DRVJRJCL) in the xxxx.WNCLIENT.JCLLIB library that you received with the SUPRA 2700 install tape. Then refer to the next section for tailoring information.



If you do not have the xxxx.WNCLIENT libraries, they can be e-mailed to you. You will receive information on how to get the datasets from the e-mail to your IBM system. For further information, contact Cincom Technical Support.

Tailoring and starting the OS/390 server component

There are certain site-specific tailoring tasks you must perform on the server component.

Tailoring the SUPRA PDM Windows Server component on OS/390 includes the following required and optional tasks:

- ◆ Modifying the JCL member (DRVRJCL)
- ◆ Defining your initialization file parameters

These tasks are discussed in the following sections.

Modifying the JCL member (DRVRJCL)

Refer to the comments in this example member, and modify as needed.

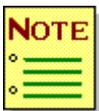
Defining your initialization parameters (OS/390)

The INIT DD may be a sequential dataset, a member of a PDS, or inline as in the example member.

- ◆ Maximum line length is 80 characters.
- ◆ Lines can be blank.
- ◆ All comment lines must begin with a pound sign (#).
- ◆ Parameters can begin anywhere on a line, but must start and end on one line.
- ◆ There can be no space between a parameter keyword and its subsequent equal sign (=). However, there can be spaces after the equal sign.
- ◆ Keywords are case-insensitive; they can be lower case, upper case, or a combination of both.
- ◆ Numbers can be specified as:

Representation	Numeric base
0xnnnn	Hexadecimal
Onnnn	Octal
nnnn	Decimal

- ◆ Enclose a text string in single quotes (*'text string'*) or double quotes (*"text string"*) if it contains whitespace characters (space, tab, newline, return, form feed). If a text string contains a single quote, enclose the string in double quotes; if a string contains a double quote, enclose the string in single quotes.
- ◆ Boolean=false can be specified by the following character options:
N/n/F/f/0
Boolean=true can be specified by the following character options:
Y/y/T/t/1



Characters following one of these option characters are ignored ("not" and "N1" would be treated as Boolean=false).

Server init file parameters

The server init file parameters on OS/390 are:

DISPLAY_OPTIONS= YES/Y/y/T/t/1
NO/N/n/F/f/0

Description	<i>Optional.</i> Specifies whether to display all initialization options and their values.	
Default	NO	
Options	YES	Display all initialization options and their values. YES can also be specified by Y/y/T/t/1.
	NO	Do not display the initialization options. NO can also be specified by N/n/F/f/0.

MAX_DATAAREA=nnnnnnnnnn

Description	<i>Optional.</i> Specifies the size of the data area for retrieving one DATBAS record.	
Default	8192	
Format	1-2,147,483,647	
Consideration	This value must be large enough to hold an entire record. The maximum value actually used is displayed at shutdown to help set this parameter.	

MAX_DATALIST=nnnnnnnnnn

Description	<i>Optional.</i> Specifies the size of the largest datalist that will be generated by the server.	
Default	4096	
Format	1-2,147,483,647	
Consideration	This value must be large enough to hold an entire datalist. The maximum value actually used is displayed at shutdown to help set this parameter.	

MAX_DATBAS_PDU=nnnnnnnnnn

- Description** *Optional.* Specifies the size of the response buffer used as the data area for CALL DATBAS functions.
- Default** 8192
- Format** 1–2147483647
- Consideration** This value must be large enough to hold all data requested by the passed element list.

MAX_RCV_PDU=nnnnnnnnnn

- Description** *Optional.* Specifies the maximum message size received by the server from the client.
- Default** 4096
- Format** 1–2147483647

PDM_THREADS=nnnnn

- Description** *Optional.* Specifies the maximum number of concurrent threads connected to the PDM.
- Default** 8192
- Format** 1–99,997
- Consideration** The server adds one thread to the specified number for its own use. This value must be less than or equal to the value for SERVER_CONNECTIONS.

SERVER_CONNECTIONS=nnnnn

- Description** *Optional.* Specifies the maximum number of concurrent client connections to the server.
- Default** 1
- Format** 1–99,997
- Consideration** The server adds two connections to the specified number for its own use. This value must be greater than or equal to the value for PDM_THREADS. Server connections are equivalent to PDM tasks.

SHARED_MEM=nnnnnnnn

Description	<i>Optional.</i> Specifies the size of shared memory requested for the schema tables and error messages.
Default	1,048,576 (1 MB)
Format	8–16,777,208 and evenly divisible by 8.
Consideration	If this value is too small, the server will not start. If the value is too large, extra memory is wasted. This memory is allocated above the 16 MB line. The value actually used is displayed at shutdown to help set this parameter.

SHUTDOWN_PASSWORD=xxxxxxxx

Description	<i>Required.</i> Specifies the password that validates server shutdown commands from the client.
Default	DBA
Format	1–8 printable characters

SHUTDOWN_USER=xxxxxxxx

Description	<i>Required.</i> Specifies the user name that validates server shutdown commands from the client.
Default	DBA
Format	1–8 printable characters

TCP_HOSTNAME_LOOKUP=YES or NO

Description	<i>Optional.</i> Specifies whether there will be a host name lookup for the IP address whenever a TCP/IP connection is established.				
Default	YES				
Options	<table> <tr> <td>YES</td> <td>Enable the lookup. YES can also be specified by Y, y, T, t, or 1.</td> </tr> <tr> <td>NO</td> <td>Do not enable the lookup. NO can also be specified by N, n, F, f, or 0.</td> </tr> </table>	YES	Enable the lookup. YES can also be specified by Y, y, T, t, or 1.	NO	Do not enable the lookup. NO can also be specified by N, n, F, f, or 0.
YES	Enable the lookup. YES can also be specified by Y, y, T, t, or 1.				
NO	Do not enable the lookup. NO can also be specified by N, n, F, f, or 0.				
Consideration	The lookup will use either a TCP/IP host file or the TCP/IP resolver. This value is ignored if TCP_IP_SUPPORT=NO.				

TCP_HOSTFILE='xxx.xxx...'

Description *Optional.* Specifies the MVS name for the file equivalent to /etc/hosts.

Default None

Format Any valid sequence of characters (enclosed in single quotes) that create a valid MVS data name.

Consideration This specification is necessary only if TCP_HOSTNAME_LOOKUP=YES and standard names are not used. This value is ignored if TCPIP_SUPPORT=NO.

TCP_LISTEN_PORT=nnnnn

Description *Optional.* Specifies the port number to be used for the listener.

Default 5010

Format 1–65,535

Consideration This value is ignored if TCPIP_SUPPORT=NO. This value must match the corresponding value in the client parameter *db_pdm-database-name*.

TCP_PREFIX=xxxx.xxxx...

Description *Optional.* Specifies a high-level qualifier added to the TCP/IP file names for TCP/IP access.

Default TCPIP

Format Any valid sequence of characters that creates a valid MVS dataname.

Consideration This value is ignored if TCPIP_SUPPORT=NO.

Example The default protocol filename is TCPIP.ETC.PROTO. The TCPIP prefix could be replaced with any valid value specified for this parameter.

TOASCII=xxxxxxx

- Description** *Optional.* Specifies the name of the load module that contains a user-defined EBCDIC-to-ASCII translation table (to replace the standard translation table).
- Default** If this parameter is not provided or has no value, the standard translation table is used.
- Format** 1–8 characters.
- Consideration** The standard translation table for EBCDIC to ASCII is provided in ["EBCDIC to ASCII default translation table"](#) on page 81.

TOEBCDIC=xxxxxxx

- Description** *Optional.* Specifies the name of the load module that contains a user-defined ASCII-to-EBCDIC translation table (to replace the standard translation table).
- Default** If this parameter is not provided or has no value, the standard translation table is used.
- Format** 1–8 characters.
- Consideration** The standard translation table for ASCII to EBCDIC is provided in ["ASCII to EBCDIC default translation table"](#) on page 82.

TRACE=nnn,nnn-nnn...

- Description** *Optional.* If specified, this parameter turns on debug tracing and specifies one or more tracing options to be used within the server component. This is a Cincom Technical Support parameter.
- Default** The parameter is not present (no tracing)
- Options** 30, 31, 38, 39, 50, 53, 54, 56, 58, 60, 61, 62, 63
- Format** One or more numbers or number ranges, separated by commas (50, 53–56, 60)
- Consideration** Tracing is usually turned on at the recommendation of Cincom Support. The specified numbers correspond to specific tracing actions and are provided by Support. When enabled, some trace options generate significant output and can degrade performance. All trace output is written to SYSPRINT DD. Tracing is only effective if you run using the xxx.WNCLIENT.DBUGLINK library.

Tailoring and starting the VMS server

SUPRAPDM/Server TCP/IP program

This program is executed by the DEC UCX TCP/IP Auxiliary server when a client request is made to the SUPRAPDM assigned port (typically 8000). The standard Auxiliary service is named SUPRA1. All socket creation/deletion is handled by the Auxiliary server. This program receives the created socket and uses it to communicate in the standard client/server loop.

Configuring a VMS Server for TCP/IP clients

The SUPRAPDM TCP/IP Server for OpenVMS is implemented as a DEC TCP/IP service similar to FTP or Telnet. This allows for flexibility in installation and monitoring of the communications resources. The server makes use of the DEC TCP/IP auxiliary server to manage the master port and create the individual client processes. Some familiarity with "DEC TCP/IP Services for VMS" (formerly known as UCX) is assumed in installation. The creation of VMS account CSIPDMTCP is also required, which may require the services of the local VMS system manager.

The requirements are as follows:

- ◆ The TCP/IP server creates one VMS process for each client connection. Maximum processes allowed on the system must be configured with this consideration in mind.
- ◆ Each client connection requires one TCP/IP socket. The DEC TCP/IP Services parameter `DEVICE_SOCKETS` must be configured for the maximum number of simultaneous sockets in use system wide, including services such as FTP, Telnet, and so on. To set this, issue the following command:

```
UCX SET COMMUNICATIONS/DEVICE_SOCKETS=xxx.
```
- ◆ By default SUPRAPDM TCP/IP clients use port 8000 as the master port. If you choose to use a port other than 8000, client systems will need to specify the port explicitly. Consult the particular client environment entries in the manuals for details of how this is performed in the various client environments.
- ◆ The TCP/IP server is used systemwide to connect to any SUPRAPDM database at the same SUPRAPDM release level. Logical names typically defined at the group level for `SUPRA_EXE`, `SUPRA_COMS`, and SUPRAPDM executables must also be defined either in the system table or the `CSI_PDM_<sypdmid>` table. This is performed by standard DCL command procedures created during SUPRAPDM installation.

To implement TCP/IP client support, perform these steps during installation or later:

1. Create VMS account CSIPDMTCP. The TCP/IP server uses the DEC TCP/IP auxiliary server for master port management. It must run from the same UIC group as other auxiliary services such as TELNET or FTP. The following sample AUTHORIZE steps create an account CSIPDMTCP within this group. Adapt them to fit your environment regarding device names, and so on.

- a. Run the AUTHORIZE utility.
- b. Type "SHOW UCX\$FTP/BRIEF". Note the group # in the UIC.
- c. Type

```
COPY UCX$FTP CSIPDMTCP
  /DEV=homedev/DIR=homedir/UIC=[xxx,yyy]
```

where:

<i>homedev</i>	=	Location of the user home directory
<i>homedir</i>	=	User home directory
<i>x,yyy</i>	=	UIC
<i>xx</i>	=	UIC group from step b above
<i>yy</i>	=	Unused UIC member within group <i>xxx</i>

- d. Create the home directory for user CSIPDMTCP.
- e. Ensure that user account CSIPDMTCP has sufficient rights and quotas for write access to the path for logging.
- f. Create an empty LOGIN.COM file in the CSIPDMTCP home directory containing only a comment line.

- Configure and start the DEC TCP/IP SUPRA1 service. A sample DCL procedure CSIPDMTCP_SETUP.COM is provided in SUPRA_COMS directory to help set up the SUPRA1 service. This procedure is run one time to create the SUPRA1 service entry in the UCX database. The following are customizable parameters:

"/FILE"	The command procedure to be invoked when the service is called. If SUPRA_COMS is not defined in the system table, this parameter must be an explicit path. The following example points to the CSIPDMTCP_RUN.COM file provided.
"/LOG"	Settings and location for the service log file. For no logging define this parameter as "NL:".
"/LIMIT"	Total number of simultaneous client TCPIP connections to all of the databases on the host system.

The command "UCX ENABLE SERVICE SUPRA1" enables the service for connections. To automatically enable the service when DEC TCP is started, either add this command to SYS\$MANAGER:UCX\$INET_SET_INTERFACES.COM file (if used on your system), or issue the command "SET CONFIGURATION ENABLE SERVICE SUPRA1".

Once the service is enabled, the command "UCX SHOW SERVICE SUPRA1/FULL" will create a display similar to the following:

```

Service: SUPRA1
State:      Enabled
Port:      8000          Protocol:  TCP          Address:  0.0.0.0
Inactivity: 5          User_name: CSIPDMTCP   Process:  SUPRA1
Limit:     100         Active:    0           Peak:    0

File:      SUPRA_COMS:CSIPDMTCP_RUN.COM
Flags:     Listen

Socket Opts: Rcheck  Scheck
Receive:    0        Send:          0

Log Opts:   Acpt Actv Dactv Conn Error Exit Logi Logo Mdfy Rjct Tim0 Addr
File:      SUPRA_COMS:CSIPDMTCP.LOG

Security
Reject msg: not defined
Accept host: 0.0.0.0
Accept netw: 0.0.0.0

```

Operational information

Once the previous steps have been followed, the SUPRA1 service is enabled and ready to establish connections. When a client request is received on port 8000, the command procedure CSIPDMTCP_RUN.COM is executed as a detached process. This command procedure runs the executable "CSIPDMTCP_xxxx.EXE", where xxxx is the SUPRAPDM release level. The detached client process is named "SUPRA1_BGxxxx", where BGxxxx is the local device/socket allocated by DEC TCP to the client connection. Upon completion of the client/server tasks, the detached process exits. Each detached process creates a log file, designated by the /LOG service parameter above. The number of log versions is dependent on the number of file versions allowed by VMS. Typically the log will contain login and logoff information, as well as minimal TCP/IP status information.

Special considerations for groupwide PDMs

You can implement one or more groupwide TCP servers either in place of or in addition to a systemwide server. To do this you would repeat the previous steps for each group with 3 changes:

1. Either identify or create a new username (other than CSIPDMTCP) within the desired group UIC to be used for the group TCP server.
2. Create another UCX service name (other than SUPRA1) which is specified to use the username created in 1) above.
3. For this new UCX service you must choose a new port number (other than 8000).

For example, to create a TCP groupwide service for group 100 with a new username NEW100USER, new service name SUPRA100, and new port 8001:

```
UAF> COPY UCX$FTP new100user /DEV=dev/DIR=dir/UIC=[100,newmember]
$ CREATE/DIR dev:[dir]
$ CREATE dev:[dir]LOGIN.COM
UCX> Set Service SUPRA100/Proc=SUPRA100/Proto=TCP/Port=8001/Limit=n -
  /File=Disk:[Directory]CsiPdmTcp_Run.Com/User=new100user -
  /Log=(All,File=Disk:[Directory]CsiPdmTcp_Run.Log)/Flag=Listen
UCX> Enable Service SUPRA100
```

Debugging

You can get extra information into the output log file by defining special logical names, CSIPDMTCP_TRACECALLS and/or CSIPDMTCP_DUMPPACKETS. These can be defined by editing CSIPDMTCP_RUN.COM or they can be defined at the group or system level, as long as the definitions can be seen by the detached process.

You can manually run the CSIPDMTCP_nnnn.EXE program and be prompted for a TCPIP communication port number by defining the logical name CSIPDMTCP_PROMPTPORT. Under normal operating conditions, this setting is never necessary and should be considered for use only under the direction of Cincom Support.

You can cause the CSIPDMTCP_nnnn.EXE program to behave similarly to csistr on UNIX by defining the logical name CSIPDMTCP_EXITONSINOF.

Logical name	Definition
\$ DEFINE CSIPDMTCP_TRACECALLS TRUE	Enables logging of internal procedure calls.
\$ DEFINE CSIPDMTCP_DUMPPACKETS TRUE	Enables logging of network packets in hexadecimal.
\$ DEFINE CSIPDMTCP_PROMPTPORT TRUE	Switches from using UCX Auxiliary Server to prompting stdin for port.
\$ DEFINE CSIPDMTCP_EXITONSINOF TRUE	Causes this program to exit after a SINOF, failed SINON, ENDT, or DYNS.

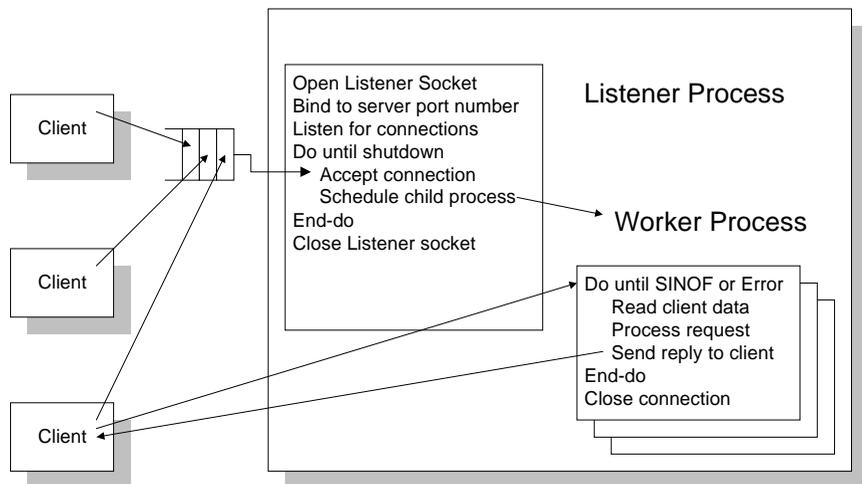
Tailoring and starting the UNIX server

SUPRA Server PDM TCP/IP program

SUPRA Server PDM for UNIX provides a TCP/IP client/server architecture as a standard feature. The server component, csistr, runs on the UNIX host machine and can connect a client to any PDM system running on that machine.

Configuring a UNIX server for SUPRA Server PDM Windows Clients

The SUPRA Server PDM Csistr Daemon, csistr, like the generic UNIX TCP/IP Server, Internet Daemon, inetd, is a listener/server program that is connection-oriented and is machine wide. It listens on the port associated with the supra1 service. When a connection is made, it forks a child/worker process which opens a socket to handle the request. After all requests are handled the socket is closed and the child process terminates (see the following figure). In addition, its application is SUPRA Server PDM-specific, as it additionally starts up the SUPRA Server PDM if CSI_AUTOSTART is set to YES, and calls Datbas on the server on the behalf of the client.



The SUPRA Server PDM TCP/IP Server for UNIX (csistr) is implemented as a UNIX TCP/IP service similar to FTP and Telnet.

The requirements to implement TCP/IP server support are as follows:

- ◆ Define the supra1 service in /etc/services as supra1 with port 8000:

```
supra1 8000/tcp
```

- ◆ If the port value 8000 conflicts with any existing service, you may override this value by specifying a different value for the supra1 service in /etc/services:

```
supra1 10000/tcp
```

- ◆ Although not necessary, if you must run more than one Csistr Daemon, you can override the supra1 service name by using the CSI_SERVICE logical name:

```
csideflog -g CSI_SERVICE service-name
```

This allows additional services to be added to the /etc/services file which can be used by the Csistr Daemon.

- ◆ The UNIX environment variable CSIRESOURCES must be defined:

```
CSIRESOURCES=/supral/pdm-system-name/slres
```

- ◆ The database logical names and PDM system name must be defined:

```
csideflog -g dbname /path/dbname.mod
csideflog -g dbname_CSI_PDM_MACS host-name
csideflog -g CSI_PDMID pdm-system-name
```

- ◆ If databases are prefixed, the logical name CSI_PREFIX must be defined:

```
csideflog -g CSI_PREFIX xxx
```

- ◆ Start the SUPRA Server PDM csistr daemon by the ROOT user at the same time as other SUPRA Server PDM daemons, csitidy, csichkpriv, csioper and csipdm. Because the csistr daemon refers to several logical names, execute it after defining the logical names. The daemon_startup and pdm_startup scripts can start up the csistr daemon if executed by the ROOT user. These scripts automatically set up the CSIRESOURCES and the CSI_PREFIX environment variables and the CSIPDM logical names. The database logical names must be added manually. Refer to the *SUPRA Server PDM UNIX PDM Installation Guide*, P25-1008.

For debugging, refer to the csistr.log file in the CSIRESOURCES directory. For additional information, refer to the *SUPRA Server PDM System Administration Guide (UNIX)*, P25-0132.

3

Programming with Windows Client Support

This chapter discusses building C, C++, COBOL, Visual Basic, and Java programs, data conversions, entry points and DML considerations.

Building C, C++, and COBOL programs

When building an MSVC++ application executable on a PC, the `datbas.lib` file must be included in the link operation. Other languages, such as Visual Basic and Microfocus COBOL, do not require use of the `datbas.lib` file.

The following table shows the header and copy books needed:

Language	Need
C	<code>#include "datbas.h"</code>
C++	<code>#include "datbas.h"</code>

For an example of COBOL code for the IBM platform, refer to the [SUPRA Server PDM DML Programming Guide \(OS/390 & VSE\)](#), P26-4340.

For an example of COBOL code for the VMS and UNIX platforms, refer to the [SUPRA Server PDM Programming Guide \(UNIX & VMS\)](#), P25-0240.

Building Visual Basic programs

If your project is going to make calls to SUPRA, then file `datbas.bas` should be included in the project.

`Datbas.bas` includes definitions, constants, and declares making use of the `Datbas.dll` easier. The definitions may be modified at your location as needed. The declare statements should not be modified except as requested by Cincom Support.

Because this module is included in the project, the variables included are available to any form or class in your project.



The calls to `Datbas` are all by reference. Therefore, the `Datbas.dll` will actually modify data in your program area. Be sure that the area you pass is large enough to hold all the information that will be returned.

Some fields that SUPRA uses can contain text as well as numeric data (refer). On a variable read, you need to place `LKXX` into refer to start reading at the head or tail of a chain. For this type of field to be useful to VB, it has to be defined as type `LONG` which is a 4 byte binary field. This means that you cannot place the string `LKXX` into this field. Notice that in the `Datbas.bas` modules, you will find `CONST` type definitions for the fields `lkxx`, `endp`, `key_equal`, `link`, `asgn`, and `last`. Any time you need to place or check a variable defined as long in one of these values, you should use a constant from `Datbas.bas` to do so.

Examples:

```
supra_refer = lkxx
supra_comitid = asgn
if supra_refer = endp then
```

All the callable functions defined in `Datbas.dll` are declared in `datbas.bas`. For information about the functions, see “[Entry points](#)“ on page 54.

There are several sample programs in the `source\vb` folder. These programs are actual working samples. You should be able to find code that uses most of the `datbas` functions (`READM`, `READV`, `ADD-M`, `ETC`) in these samples. For VB, the code is in a `.frm` type file.

Data types that are supported by SUPRA but not VB must be converted to a data type that VB can understand. Two functions to convert from SUPRA types to Text for VB are supplied. For additional information, see “[datbasConvertToText\(indata, outtext, type, sign, length, decimals\)](#)” on page 58 and “[datbasConvertFromText\(intext, outdata, type, sign, length, decimals\)](#)” on page 59. After the data is converted to text, use VB functions to convert this to any type VB understands. Before the data is written back to SUPRA, you must convert this back to the data type that SUPRA needs. A function to convert text back to the SUPRA data types is also supplied. The types not supported by VB are listed below:

- ◆ Packed - Cobol comp-3
- ◆ Zoned signed - Cobol S9
- ◆ Numeric Trailing Over Punch Sign
- ◆ Numeric Leading Sign

The area used to read the data into must have the unsupported data types defined as fixed length strings. For example, a 4 byte packed data field would be defined as string * 4. The variable used to hold the converted value must always be larger than the sending variable. For zoned, numeric trailing, and numeric leading, the length of the receiving variable should be the length of the sending variable + 2. For packed, the receiving variable must be 2 * sending variable length + 1.

Because of memory alignment in user type definitions, binary data may also have to be converted to text to be used by VB. SUPRA returns the data to VB without considering alignment. If this is the case with the data you requested, then binary must also be converted to text. SUPRA supports the following binary types. All types can be signed or unsigned.

SUPRA definition	VB initial	Text definition
1 byte	as string * 1	as string * 5
2 byte	as string * 2	as string * 7
4 byte	as string * 4	as string * 12
8 byte	as string * 8	as string * 21

Conversion is supplied for the following types:

- C Character
- B Binary
- F Floating point
- Z Zoned
- P Packed
- L * Numeric leading sign
- N * Numeric trailing over punch sign

* SUPRA VMS and UNIX only.

Building Java programs

To build a Java application on the PC, the `datbas.ini` file must exist in the current working directory of the running program. The `datbas.dll` file must be in your `PATH` as described elsewhere in this manual.

- ◆ `datbasjni.dll` contains a C written Java Native Interface. The directory containing this DLL needs to be included in your `PATH` environment variable.
- ◆ `datbasjni.jar` provides `datbas` class for use by applications. The directory containing this JAR needs to be included in your `CLASSPATH` environment variable.
- ◆ `"import datbasjni.datbas"` is used to get `datbas` class from `datbasjni.jar`. This line must be included in your Java source code.
- ◆ `"datbas db = new datbas();"` is used to make an instance of class `datbas`. This line must be included in your Java source code.
- ◆ Your Java source code must call the `datbas()` method corresponding to the number of arguments in the call. Use `db.datbas4()` for 4 arguments, `db.datbas5()` for 5 arguments, etc. Note that all arguments must be of type `byte[]`, *not* `char[]`. See example Java code in file `datbascallFrame.java`.

- ◆ Here's a way to get a Java `int` from the middle of a `datbas()` `dataarea`:

```
byte[] baDataarea = new byte[4096];
|
int DATAAREABEGIN = 10;
int nValue = 0;
int base = 1;
for( int i = DATAAREABEGIN; i < DATAAREABEGIN+4; i++ )
{
    nValue += base * baDataarea[i];
    base *= 256;
}
```

and here's a way to put a Java `int` into the middle of a `datbas()` `dataarea`:

```
byte[] baDataarea = new byte[4096];
|
int nValue = <some-value>;
int DATAAREABEGIN = 10;
baDataarea[DATAAREABEGIN+0] = (byte)((nValue & 0XFF000000) >> 24);
baDataarea[DATAAREABEGIN+1] = (byte)((nValue & 0X00FF0000) >> 16);
baDataarea[DATAAREABEGIN+2] = (byte)((nValue & 0X0000FF00) >> 8);
baDataarea[DATAAREABEGIN+3] = (byte)((nValue & 0X000000FF) >> 0);
```

There are similar ways to do other data types. This is left to the application developer.

Data conversion

Data values have various formats on different host server platforms. The following sections describe how these formats are automatically converted to the Windows client format. However, some Windows client programming languages, such as C and BASIC, do not have support for certain data types, such as packed and zoned. Therefore, manual conversion routines that are capable of converting any supported type into a text string usable by any programming language have been provided.

Manual data conversions

To convert a value of any supported data type to or from text, see the routines for “[datbasConvertToText\(\)](#)” on page 58 and “[datbasConvertFromText\(\)](#)” on page 59.

Automatic data conversions

Values in all arguments in a `datbas()` call are automatically converted between PC client and host server formats. These conversions occur automatically and transparently to the PC calling application. PC application values are converted into a network packet which is sent to the host server, and results from the host server are converted to PC format before being returned to the PC application. The following automatic conversions take place:

- ◆ **Character.** EBCDIC \leftrightarrow ASCII conversions are performed for IBM host servers for text string arguments such as `dml` function, return status, end parameter, etc.
- ◆ **Binary numbers.** Byte reversals are performed between little endian and big endian formats when the client and host are different.
- ◆ **Floating point.** Wintel platforms use IEEE floating point format. Data values are converted to/from DEC VAX floating point format or IBM floating point format as needed.
- ◆ **Packed decimal.** No conversion and no byte reversal is performed nor needed.
- ◆ **Zoned decimal.** EBCDIC \leftrightarrow ASCII conversion performed as needed with special attention given to the sign character.
- ◆ **Leading Separate Numeric.** Treated like character types with EBCDIC \leftrightarrow ASCII conversion performed as needed.
- ◆ **Numeric Trailing Overpunch.** EBCDIC \leftrightarrow ASCII conversion performed as needed with special attention given to the sign character.

Data area arguments have their contents converted field by field according to the data types of the elements contained in the element list. This can only take place if each element in the element list has a data type specified for it in the database description. Otherwise automatic data conversion is not possible for that element and results from the host server are returned to the PC application unconverted (other elements in the same element list can still be converted).

The data types are given in the following tables using ANSI C types to define the data types supported for client applications.

IBM types			ANSI C types	
Type	Sign	Length	Decimals	Type
character		1–32767	0	char
binary	signed, unsigned	1	0–2	char (MS __int8)
binary	signed, unsigned	2	0–4	int (MS __int16)
binary	signed, unsigned	4	0–9	long (MS __int32)
binary	signed, unsigned	8	0–18	(MS __int64)
float	signed	4	0	float
float	signed	8	0	double
packed decimal	signed, unsigned	$n = 1-16$	$0-(2n-1)$, 31 max	none
zoned decimal	signed, unsigned	$n = 1-18$	$0-(n)$, 18 max	none

OpenVMS & UNIX types			ANSI C types	
Type	Sign	Length	Decimals	Type
character		1–32767 Unix 1–4096 OpenVMS	0	char
binary	signed, unsigned	1	0–2	char (MS __int8)
binary	signed, unsigned	2	0–4	int (MS __int16)
binary	signed, unsigned	4	0–9	long (MS __int32)
binary	signed, unsigned	8	0–18	(MS __int64)
floating	signed	4	0	float
floating	signed	8	0	double
packed decimal	signed	$n = 1-10$	$0-(2n-1)$, 19 max	none
zoned decimal	signed, unsigned	$n = 1-18$	$0-(n)$, 18 max	none
leading separate numeric	signed, unsigned	$n = 1-18$	$0-(n-1)$, 17 max	none
numeric trailing overpunch	signed, unsigned	$n = 1-18$	$0-(n)$, 18 max	none

Entry points

The following `datbas.dll` entry points are described:

```
void DATBAS( function, status, ... )
void datbas( function, status, ... )
void datbas3( function, status, arg3 )
void datbas4( function, status, arg3, arg4 )
void datbas5( function, status, arg3, arg4, arg5 )
void datbas6( function, status, arg3, arg4, arg5, arg6 )
void datbas7( function, status, arg3, arg4, arg5, arg6, arg7 )
void datbas8( function, status, arg3, arg4, arg5, arg6, arg7, arg8 )
void datbas9( function, status, arg3, arg4, arg5, arg6, arg7, arg8, arg9 )
void datbasID( id, function, status, ... )
void datbasID3( id, function, status, arg3 )
void datbasID4( id, function, status, arg3, arg4 )
void datbasID5( id, function, status, arg3, arg4, arg5 )
void datbasID6( id, function, status, arg3, arg4, arg5, arg6 )
void datbasID7( id, function, status, arg3, arg4, arg5, arg6, arg7 )
void datbasID8( id, function, status, arg3, arg4, arg5, arg6, arg7, arg8 )
void datbasID9( id, function, status, arg3, arg4, arg5, arg6, arg7, arg8,
    arg9 )
DATBASID datbasGetCurrentID()
bool datbasSetCurrentID( id )
int datbas_putenv( envvar )
DATBASID datbasConnectID( domessagebox, servertype, servername, transport,
    portinfo, csipdmid, scope, group, user, prefix, transaction )
DATBASID datbasGetConnectParams( domessagebox, servertype, servername,
    transport, portinfo, csipdmid, scope, group, user, prefix, transaction )
bool datbasDisconnectID( id )
bool datbasConvertToText( indata, outtext, type, sign, length, decimals )
bool datbasConvertFromText( intext, outdata, type, sign, length, decimals )
```

void DATBAS(function, status, ...)**void datbas(function, status, ...)**

- Description** These generic datbas() functions are the same as the corresponding functions as described in the *SUPRA Server PDM Programming Guide (UNIX & VMS)*, P25-0240 and in the *SUPRA Server PDM DML Programming Guide (OS/390 & VSE)*, P26-4340.
- Format** Accepts a varying number of arguments between 3 and 9 of varying types and lengths depending upon the function specified in the first argument.
- Consideration** In a multiple connect application any calls to the datbas() function, which does not use an id argument, uses the value of the previously set id. For additional information, see “[bool datbasSetCurrentID \(id\)](#)” on page 56.

void datbasn(function, status, argn)

- Description** These 7 functions are provided for use with languages that do not support calls to functions that accept varying numbers of arguments. The usage is the same as for the generic datbas() function except that the appropriate datbasn() function must be called where *n* matches the number of arguments in the call. For example, both of the following calls are treated identically and return the same results:

```
datbas( "SINON", caReturnStatus, caOptionList, "END." )  
datbas4( "SINON", caReturnStatus, caOptionList, "END." )
```

The results for calling datbasn() where *n* does not match the number of arguments are unpredictable.

void datbasID(id, function, status, ...)

- Description** This is the same as the datbas() functions with the addition of the id argument. The id argument is used to specify which of several simultaneous connections is to be used for the datbas operation. Note that datbasConnectID() must be called successfully before datbasID() can be called with a particular id value.

void datbasIDn(id, function, status, argn)

- Description** These are the same as the datbasn() functions with the addition of the id argument. The id argument is used to specify which of several simultaneous connections is to be used for the datbas operation. Note that datbasConnectID() must be called successfully before datbasIDn() can be called with a particular id value.

DATBASID datbasGetCurrentID()

Description Returns the 4-byte integer value of the current id, 0 if no id has ever been used.

bool datbasSetCurrentID(id)

Description Attempts to set the 4-byte integer value of the current id where the value of the id argument must have been previously returned by a call to datbasConnectID(). It returns TRUE if successful, FALSE if not. It is not necessary to ever use this function since a call to datbasID() performs an implicit datbasSetCurrentID() before proceeding with its datbas operation. However, it may be useful when it is preferred to use calls to datbas() rather than calls to datbasID(). Any calls to a datbas() function that do not use an id argument use the value of the most recently set id.

int datbas_putenv(envvar)

Description Replaces the Windows putenv() function. This function is needed because datbas.dll may be in a different environment where calls to the Windows putenv() would have no affect on the datbas.dll environment. It can be used to programmatically turn the serial readahead cache feature on and off.

DATBASID datbasConnectID(domessagebox, servertype, servername, transport, portinfo, csipdmid, scope, group, user, prefix, transaction)

Description Specifies all the parameters for a new connection to a PDM server. When datbasConnectID() is used, no datbas.ini file is required nor used. The 11 arguments 'domessagebox' through 'transaction' are character array pointers and replace the 11 corresponding entries in the datbas.ini file and are fully described in that section. datbasConnectID() must be called to acquire the id before any calls to datbasID().

Returns a 4-byte integer representing a unique id to be used in subsequent calls to datbasID(). For example:

```
nId = datbasConnectID( ... );  
if( nId == 0 )  
    do error handling...  
datbasID( nId, ... );
```

DATBASID datbasGetConnectParams(domessagebox, servertype, servername, transport, portinfo, csipdmid, scope, group, user, prefix, transaction)

Description Fetches all the parameters for the current connection to a PDM server. The 11 arguments 'domessagebox' through 'transaction' are character array pointers corresponding to the 11 entries in the datbas.ini file and are fully described in that section.

Returns a 4-byte integer representing a unique id to be used in subsequent calls to datbasID(). For example:

```
nId = datbasGetConnectParams( ... );  
if( nId == 0 )  
    do error handling...  
datbasID( nId, ... );
```

bool datbasDisconnectID(id)

Description Disconnects from the remote PDM server specified by the connection id, and destructs the internal objects used to hold the state of the connection.

bool datbasConvertToText(indata, outtext, type, sign, length, decimals)

Description This is a data conversion routine provided for use by programming languages that do not support certain data types. The routine will convert any data type to its text string equivalent. The 'type' argument is a single character representing any of the following data types:

Type argument	Data type
C	Character
B	Binary
F	Floating point
Z	Zoned decimal
P	Packed decimal
L	Leading separate numeric
N	Numeric trailing overpunch

For example:

```
datbas( "RDNXT", caStatus, "ZONE", refer, "ZONE0302END.", caDataarea, "END."
);
if( caStatus[0] != '*' )
    do error handling...
bStatus = datbasConvertToText( caDataarea, caOutText, 'Z', 'S', 3, 2 );
if( !bStatus )
    do error handling...
printf( "The zoned value is %s", caOutText );
```

bool datbasConvertFromText(intext, outdata, type, sign, length, decimals)

Description This is a data conversion routine provided for use by programming languages that do not support certain data types. The routine will convert a text string to any data type equivalent. The 'type' argument is a single character representing any of the same data types as for the datbasConvertToText() function.



Some language compilers may require the use of the mangled form of the dll entry points. If you experience difficulties linking to the documented entry points then try the following names:

```

DATBAS
datbas
_datbas3@12
_datbas4@16
_datbas5@20
_datbas6@24
_datbas7@28
_datbas8@32
_datbas9@36
datbasID
_datbasID3@16
_datbasID4@20
_datbasID5@24
_datbasID6@28
_datbasID7@32
_datbasID8@36
_datbasID9@40
_datbas_putenv@4
_datbasConnectID@44
_datbasDisconnectID@4
_datbasGetCurrentID@0
_datbasSetCurrentID@8
_datbasGetConnectParams@44
_datbasConvertToText@24
_datbasConvertFromText@24

```

DML considerations

When using Windows Client Support with coded record element lists on both OpenVMS and Unix, there can be only one record code per element list. The same restriction applies to IBM OS/390 except when ****CODE**** is used, in which case multiple record codes may be present.



WRITD is a recovery function and is not supported.

Sample coding techniques



Error handling has been omitted for brevity.

Turning READAHEAD off/on

```
datbas_putenv( "CSI_READAHEAD=FALSE" );  
nID = datbasConnectID( ... );  
datbas_putenv( "CSI_READAHEAD=TRUE" );
```

Using datbasConnectID to connect to the host (without using datbas.ini)

```
nID = datbasConnectID( ... );  
datbasID( nID, SINON, status, ... );
```

Using multiple datbasConnectID functions to SINON to multiple databases from the same application

```
ID0001 = datbasConnectID( ... );  
DatbasID4( ID0001, SINON, status, schema1, endp );  
ID0002 = datbasConnectID( ... );  
DatbasID4( ID0002, SINON, status, schema2, endp );
```

Using `datbasSetCurrentID`

```
ID0001 = datbasConnectID( ... );  
datbasID4( ID0001, SINON, status, schema1, endp );  
ID0002 = datbasConnectID( ... );  
datbasID4( ID0002, SINON, status, schema2, endp );  
datbasSetCurrentID( ID0001 );  
datbas( READM, status, file, key, list, area, endp );  
datbas( SINOF, status, schema1, endp );  
datbasID4( ID0002, SINOF, status, schema2, endp );
```

Using `datbasGetCurrentID`

```
nID = datbasGetCurrentID();  
datbasID( nID, RDNXT, file, status, qualifier, list, area, endp );
```

Using `datbasConvertToText`

```
datbas( RDNXT, status, file, status, list, area, endp );  
status = datbasConvertToText ( area, outText, 'Z', 'S', 3, 2 );
```

4

Error messages

SUPRA PDM Windows Client messages

- CSTJ100F** Windows AfxSocketInit() failed
- Explanation** The call to the Microsoft socket initialization routine has failed.
- User action** Be sure that TCP/IP is properly installed on your system. Refer to Microsoft documentation.
- CSTJ101F** new() failed for <object> in <class>::<method>()
- Explanation** A memory allocation request for the specified <object> has failed.
- User action** Your system may need more virtual memory. Investigate the possibility of increasing the size of the pagefile.sys file.
- CSTJ102F** Invalid <entry-type>: <entry>
In init file .\datbas.ini, section [datbas]
Valid <entry-type>s are <entry-list..> (1)
Character '<bad-char>' is not a digit ~or~ (2)
Length must be 3 or less characters ~or~ (3)
Length must be 1 to 4 characters ~or~ (4)
Transaction only valid for IBM CICS ServerType ~or~ (5)
- Explanation** An invalid value has been found for the specified <entry-type> in the datbas.ini file or in the equivalent call to the datbasConnectID() function. Only one of the above five lines is issued as the third line of this message.
- User action** Correct the entry in the datbas.ini file or the datbasConnectID() call.

- CSTJ103F** Missing <entry-type> entry
In init file .\datbas.ini, section [datbas]
- Explanation** No value has been found for the specified <entry-type> in the datbas.ini file or in the equivalent call to the datbasConnectID() function.
- User action** Add the entry in the datbas.ini file or the datbasConnectID() call.
- CSTJ104E** Status <dml-status> returned for <dml-function> in <class>::<method>
option-list = <option-list>, data-list = <data-list>
- Explanation** An internal datbas() call has failed. A second line for this message, giving the option-list and data-list, is only displayed if the <dml-function> is SHOWX.
- User action** See the PDM Server Messages and Codes Manual for the action for the specified <dml-status>.
- CSTJ105F** Socket create error: <error-text>
- Explanation** An internal call to the MFC CSocket Create() function has failed to create a TCP/IP socket for communicating to the remote server host machine.
- User action** See your TCP/IP documentation for the action for the specified <error-text>.
- CSTJ106E** Socket connect error: <error-text>
Connecting to server <servername>, port <portinfo>
- Explanation** An internal call to the MFC CSocket Connect() function has failed.
- User action** See your TCP/IP documentation for the action for the specified <error-text>.

- CSTJ107E** PDM connect error: <connect-status>
Connecting to server <servername>, port <portinfo>
'<connect-string>'
- Explanation** An error connect status has been returned from the remote server in response to the <connect-string>. The possible return statuses are:
- ◆ "TIME" The attempt to connect has timed out.
 - ◆ "WAIT" The connection is in progress.
 - ◆ "XMIT" Retransmit the message, there was a possible transmission error.
 - ◆ "FAIL" The connection has failed.
 - ◆ "GOOD" A good connect does not produce this message.
- User action** For "TIME", "WAIT", and "XMIT" retry the operation. For "FAIL" there is most likely a bad entry in the <connect-string>. Since this string is built from the entries in the datbas.ini file, or the corresponding datbasConnectID() call, you should verify all those entries for correct values.
- CSTJ108E** Socket send error: <error-text>
Sent <sent-count> out of <send-length> bytes
- Explanation** An internal call to the MFC CSocket Send() function has failed.
- User action** See your TCP/IP documentation for the action for the specified <error-text>.
- CSTJ109E** Socket receive error: <error-text>
Received <received-count> out of <receive-length> bytes
- Explanation** An internal call to the MFC CSocket Receive() function has failed.
- User action** See your TCP/IP documentation for the action for the specified <error-text>.

- CSTJ110E** Remote PDM sent a packet length of <>, expected <>
- Explanation** The remote PDM server sent a packet of data of an unexpected length that cannot be handled.
- User action** You may need to use the debugging/tracing version of datbas.dll and examine the datbas.log file produced for more information.
- CSTJ111I**
- | SINOF statistics | <progrname> | <dbmod> | Total Calls | Buffered Calls |
|------------------|-------------|---------|-------------|----------------|
| | RDNXT | | nnnn | nnnn |
| | READV/READR | | nnnn | nnnn |
| | READX | | nnnn | nnnn |
- Explanation** The program <progrname> has signed off the database <dbmod> with these serial readahead cache statistics. A buffered call is one for which the Windows Client did not communicate with the PDM server, but instead retrieved the data from its local cache. This message is enabled and disabled by defining the environment variable CSI_READAHEAD_STATISTICS as TRUE or FALSE respectively.
- User action** None, informational only.
- CSTJ112E** CSI_READAHEAD has been disabled due to previous error.
- Explanation** The serial readahead cache feature has been dynamically disabled due to the previous error message.
- User action** See the description for the previous error message.
- CSTJ113F** Security patch has expired on dd-mmm-yyyy
- Explanation** The security codes defined by the environment variable CSIPDM_PATCH have expired and the product cannot be used.
- User action** Contact your Cincom account representative to acquire new security codes.
- CSTJ114I** Security patch will expire on dd-mmm-yyyy
- Explanation** The security codes defined by the environment variable CSIPDM_PATCH will expire on the indicated date.
- User action** None, informational only.

CSTJ114W	Security patch will expire on <i>dd-mmm-yyyy</i>
Explanation	The security codes defined by the environment variable CSIPDM_PATCH will expire soon. After that date the product cannot be used.
User action	Contact your Cincom account representative to acquire new security codes.
CSTJ115F	Invalid Security patch
Explanation	The security codes defined by the environment variable CSIPDM_PATCH are invalid and the product cannot be used.
User action	Contact your Cincom account representative to acquire new security codes.
CSTJ116F	Security patch not defined
Explanation	The environment variable CSIPDM_PATCH, which defines the security codes, is not defined.
User action	Contact your Cincom account representative to acquire new security codes.
CSTJ117I	Init file found: <code>.\datbas.ini</code> in current directory: <code><drive-letter>:<directory></code>
Explanation	The datbas.ini file containing initialization settings has been found in the specified current directory.
User action	None, informational only.
CSTJ118F	Init file not found: <code>.\datbas.ini</code> in current directory: <code><drive-letter>:<directory></code>
Explanation	The datbas.ini file containing initialization settings could not be found in the specified current directory. Datbas only looks for this file in the current working directory.
User action	Either place the datbas.ini file in the current directory, or change the current working directory to be the same as the location of the datbas.ini file, or use the datbasConnectID() entry point within the application program to prevent use of a datbas.ini file.

SUPRA PDM Windows Server messages

The following messages can be returned by the SUPRA PDM Windows Server components:

- CSTJ450I** The list of initialization parameters follows.
- Description** A list of all initialization parameters and their current values follows this message.
- User action** None.
- CSTJ451I** Parameter *parameter-name* *parameter-value*.
- Description** This message is used to display the value of a string initialization parameter.
- User action** None.
- CSTJ452I** Parameter *parameter-name* *parameter-value*.
- Description** This message is used to display the value of a numeric initialization parameter.
- User action** None.
- CSTJ453I** Parameter *parameter-name*FALSE.
- Description** This message is used to display the value of a Boolean parameter with the value FALSE.
- User action** None.
- CSTJ454I** Parameter *parameter-name*TRUE.
- Description** This message is used to display the value of a Boolean parameter with the value TRUE.
- User action** None.

- CSTJ456W** Cannot open *file-name* for read. Errno: *nnn*
- Description** The specified file cannot be opened.
- User action** Check the JCL.
- CSTJ457W** Invalid keyword *keyword-name* on line *line-number*.
- Description** The specified keyword is unknown. It is located at the specified line in the INIT file.
- User action** Correct the keyword.
- CSTJ458W** Invalid numeric parameter *parameter-name* on line *line-number*.
- Description** The value in the specified parameter on the specified line is not numeric.
- User action** Correct the parameter.
- CSTJ459W** Input parameter *parameter-name* on line *line-number* is longer than max.
- Description** The value of the specified parameter on the specified line is larger than the allowed maximum.
- User action** Correct the value.
- CSTJ465W** Invalid boolean value *parameter-name* on line *line-number* .
- Description** The specified parameter on the specified line has an unknown boolean value.
- User action** Correct the value to be Yes/No/True/False/1/0.
- CSTJ467E** Parameter *parameter-name* value *value* on line *line-number* out of range. Min: *min-value*. Max: *max-value*.
- Description** The specified parameter has a value outside the legal range.
- User action** Correct the parameter.

- CSTJ500I** SUPRA Server V1 Driver Release *n.n.nn.nx* started.
- Description** The driver has started.
- User action** None. Information only.
- CSTJ502I** SUPRA Server V1 Driver Release *n.n.nn.nx* terminated.
- Description** The driver has terminated. This is the last message from the V1 Driver.
- User action** None. Information only.
- CSTJ515E** Load failed for *module-name*. Reason: *reason-string*
- Description** Load of the specified module failed.
- User action** Use the reason string to determine and correct the problem. The reason string specifies what operating system request failed, and the return codes. The problem is usually an environment problem, such as incorrect JCL, not enough memory, and so on.
- CSTJ521E** Unexpected Status from PDM function *function-name*.
Stat: *status-code*
- Description** The PDM returned an unexpected status from the specified function call.
- User action** Use the status to help determine and correct the problem. You may need to contact Cincom Technical Support.
- CSTJ522I** SUPRA Server V1 Driver Release *n.n.nn.nx* waiting for connections.
- Description** The driver has finished initialization and is waiting for connections.
- User action** None. Information only.

- CSTJ523I** *requested-bytes* of shared memory were requested.
used-bytes were used.
- Description** Information only. This message indicates how much shared memory was used by the driver. You can use this value to tune the SHARED_MEM parameter in the initialization file.
- User action** None. Information only.
- CSTJ524I** Database Name: *database-name*
- Description** Information only. Specifies the name of the database connected to this driver.
- User action** None. Information only.
- CSTJ550E** *function-name* error. Errno: *nnn*. Server PID: *nnn*
- Description** The TCP communication processor socket library returned an error when performing the specified function.
- User action** Use the error number and error text to determine and correct the problem. You may have to contact Cincom Technical Support.
- CSTJ551E** *function-name* error on *socket-name* socket. Errno: *nnn*.
Server PID: *nnn*
- Description** The TCP socket library returned an error when performing the specified function on the specified socket.
- User action** Use the error number and error text to determine and correct the problem. You may have to contact Cincom Technical Support.
- CSTJ552I** Connection request received from *host-name*. Port:
port-number
- Description** A connection request was received from the specified host on the specified port.
- User action** None. Information only.

CSTJ553I Connection rejected for *host-name*. Reason *nnn reason-text*

Description A connection request from the specified host was rejected.

User action None. Information only.

CSTJ556E Expected data base name: *expected*. Actual database name: *actual*

Description If the connection was rejected because of a database name mismatch, the expected and actual names are printed in this message.

User action None. Information only.

CSTJ557I Session canceled for *hostname*. Server PID *nnn*

Description The session was canceled by the server.

User action None. Information only.

CSTJ600E Unable to allocate memory for receive PDU. Requested size: *nnn*. Server PID: *nnn*

Description There is not enough memory to allocate a receive buffer.

User action Increase the region size on the JOB or EXEC card.

CSTJ602E FAP version mismatch. Received *nnn*. Expected *nnn*

Description The client and server are at different, incompatible service levels.

User action Check the installation and correct whichever is at the incorrect level.

- CSTJ603E** *pdm-function* returned unexpected status. Stat: *status*.
Server PID: *nnn*
- Description** The specified PDM function returned an unexpected status.
- User action** Use the error status to determine and correct the problem. You may have to contact Cincom Technical Support.
- CSTJ606I** Extended status: *status-code*. Server PID: *nnn*
- Description** The PDM returned an extended status code. This message comes out in conjunction with message 603E.
- User action** Use the error status to determine and correct the problem. You may have to contact Cincom Technical Support.
- CSTI350I** Remote connection daemon CSIPDMTCP %s running on node %s
- Description** The SUPRA PDM Windows Server has successfully been started on an OpenVMS machine.
- User action** None, information only.
- CSTI351F** Failed to get host name
- Description** The gethostname() function has failed on the server machine and the SUPRA PDM Windows Server cannot continue.
- User action** This function is external to Cincom code. Either there is a problem with the TCP/IP library or the hostname exceeds the maximum of 255 characters. See the accompanying error message for more detail.

- CSTI352I** Remote connection daemon CSIPDMTCP %s terminated
- Description** The SUPRA PDM Windows Server has successfully been terminated on an OpenVMS machine.
- User action** None, information only.
- CSTI353F** Failed to get socket from UCX Auxiliary server
- Description** The function call `socket(UCX$C_AUXS,SOCK_STREAM,0)` has failed and the SUPRA PDM Windows Server cannot continue.
- User action** See the accompanying error message for more detail.
- CSTI354F** Failed to set socket option for KEEPALIVE
- Description** The function call `setsockopt(sd,SOL_SOCKET,SO_KEEPALIVE,...)` has failed and the SUPRA PDM Windows Server cannot continue.
- User action** See the accompanying error message for more detail.
- CSTI355F** Error reading data from remote task
- Description** The function call `recv()` has failed and the SUPRA PDM Windows Server cannot continue.
- User action** See the accompanying error message for more detail.
- CSTI356F** `recv()` function returned %d bytes when %d were needed
- Description** A network packet has been received for which the actual length is not the stated length.
- User action** This condition indicates a network communications problem. Or it may indicate that the remote client has abended and/or severed the socket connection.

- CSTI357F** Received buffer size %d too large for internal buffer size %d
- Description** A network packet is larger than the maximum internal buffer size of 32768 bytes.
- User action** A client application call to datbas() probably has inappropriate parameters. Define the logical name csipdmtcp_dump packets as TRUE, rerun the application, and investigate the contents of the csipdmtcp log file. Also, you can use the datbasd.dll to make a datbas.log trace file on the client side.
- CSTI358E** Unknown print_sysmsg() type %d
- Description** This is an internal error.
- User action** Report the error to your Cincom support center.
- CSTI359I** Connection packet from client is: "%s"
- Description** A connect string has been received from the client.
- User action** None, informational only.
- CSTI360I** Received good connection packet from client
- Description** The connect string from the client has been verified and accepted.
- User action** None, informational only.

- CSTI361F** Rejected client because node "%s" is not local host "%s"
- Description** The connect string from the client has been rejected because it comes from the same node the server is running on.
- User action** Only attempt to connect TCP/IP clients from remote clients.
- CSTI362F** Received bad connection packet from client
- Description** The connect string from the client has been rejected because it is incomplete.
- User action** Retry the connection attempt. If the error persists then contact your Cincom support center.
- CSTI363E** Error writing data to remote task
- Description** The function call send() has failed and the SUPRA PDM Windows Server cannot continue.
- User action** See the accompanying error message for more detail.
- CSTI364E** Error writing data to remote task %d sent out of %d
- Description** A network packet was incompletely sent using the send() function.
- User action** This condition indicates a network communications problem. Or it may indicate that the remote client has abended and/or severed the socket connection.
- CSTI365E** Error defining logical name %s
- Description** A call to the system service sys\$crelnm() has failed.
- User action** See the accompanying error message for more detail.

CSTI366F	Failed to activate datbas()
	<p>Description A call to the system library routine lib\$find_image_symbol() has failed to load the shareable images for SUPRAPDM.</p> <p>User action See the accompanying error message for more detail.</p>
CSTI367F	Rejected client because pdmid "%s" differs from local "%s"
	<p>Description The connect string from the client has been rejected because the csipdmid entry does not match the local csipdmid for this group.</p> <p>User action Verify that the CSIPDMID entry in the datbas.ini file, or the call to datbasConnectID(), are appropriate for this group PDM.</p>
CSTI368E	Error translating logical name %s
	<p>Description A call to the system service sys\$trnlm() has failed.</p> <p>User action See the accompanying error message for more detail.</p>
CSTI369F	Rejected client because group %06o differs from local %06o
	<p>Description The connect string from the client has been rejected because the group entry does not match the local group number.</p> <p>User action Verify that the GROUP entry in the datbas.ini file, or the call to datbasConnectID(), are appropriate for this group PDM.</p>
CSTI370E	Cannot access datbas_pointers() for parameter return lengths
	<p>Description A call to the system library routine lib\$find_image_symbol() has failed to find the entry point for the call to datbas_pointers() in the shareable images for SUPRAPDM.</p> <p>User action See the accompanying error message for more detail. Be sure you are running SUPRA/VMS 2.3 or greater.</p>

CSTI371I

EXITONSINOF -- function=%.5s, status=%.4s

Description The SUPRA PDM Windows Server has successfully terminated because the logical name csipdmtcp_exitonsinof is defined as TRUE and the SUPRA PDM Windows Client has done either SINOF, failed SINON, received ENDT status, or received DYNS status.

User action None, informational only.

5

Troubleshooting

This section provides suggestions for resolving some situations that may occur.



Always refer to "Error messages" on page 63. Also, refer to the *SUPRA Server PDM Messages and Codes Reference Manual (PDM/RDM Support for UNIX & VMS)*, P25-0022 for VMS and UNIX, and the *SUPRA Server PDM Messages and Codes Reference Manual (RDM/PDM Support for OS/390 & VSE)*, P26-0126 for IBM.

Location of the datbas.ini file

SUPRA Server PDM Windows Client Support expects the datbas.ini file to be located in the current working directory of the running application. There will be a message in the datbas.log file (CSTJ1171) with the location that is being used. If you are experiencing an error message regarding the connection to the server, determine which datbas.ini file is being used. It may not be the one that you expected.

Unable to connect to Unix server

If you are experiencing connection errors, please refer to the datbas.log file on your PC. Also, refer to the csistr.log file on the server. You may wish to reference the *SUPRA Server PDM System Administration Guide (UNIX)*, P25-0132, Chapter 2 "Connecting to a remote PDM" for information on setting up the environment to run the csistr daemon.

Debugging SUPRA Server PDM Windows Client Support

SUPRA Server PDM Windows Client Support provides a debugging mechanism that is used by Cincom Support. You may be requested to provide Cincom with a trace of Windows Client that can be obtained by executing the debug version of the datbas.dll file. You will need to rename the current datbas.dll as datbas.dll.sav, and copy datbasd.dll to datbas.dll. Then, run your application normally. A trace of the functions within Windows Client will be printed in the datbas.log file. Save this file for Cincom Support. After this is done, reinstate the non-debug version of datbas.dll. Not doing so will affect performance. Simply rename datbas.dll.sav as datbas.dll.

A

Translation tables

EBCDIC to ASCII default translation table

```
unsigned char EbcDicToAscii[256] = {
    0x00, 0x01, 0x02, 0x03, 0x9c, 0x09, 0x86, 0x7f, /* 00 - 07 */
    0x97, 0x8d, 0x8e, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, /* 08 - 0F */
    0x10, 0x11, 0x12, 0x13, 0x9d, 0x0a, 0x08, 0x87, /* 10 - 17 */
    0x18, 0x19, 0x92, 0x8f, 0x1c, 0x1d, 0x1e, 0x1f, /* 18 - 1F */
    0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x17, 0x1b, /* 20 - 27 */
    0x88, 0x89, 0x8a, 0x8b, 0x8c, 0x05, 0x06, 0x07, /* 28 - 2F */
    0x90, 0x91, 0x16, 0x93, 0x94, 0x95, 0x96, 0x04, /* 30 - 37 */
    0x98, 0x99, 0x9a, 0x9b, 0x14, 0x15, 0x9e, 0x1a, /* 38 - 3F */
    0x20, 0xa0, 0xe2, 0xe4, 0xe0, 0xe1, 0xe3, 0xe5, /* 40 - 47 */
    0xe7, 0xf1, 0xa2, 0x2e, 0x3c, 0x28, 0x2b, 0x7c, /* 48 - 4F */
    0x26, 0xe9, 0xea, 0xeb, 0xe8, 0xed, 0xee, 0xef, /* 50 - 57 */
    0xec, 0xdf, 0x21, 0x24, 0x2a, 0x29, 0x3b, 0x5e, /* 58 - 5F */
    0x2d, 0x2f, 0xc2, 0xc4, 0xc0, 0xc1, 0xc3, 0xc5, /* 60 - 67 */
    0xc7, 0xd1, 0xa6, 0x2c, 0x25, 0x5f, 0x3e, 0x3f, /* 68 - 6F */
    0xf8, 0xc9, 0xca, 0xcb, 0xc8, 0xcd, 0xce, 0xcf, /* 70 - 77 */
    0xcc, 0x60, 0x3a, 0x23, 0x40, 0x27, 0x3d, 0x22, /* 78 - 7F */
    0xd8, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, /* 80 - 87 */
    0x68, 0x69, 0xab, 0xbb, 0xf0, 0xfd, 0xfe, 0xb1, /* 88 - 8F */
    0xb0, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, 0x70, /* 90 - 97 */
    0x71, 0x72, 0xaa, 0xba, 0xe6, 0xb8, 0xc6, 0xa4, /* 98 - 9F */
    0xb5, 0x7e, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, /* A0 - A7 */
    0x79, 0x7a, 0xa1, 0xbf, 0xd0, 0x5b, 0xde, 0xae, /* A8 - AF */
    0xac, 0xa3, 0xa5, 0xb7, 0xa9, 0xa7, 0xb6, 0xbc, /* B0 - B7 */
    0xbd, 0xbe, 0xdd, 0xa8, 0xaf, 0x5d, 0xb4, 0xd7, /* B8 - BF */
    0x7b, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, /* C0 - C7 */
    0x48, 0x49, 0xad, 0xf4, 0xf6, 0xf2, 0xf3, 0xf5, /* C8 - CF */
    0x7d, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, 0x50, /* D0 - D7 */
    0x51, 0x52, 0xb9, 0xfb, 0xfc, 0xf9, 0xfa, 0xff, /* D8 - DF */
    0x5c, 0xf7, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, /* E0 - E7 */
    0x59, 0x5a, 0xb2, 0xd4, 0xd6, 0xd2, 0xd3, 0xd5, /* E8 - EF */
    0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, /* F0 - F7 */
    0x38, 0x39, 0xb3, 0xdb, 0xdc, 0xd9, 0xda, 0x9f /* F8 - FF */
};
```

ASCII to EBCDIC default translation table

```

unsigned char AsciiToEbcidic[256] = {
    0x00, 0x01, 0x02, 0x03, 0x37, 0x2d, 0x2e, 0x2f, /* 00 - 07 */
    0x16, 0x05, 0x15, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, /* 08 - 0F */
    0x10, 0x11, 0x12, 0x13, 0x3c, 0x3d, 0x32, 0x26, /* 10 - 17 */
    0x18, 0x19, 0x3f, 0x27, 0x1c, 0x1d, 0x1e, 0x1f, /* 18 - 1F */
    0x40, 0x5a, 0x7f, 0x7b, 0x5b, 0x6c, 0x50, 0x7d, /* 20 - 27 */
    0x4d, 0x5d, 0x5c, 0x4e, 0x6b, 0x60, 0x4b, 0x61, /* 28 - 2F */
    0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, /* 30 - 37 */
    0xf8, 0xf9, 0x7a, 0x5e, 0x4c, 0x7e, 0x6e, 0x6f, /* 38 - 3F */
    0x7c, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, /* 40 - 47 */
    0xc8, 0xc9, 0xd1, 0xd2, 0xd3, 0xd4, 0xd5, 0xd6, /* 48 - 4F */
    0xd7, 0xd8, 0xd9, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6, /* 50 - 57 */
    0xe7, 0xe8, 0xe9, 0xad, 0xe0, 0xbd, 0x5f, 0x6d, /* 58 - 5F */
    0x79, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, /* 60 - 67 */
    0x88, 0x89, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, /* 68 - 6F */
    0x97, 0x98, 0x99, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, /* 70 - 77 */
    0xa7, 0xa8, 0xa9, 0xc0, 0x4f, 0xd0, 0xa1, 0x07, /* 78 - 7F */
    0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x06, 0x17, /* 80 - 87 */
    0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x09, 0x0a, 0x1b, /* 88 - 8F */
    0x30, 0x31, 0x1a, 0x33, 0x34, 0x35, 0x36, 0x08, /* 90 - 97 */
    0x38, 0x39, 0x3a, 0x3b, 0x04, 0x14, 0x3e, 0xff, /* 98 - 9F */
    0x41, 0xaa, 0x4a, 0xb1, 0x9f, 0xb2, 0x6a, 0xb5, /* A0 - A7 */
    0xbb, 0xb4, 0x9a, 0x8a, 0xb0, 0xca, 0xaf, 0xbc, /* A8 - AF */
    0x90, 0x8f, 0xea, 0xfa, 0xbe, 0xa0, 0xb6, 0xb3, /* B0 - B7 */
    0x9d, 0xda, 0x9b, 0x8b, 0xb7, 0xb8, 0xb9, 0xab, /* B8 - CF */
    0x64, 0x65, 0x62, 0x66, 0x63, 0x67, 0x9e, 0x68, /* C0 - C7 */
    0x74, 0x71, 0x72, 0x73, 0x78, 0x75, 0x76, 0x77, /* C8 - DF */
    0xac, 0x69, 0xed, 0xee, 0xeb, 0xef, 0xec, 0xbf, /* D0 - D7 */
    0x80, 0xfd, 0xfe, 0xfb, 0xfc, 0xba, 0xae, 0x59, /* D8 - DF */
    0x44, 0x45, 0x42, 0x46, 0x43, 0x47, 0x9c, 0x48, /* E0 - E7 */
    0x54, 0x51, 0x52, 0x53, 0x58, 0x55, 0x56, 0x57, /* E8 - EF */
    0x8c, 0x49, 0xcd, 0xce, 0xcb, 0xcf, 0xcc, 0xe1, /* F0 - F7 */
    0x70, 0xdd, 0xde, 0xdb, 0xdc, 0x8d, 0x8e, 0xdf, /* F8 - FF */
};

```

Index

A

arguments, varying numbers of
55
ASCII to EBCDIC translation 37
automatic data conversions 51

B

binary numbers 51
binary types SUPRA supports 48

C

C and C++ programs 45
character 51
client component
 installation 20
 requirements 19
 tailoring 23, 24
COBOL programs 45
code samples 61
coded record 60
communication protocols 18
compact setup 21
concurrent connections,
 maximum number 34
concurrent threads, number of 34
configuration 18
ConnectID
 sample code 61
connection id 55, 56
CSI_PREFIX logical name 28
CSIPDM_PATCH 22
Cspidmid, datbas.ini file entry 26
csistr 43
custom setup 21

D

data area size 33
data conversion
 automatic data 51
 datbas 58, 59
 manual data 50
data types 52
data types supported 47
data types, support for client
 applications 52
datalist size 33
datbas functions
 connection id 55, 56
 current id 56
 data conversion 58, 59
 disconnect 57
 fetch current parameters 57
 generic 55
 putenv 56
 varying numbers of arguments
 55
datbas, datbas.ini file entry 25
datbas.bas 46
datbas.ini
 configuration 22
 creating 24
 troubleshooting 79
debug tracing, enabling 37
debugging, VMS 42
destination location 20
detached client process 41
disconnect, datbas 57
DISPLAY_OPTIONS, server
 parameter (OS/390) 33
displaying initialization options 33
DML considerations 60
DML support 17
DoMessageBox, datbas.ini file
 entry 25
DRVRJCL 30, 31

E

EBCDIC to ASCII translation 37
enabling Serial ReadAhead
 Cache 29
entry points 54
environment variable
 CSIPDM_PATCH 22
error messages 63
exceptions, displaying messages
 25

- F**
- fetching current parameters 57
 - floating point 51
- G**
- generic datbas functions 55
 - Group, datbas.ini file entry 27
 - groupwide PDMs 41
- H**
- host file name 36
 - host name 25
 - host name lookup 35
- I**
- implementing Windows Client support, VMS 39
 - init file section name 25
 - initialization options, displaying 33
 - initialization parameters, server component 32
 - installation
 - client component 20
 - destination 20
 - options 22
 - program folder 21
 - requirements 19
 - summary 21
 - type of setup 21
- J**
- Java programs 49
 - JCL member 30, 31
- L**
- leading separate numeric 51
 - listener port number 36
 - logging internal procedure calls 42
 - logging network packets in
 - hexadecimal 42
 - logical names 42
- M**
- manual data conversions 50
 - MAX_DATAAREA, server parameter (OS/390) 33
 - MAX_DATALIST, server parameter (OS/390) 33
 - MAX_DATBAS_PDU, server parameter (OS/390) 34
 - MAX_RCV_PDU, server parameter (OS/390) 34
 - memory, size of shared 35
 - message size, maximum 34
 - message size, specifying (OS/390) 34
 - messages
 - SUPRA PDM Windows Client 63
 - SUPRA PDM Windows Server 68
 - messages, displaying 25
- N**
- numeric trailing overpunch 51
- O**
- options, installation 22
 - overview 17, 36
- P**
- packed decimal 51
 - parameters
 - CICS server init file 33
 - password, shutdown 35
 - PDM
 - group number 27
 - name 26
 - scope 27
 - PDM Windows Client support
 - description 17
 - PDM_THREADS, server parameter (OS/390) 34
 - port number 26
 - port number, VMS 38
 - PortInfo, datbas.ini file entry 26
 - Prefix, datbas.ini file entry 28
 - processes, VMS 38
 - program folder 21

programming languages 17, 45
 programs
 C, C++, and COBOL 45
 Java 49
 Visual Basic 46
 putenv function 56

Q

qualifier, TCP/IP 36

R

READAHEAD
 sample code 61
 recovery function, WRITD 60
 reinstalling 23
 response buffer size 34

S

sample coding techniques 61
 Scope, datbas.ini file entry 27
 security codes 22
 Serial ReadAhead Cache,
 enabling 29
 server component
 tailoring (OS/390) 31
 server init file parameters (CICS)
 33
 server name 25
 server site requirements 19
 server type 25
 SERVER_CONNECTIONS,
 server parameter (OS/390)
 34
 ServerName, datbas.ini file entry
 25
 ServerType, datbas.ini file entry
 25
 shared memory size 35
 SHARED_MEM, server
 parameter (OS/390) 35
 SHUTDOWN_PASSWORD,
 server parameter (OS/390)
 35
 SHUTDOWN_USER, server
 parameter (OS/390) 35

socket communication, port
 number 26
 source vb folder 46
 supported data types 47
 supported platforms 17
 SUPRA
 supported releases 19
 SUPRA PDM Windows Client
 messages 63
 SUPRA PDM Windows Server
 messages 68

T

tailoring
 server component (OS/390) 31
 UNIX server 43
 VMS server 38
 TCP/IP sockets, VMS 38
 TCP/IP support 17
 TCP_HOSTFILE, server
 parameter (OS/390) 36
 TCP_HOSTNAME_LOOKUP,
 server parameter (OS/390)
 35
 TCP_LISTEN_PORT, server
 parameter (OS/390) 36
 TCP_PREFIX, server parameter
 (OS/390) 36
 threads, number of concurrent 34
 TOASCII, server parameter
 (OS/390) 37
 TOEBCDIC, server parameter
 (OS/390) 37
 TRACE
 server parameter (OS/390) 37
 transaction id 28
 Transaction, datbas.ini file entry
 28
 translation tables
 ASCII to EBCDIC 82
 EBCDIC to ASCII 81
 specifying 37
 transport type 26
 Transport, datbas.ini file entry 26
 troubleshooting 79
 types of setup 21
 typical setup 21

U

- uninstalling 23
- UNIX
 - server, tailoring 43
- user id 28
- User, datbas.ini file entry 28
- user, shutdown 35

V

- Visual Basic programs 46
- VMS
 - debugging 42
 - implementing TCP/IP client support 39
 - port number 38
 - processes 38
 - server, tailoring 38
 - TCP/IP sockets 38

W

- Windows platforms 17
- WNCLIENT library 30
- WRITD, recovery function 60

Z

- zoned decimal 51